Jiří Fiala
Václav Koubek
Jan Kratochvíl (Eds.)

LNCS 3153

Mathematical Foundations of Computer Science 2004

29th International Symposium, MFCS 2004
Prague, Czech Republic, August 2004
Proceedings



Springer

Lecture Notes in Computer Science 3153

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
New York University, NY, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

This page intentionally left blank

Jiří Fiala Václav Koubek
Jan Kratochvíl (Eds.)

Mathematical Foundations of Computer Science 2004

29th International Symposium, MFCS 2004
Prague, Czech Republic, August 22-27, 2004
Proceedings

Springer

eBook ISBN: 3-540-28629-2
Print ISBN: 3-540-22823-3

©2005 Springer Science + Business Media, Inc.

Print ©2004 Springer-Verlag
Berlin Heidelberg

All rights reserved

No part of this eBook may be reproduced or transmitted in any form or by any means, electronic, mechanical, recording, or otherwise, without written consent from the Publisher

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.springerlink.com
and the Springer Global Website Online at: http://www.springeronline.com

Preface

This volume contains the papers presented at the 29th Symposium on Mathematical Foundations of Computer Science, MFCS 2004, held in Prague, Czech Republic, August 22–27, 2004. The conference was organized by the Institute for Theoretical Computer Science (ITI) and the Department of Theoretical Computer Science and Mathematical Logic (KTIML) of the Faculty of Mathematics and Physics of Charles University in Prague. It was supported in part by the European Association for Theoretical Computer Science (EATCS) and the European Research Consortium for Informatics and Mathematics (ERCIM).

Traditionally, the MFCS symposia encourage high-quality research in all branches of theoretical computer science. Ranging in scope from automata, formal languages, data structures, algorithms and computational geometry to complexity theory, models of computation, and applications including computational biology, cryptography, security and artificial intelligence, the conference offers a unique opportunity to researchers from diverse areas to meet and present their results to a general audience.

The scientific program of this year's MFCS took place in the lecture halls of the recently reconstructed building of the Faculty of Mathematics and Physics in the historical center of Prague, with the famous Prague Castle and other celebrated historical monuments in sight. The view from the windows was a challenging competition for the speakers in the fight for the attention of the audience. But we did not fear the result: Due to the unusually tough competition for this year's MFCS, the admitted presentations certainly attracted considerable interest. The conference program (and the proceedings) consisted of 60 contributed papers selected by the Program Committee from a total of 167 submissions. These are accompanied in the proceedings by abstracts or full versions of the 10 invited talks. It has already become a tradition that EATCS offers a Best Student Paper Award for the best paper submitted to MFCS and authored solely by students. The winner of the award was announced during the conference.

As the editors of these proceedings, we would like to thank everyone who contributed to the success of the symposium and to its scientific merit. First of all the authors of the contributed papers for the record number of submissions, the invited speakers for accepting our invitation and sharing their knowledge and skills with us, the Program Committee members for their demanding and responsible work, their subreferees for careful reading of all the submissions, Springer-Verlag for excellent cooperation in the publication of this volume, and last but not least the Organizing Committee and Action-M Agency (our partner responsible for the local arrangements) for smooth running of the symposium. We hope the attendees all had a fruitful and enjoyable time in Prague.

August 2004

Jiří Fiala
Václav Koubek
Jan Kratochvíl

Organizers

Faculty of Mathematics and Physics, Charles University, Prague

- Institute for Theoretical Computer Science
- Department of Theoretical Computer Science and Mathematical Logic

Action-M Agency (local arrangements)

Supporters

European Association for Theoretical Computer Science (EATCS)
European Research Consortium for Informatics and Mathematics (ERCIM)
Institute for Theoretical Computer Science (ITI) — provided support for 11 students to participate at MFCS 2004

Organizing Committee

Rostislav Caha
Ondřej Čepek
Jiří Fiala
Václav Koubek (Co-chair)
Antonín Kučera
František Mráz
Petr Štěpánek (Chair)

Program Committee

Manfred Broy	(Munich)
Bernard Chazelle	(Princeton)
Bruno Codenotti	(Pisa and Iowa City)
Josep Díaz	(Barcelona)
Peter van Emde Boas	(Amsterdam)
Michael Fellows	(Newcastle)
Irene Guessarian	(Paris)
Juhani Karhumaki	(Turku)
Václav Koubek	(Prague, Co-chair)
Dexter Kozen	(Cornell)
Dieter Kratsch	(Metz)
Jan Kratochvíl	(Prague, Chair)
Antonín Kučera	(Brno)
Janos Pach	(Budapest and New York)
Branislav Rovan	(Bratislava)
Georg Schnitger	(Frankfurt)
Paul Spirakis	(Patras)
Ondřej Sýkora	(Loughborough)
Jan Arne Telle	(Bergen)
Paul Vitanyi	(Amsterdam)
Peter Widmayer	(Zurich)

External Referees

Serge Abiteboul
Dimitris Achlioptas
Helmut Alt
Carme Àlvarez
Christoph Ambuhl
Luzi Anderegg
Albert Atserias
Jean-Michel Autebert
Volker Baier
Clemens Ballarin
Jiří Barnat
Eulalia Barriere
Andreas Bauer
Gertrud Bauer
Thomas Bayer
Marie-Pierre Béal
Daniéle Beauquier
Jean Berstel
Nicole Bidoit
Stephen L. Bloom
Nino Boccara
Ahmed Bouajjani
Gérard Boudol
Julian Bradfield
Andreas Brandstädt
Tomáš Brázdil
Herve Bronnimann
Wilfried Buchholz
Andrei Bulatov
Harry Buhrman
Olivier Carton
Didier Caucal
Patrick Cegielski
Ivana Černá
Chandra Chekuri
Christian Choffrut
Jan Chomicki
Anne Condon
Bruno Courcelle
Nadia Creignou
Eugen Czeizler
Artur Czumaj
Pierpaolo Degano
Jorg Derungs
Mariangiola Dezani-Ciancaglini
Volker Diekert
Rod Downey
Olivier Dubois
Pavol Duris
Jan Ernst
Panagiota Fatourou
Francesca Fiorenzi
Fedor Fomin
Lance Fortnow
Rusins Freivalds
Murdoch Gabbay
Paul Gastin
Viliam Geffert
Francoise Gire
Andreas Goerdt
Serge Grigorieff
Martin Grohe
Jozef Gruska
Peter Habermehl
Magnús Halldórsson
Hans-Dietrich Hecker
Jan Heering
Pinar Heggernes
Harald Hempel
Volker Heun
Daniel Hirschkoff
Ron Hirschorn
Hein van der Holst
Klaus Holzapfel
Markus Holzer
Mirko Horňák
Petr Jančar
Natasha Jonoska
Gabriel Juhas
Stasys Jukna
Marcin Jurdzinski
Jan Jürjens
Jarkko Kari
Hartmut Klauck
Jeroen Ketema
Astrid Kiehn
Lefteris Kirousis
Martin Klazar
Ondřej Klíma
Ton Kloks
Leonid Kof
Barbara König
Spyros Kontogiannis
Sven Kosub
Daniel Král'
Rastislav Kralovic
Evangelos Kranakis
Pavel Krčál
Mojmír Křetínský
Antonín Kučera
Gregory Kucherov
Armin Kuehnemann
Michal Kunc
Petr Kurka
Alexander Kurz
Jeff Lagarias
Klaus-Jörn Lange
Kim Larsen
Michel Latteux
Reinhold Letz
Leonid Libkin
Benedikt Löwe
Christos Makris
Maurice Margenstern
Jiří Matoušek
Richard Mayr
Catherine McCartin
Daniel Mölle
Kenichi Morita
Haiko Müller
Anca Muscholl
Rolf Niedermeier
Sotiris Nikoletseas
Tobias Nipkow
Marc Nunkesser
Jan Obdržálek
Vicky Papadopoulou
Daniël Paulusma
Leon Peeters
Radek Pelánek
Jean-Eric Pin

Wojciech Plandowski
Libor Polák
Wolfgang Prenninger
Andrzej Proskurowski
Rajiv Raman
Giovanni Resta
Stefan Richter
Zsuzsanna Roka
Jan Romberg
Peter Rossmanith
Joerg Rothe
Michel de Rougemont
Salvador Roura
Zdeněk Ryjáček
Kai Salomaa
Jacques Sakarovitch
Antonino Salibra
Robert Šámal
Massimo Santini
Bernhard Schaetz
Schmidt Schauss
Nicole Schweikardt
Thomas Schwentick
Alberto Segre
Géraud Senizergues
Olivier Serre
Jiří Sgall
Detlef Sieling
Daniel Singer
Anatol Slissenko
Jan Slovák
Maria Spichkova
Katharina Spies
Jiří Srba
Oldřich Stražovský
Gernot Stenz
Martin Strecker
Jan Strejček
Gabor Szabo
Dimitrios Thilikos
Wolfgang Thomas
Cesare Tinelli
Sophie Tison
Jacobo Torán
Daniele Varacca
Helmut Veith
Mahe Velauthapillai
Victor Vianu
Walter Vogler
Sergei Vorobyov
Imrich Vrt'o
Mirjam Wattenhofer
Birgitta Weber
Martin Wildmoser
Gerhard J. Woeginger
Ronald de Wolf
Christos Zaroliagis
Wieslaw Zielonka

Previous MFCS

MFCS symposia have been organized in Czechia, Poland or Slovakia since 1972. The previous meetings took place in:

1972 Jablonna (Poland)
1973 Štrbské Pleso (Czechoslovakia)
1974 Jadwisin (Poland)
1975 Mariánské Lázně (Czechoslovakia)
1976 Gdańsk (Poland)
1977 Tatranská Lomnica (Czechoslovakia)
1978 Zakopane (Poland)
1979 Olomouc (Czechoslovakia)
1980 Rydzyna (Poland)
1981 Štrbské Pleso (Czechoslovakia)
1984 Praha (Czechoslovakia)
1986 Bratislava (Czechoslovakia)
1988 Karlovy Vary (Czechoslovakia)
1989 Porabka-Kozubnik (Poland)
1990 Banská Bystrica (Czechoslovakia)
1991 Kazimierz Dolny (Poland)
1992 Praha (Czechoslovakia)
1993 Gdańsk (Poland)
1994 Košice (Slovakia)
1995 Praha (Czech Republic)
1996 Kraków (Poland)
1997 Bratislava (Slovakia)
1998 Brno (Czech Republic)
1999 Szklarska Poreba (Poland)
2000 Bratislava (Slovakia)
2001 Mariánské Lázně (Czech Republic)
2002 Warsaw-Otwock (Poland)
2003 Bratislava (Slovakia)

Table of Contents

Invited Lectures

A Case Study of Genome Evolution: From Continuous to Discrete Time Model 1
Jerzy Tiuryn, Ryszard Rudnicki, Damian Wójtowicz

Multicoloring: Problems and Techniques 25
Magnús M. Halldórsson, Guy Kortsarz

Some Recent Progress in Algorithmic Randomness 42
Rod Downey

Ubiquitous Parameterization — Invitation to Fixed-Parameter Algorithms 84
Rolf Niedermeier

PRAM-On-Chip: A Quest for Not-So-Obvious Non-obviousness 104
Uzi Vishkin

Theory and Applied Computing: Observations and Anecdotes 106
Matthew Brand, Sarah Frisken, Neal Lesh, Joe Marks, Daniel Nikovski, Ron Perry, Jonathan Yedidia

Boxed Ambients with Communication Interfaces 119
Eduardo Bonelli, Adriana Compagnoni, Mariangiola Dezani-Ciancaglini, Pablo Garralda

Algebraic Recognizability of Languages 149
Pascal Weil

Geometric Optimization and Unique Sink Orientations of Cubes 176
Emo Welzl

Congestion Games and Coordination Mechanisms 177
Elias Koutsoupias

Graph Algorithms

Equitable Colorings of Bounded Treewidth Graphs 180
Hans L. Bodlaender, Fedor V. Fomin

The Bidimensional Theory of Bounded-Genus Graphs 191
Erik D. Demaine, MohammadTaghi Hajiaghayi, Dimitrios M. Thilikos

Parallel Knock-Out Schemes in Networks 204
Hajo Broersma, Fedor V. Fomin, Gerhard J. Woeginger

Online Algorithms for Disk Graphs 215
Ioannis Caragiannis, Aleksei Fishkin, Christos Kaklamanis, Evi Papaioannou

Approximations

Protein Folding in the HP Model on Grid Lattices with Diagonals 227
Hans-Joachim Böckenhauer, Dirk Bongartz

Optimization, Games, and Quantified Constraint Satisfaction 239
Hubie Chen, Martin Pál

Approximating Boolean Functions by OBDDs 251
Andre Gronemeier

On Approximation Hardness of the Minimum 2SAT-DELETION Problem 263
Miroslav Chlebík, Janka Chlebíková

Graphs and Complexity

Group Coloring and List Group Coloring Are Π_2^P-Complete 274
Daniel Král', Pavel Nejedlý

Complexity Results in Graph Reconstruction 287
Edith Hemaspaandra, Lane A. Hemaspaandra, Stanisław Radziszowski, Rahul Tripathi

Generating Paths and Cuts in Multi-pole (Di)graphs 298
Endre Boros, Khaled Elbassioni, Vladimir Gurvich, Leonid Khachiyan, Kazuhisa Makino

Packing Directed Cycles Efficiently 310
Zeev Nutov, Raphael Yuster

Circuits

The Complexity of Membership Problems for Circuits over Sets of Integers 322
Stephen D. Travers

Some Meet-in-the-Middle Circuit Lower Bounds 334
Kristoffer Arnsfelt Hansen, Peter Bro Miltersen

The Enumerability of P Collapses P to NC 346
Alina Beygelzimer, Mitsunori Ogihara

On NC^1 Boolean Circuit Composition of Non-interactive Perfect Zero-Knowledge 356
Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano

General Complexity

All Superlinear Inverse Schemes Are coNP-Hard 368
Edith Hemaspaandra, Lane A. Hemaspaandra, Harald Hempel

The Complexity of Equivalence and Isomorphism of Systems of Equations over Finite Groups 380
Gustav Nordh

Generation Problems 392
Elmar Böhler, Christian Glaßer, Bernhard Schwarz, Klaus Wagner

One Query Reducibilities Between Partial Information Classes 404
Sebastian Bab, Arfst Nickelsen

Automata

A New Dimension Sensitive Property for Cellular Automata 416
Vincent Bernardi, Bruno Durand, Enrico Formenti, Jarkko Kari

Captive Cellular Automata 427
Guillaume Theyssier

Simulating 3D Cellular Automata with 2D Cellular Automata 439
Victor Poupet

Graph Exploration by a Finite Automaton 451
Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, David Peleg

Parametrized and Kolmogorov Complexity

On Polynomially Time Bounded Symmetry of Information 463
Troy Lee, Andrei Romashchenko

Scaled Dimension and the Kolmogorov Complexity of Turing-Hard Sets 476
John M. Hitchcock, María López-Valdés, Elvira Mayordomo

A Geometric Approach to Parameterized Algorithms for Domination Problems on Planar Graphs 488
Henning Fernau, David Juedes

Polynomial Time Approximation Schemes
and Parameterized Complexity 500
Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, Ge Xia

Semantics

Epistemic Foundation of the Well-Founded Semantics over Bilattices 513
Yann Loyer, Umberto Straccia

Structural Model Checking for Communicating Hierarchical Machines ... 525
Ruggero Lanotte, Andrea Maggiolo-Schettini, Adriano Peron

Compositional Verification: Decidability Issues
Using Graph Substitutions 537
Olivier Ly

Event Structures for Resolvable Conflict 550
Rob van Glabbeek, Gordon Plotkin

Scheduling

Optimal Preemptive Scheduling for General Target Functions..... 562
Leah Epstein, Tamir Tassa

The Price of Anarchy for Polynomial Social Cost..... 574
Martin Gairing, Thomas Lücking, Marios Mavronicolas, Burkhard Monien

Agent-Based Information Handling in Large Networks 586
Robert Elsässer, Ulf Lorenz, Thomas Sauerwald

Approximating Earliest Arrival Flows with Flow-Dependent
Transit Times 599
Nadine Baumann, Ekkehard Köhler

Algebraic Theory of Languages

A Hierarchy of Irreducible Sofic Shifts 611
Marie-Pierre Béal, Francesca Fiorenzi, Dominique Perrin

Membership and Reachability Problems
for Row-Monomial Transformations 623
Alexei Lisitsa, Igor Potapov

On Pseudovarieties of Semiring Homomorphisms 635
Libor Polák

An Algebraic Generalization of ω-Regular Languages 648
Zoltán Ésik, Werner Kuich

Games

A Protocol for Serializing Unique Strategies 660
Marcel Crasmaru, Christian Glaßer, Kenneth W. Regan, Samik Sengupta

A Combinatorial Strongly Subexponential Strategy Improvement Algorithm for Mean Payoff Games 673
Henrik Björklund, Sven Sandberg, Sergei Vorobyov

When Can You Play Positionally?..... 686
Hugo Gimbert, Wiesław Zielonka

Languages

The Dual of Concatenation..... 698
Alexander Okhotin

Computational Aspects of Disjunctive Sequences..... 711
Klaus Ambos-Spies, Edgar Busse

Decidability of Trajectory-Based Equations..... 723
Michael Domaratzki, Kai Salomaa

Geometry

Efficient View Point Selection for Silhouettes of Convex Polyhedra 735
Therese Biedl, Masud Hasan, Alejandro López-Ortiz

Angles and Lengths in Reconfigurations of Polygons and Polyhedra 748
Therese Biedl, Anna Lubiw, Michael J. Spriggs

Improved Bounds and Schemes for the Declustering Problem 760
Benjamin Doerr, Nils Hebbinghaus, Sören Werth

Crossing Number Is Hard for Cubic Graphs 772
Petr Hliněný

Languages and Complexity

A Reducibility for the Dot-Depth Hierarchy 783
Victor L. Selivanov, Klaus W. Wagner

Sublogarithmic Ambiguity 794
Klaus Wich

An Elementary Proof for the Non-parametrizability of the Equation $xyz = zvx$ 807
Elena Petre

A Generalization of Repetition Threshold 818
Lucian Ilie, Pascal Ochem, Jeffrey Shallit

Quantum Computing

An Algorithmic Argument for Nonadaptive Query Complexity Lower Bounds on Advised Quantum Computation 827
Harumichi Nishimura, Tomoyuki Yamakami

Universal Test for Quantum One-Way Permutations 839
Akinori Kawachi, Hirotada Kobayashi, Takeshi Koshiba, Raymond H. Putra

A Common Algebraic Description for Probabilistic and Quantum Computations 851
Martin Beaudry, José M. Fernandez, Markus Holzer

XML

Extraction and Implication of Path Constraints 863
Yves André, Anne-Cécile Caron, Denis Debarbieux, Yves Roos, Sophie Tison

Schema Evolution for XML: A Consistency-Preserving Approach 876
Béatrice Bouchou, Denio Duarte, Mírian Halfeld Ferrari Alves, Dominique Laurent, Martin A. Musicante

Complexity of Decision Problems for Simple Regular Expressions 889
Wim Martens, Frank Neven, Thomas Schwentick

Author Index 901

A Case Study of Genome Evolution: From Continuous to Discrete Time Model*

Jerzy Tiuryn[1], Ryszard Rudnicki[2], and Damian Wójtowicz[1]

[1] Institute of Informatics, Warsaw University
[2] Institute of Mathematics, Polish Academy of Sciences

Abstract. We introduce and analyse a simple model of genome evolution. It is based on two fundamental evolutionary events: gene loss and gene duplication. We are mainly interested in asymptotic distributions of gene families in a genome. This is motovated by previous work which consisted in fitting the available genomic data into, what is called *paralog distributions*. Two approaches are presented in this paper: continuous and discrete time models. A comparison of them is presented too – the asymptotic distribution for the continuous time model can be seen as a limit of the discrete time distributions, when probabilities of gene loss and gene duplication tend to zero. We view this paper as an intermediate step towards mathematically settling the problem of characterizing the shape of paralog distribution in bacterial genomes.

1 Introduction

Fitting data into various kinds of plots is a common practice of modern biology. A typical case is a study of genome organization and evolution, which can be viewed as a branch of a relatively new area of computational biology, called *comparative genomics* (see [9]). We can view a genome not simply as a set of genes, but rather as a dynamic collection of genes which changes in time. Various biochemical processes (e.g. point mutation, recombination, gene conversion, replication, DNA repair, translocation, horizontal transfer) constantly act on genomes and drive them to evolve dynamically. A problem which has been addressed in late 90's in this framework is an estimate of the distribution of paralogs in a genome. Two genes in a genome are said to be *paralogs* if they have evolved through duplication from a single ancestral gene. We do not discuss here the important issue of deciding which genes are paralogs. We assume that all genes have been clustered into groups of pairwise paralogous genes. The question which was asked in 1998 by P. Slonimski ([12,13]) and independently by M.A. Huynen and E. van Nimwegen [3] was about the distribution of the numbers of i-element clusters of paralogous genes (for consequtive i's) in several microbial genomes which have been sequenced till then. The distribution was estimated by fitting

* This research was partially supported by the State Committee for Scientific Research (Poland) Grants No. 2 P03A 031 25, and 7 T11F 016 21 and by the EC programme *Centres of Excellence for States in phase of pre-accession,* No. ICA1-CT-2000-70024.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 1–24, 2004.
© Springer-Verlag Berlin Heidelberg 2004

the available data. Since the method of deciding paralogy is only approximate and the size of the genomes was not large, as a consequence both authors came up with different answers: [12] claims that the distribution is logarithmic, while [3] claims that it follows the so called power law distribution. In 2001 Jordan et al. [4] have analysed 21 completely sequenced bacterial genomes and claimed that the logarithmic approximation fits the distribution slightly better than the power law approximation, although the difference between the two fits is not significant.

It should be obvious from the above description that it will be impossible to decide what actually is the observed distribution if we rely merely on the biological data. A decisive answer should come by adopting a certain mathematical model of genome evolution together with a rigorous analysis of the asymtotic distribution within this model. This is the main motivation for the present paper to build and analyse a simple model of genome evolution. The model we study is very simple indeed. It addresses only two evolutionary events: gene loss and gene duplication. Even though it is too simple to settle the problem of distribution of paralog families in genomes it can be used to study various subtleties of the model. We treat this paper as an intermediate step towards analysis of a more complicated model, which we postpone for future publication.

There is a short history of mathematical modeling of genome evolution. In 2000 Yanai et al. [16] designed a simple model of genome evolution based on random gene duplication and point mutations. The paper did not analyse the model. The main result consisted in showing that it is possible for each of the 20 microbal genomes to tune the parameters of the model so that the obtained distribution matches closely the paralog distribution of the genome. Recently Koonin's group has published in a series of papers [8,5,6,7] a simple model (called BDIM) of genome evolution which resembles our continuous time model. However, there are two important differences between the two models. In BDIM model in addition to gene loss and duplication there is an external source of new genes, called *invention.* This source is used to stabilize the asymptotic behavior of the model, i.e. to make sure that the supply of genes does not vanish at some point of evolution. On the contrary, we are interested in asymptotic distributions for the two extreme situtations: genome collapse and genome explosion. The reason for this is that if we assume that the two events: gene loss and gene duplication are independent of each other, it follows that we have to assume that their probabilities should not be equal. This leads the model to one of the two extreme situations. The second difference is more important. BDIM model sets an upper bound on the maximal size of gene family in the model. Technically speaking, this assumption implies that the system of differential equations is finite and the theory of finite dimentional matrices is applicable here. In the model which we investigate in this paper we do not impose any bound on the maximal size of a gene family and we end up with an infinite system of differential equations, for which existence of stationary solutions needs a special justification (see Theorem 1).

Continuous time model represents an ideal situation: in one unit of time two or more events can happen to a single gene, even though the probability of this is very low. In discrete time model we assume that in one unit of time every gene of the genome is subject to exactly one of the following events: removal, duplication, idle; each with a fixed probability. So, discrete time model is an approximation (and simplification) of the continuous model. Discrete model is much more suitable for computer simulations. Also, as we will see, the asymptotic distributions for both models are always different. The analysis of the discrete model is apparently more complicated, presumably due to lack of strong analytical tools. Moreover, as it follows from one of our results (see Theorem 7), the distribution for a continuous model can be seen as a limit of the discrete time distribution, when the probabilities of gene loss and duplication tend to zero. Another noteworthy property of the discrete model is a very nice isomorphism (see Theorem 8) between the situation of genome collapse (i.e. when Prob(gene duplication) < Prob(gene loss)) and a genome explosion (when Prob(gene duplication) > Prob(gene loss)). This allows us to reduce the latter situtation to the former. It appears that direct analysis of genome explosion is very difficult since the distribution looks more like a uniform distribution on an infinite set. The discrete model presented in this paper is in the same spirit as the model of DNA evolution presented in [14,15].

The paper is organized as follows. Section 2 contains a description of results for the continuous time model, together with asymptotic distributions for genome collapse (Theorem 2) and explosion (Theorem 3). Section 3 is devoted to discrete time. In particular we give a charcterization of a generating function for the asymptotic distribution for collapse (Theorem 6). All longer proofs are moved to the Appendix.

2 Continuous Time Model

Before we start a description of the genome evolution, let us introduce all entities used in our model: genes, gene families, class of gene families and genomes. *Genes* are atomic units, i.e. we do not assume any internal structure of these objects. A *genome* is a finite set of all genes. A *gene family* in a genome is a set of genes of that genome which are paralogs. We group families according to their size. *Classes* are sets of gene families which have the same number of elements, i.e. S is a class i if every family in S has i elements[1]. One gene duplication in a family belonging to class i results in relocation of this family from class i to class $i+1$. Conversely, one gene removal relocates the corresponding family to class $i-1$ or eliminates this family if $i = 1$. In this section, we consider time to be continous.

We assume that the probability of a gene duplication to happen during time interval of length Δt is $d \cdot \Delta t + o(\Delta t)$. Similarly, probability of gene removal in time interval Δt is $r \cdot \Delta t + o(\Delta t)$. It is assumed that $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$. Moreover, we assume that all elementary events (gene duplication and removal) are independent of each other.

[1] Obviously, a class may include families that are completely unreleted biologically.

Let $C_i(t)$ be the number of i-element families in our model at the time t. It follows from the description of our model that we have the following equations which describe an increment $\Delta C_i(t) = C_i(t + \Delta t) - C_i(t)$ during time interval Δt:

$$\Delta C_1(t) = -(d + r)C_1(t)\Delta t + 2rC_2(t)\Delta t + o(\Delta t)$$

and

$$\Delta C_i(t) = d(i - 1)C_{i-1}(t)\Delta t - (d + r)iC_i(t)\Delta t + r(i + 1)C_{i+1}(t)\Delta t + o(\Delta t),$$

for $i \geq 2$.

Hence, dividing both sides of the above equations by Δt and passing with Δt to 0, we obtain the following infinite system of differential equations:

$$C_i'(t) = d(i - 1)C_{i-1}(t) - (d + r)iC_i(t) + r(i + 1)C_{i+1}(t), \tag{1}$$

where $i \geq 1$. The above equation for $i = 1$ reduces to $C_1'(t) = -(d + r)C_1(t) + 2rC_2(t)$, independently of the value of $C_0(t)$. We assume that the latter is just $C_0(t) = 0$. Let us also observe that $C_i(t) = 0$, for $i \geq 1$ and $t \in \mathbb{R}$ is a (trivial) solution of (1).

Theorem 1. *If $r > 0$ and $d > 0$, then for each non-zero and non-negative absolutely summable sequence $(C_i(0))_{i \geq 1}$ equation (1) has a unique solution such that $C_i(t) > 0$ for all $t > 0$ and all positive integers i.*

The remainder of this section is devoted to the asymptotic behavior of a solution of (1), as $t \to \infty$. It turns out that the behavior of the system is quite different, depending on whether $r > d$, or $r < d$. In the former case all genes are eventually removed, while in the latter case we have an exponential explosion of the number of genes in the genome. We consider each case separately.

2.1 Collapse of the Genome: $r > d$

The next result characterizes an asymptotic behavior of solutions of (1) when $r > d$.

Theorem 2. *Let $(C_i(t))_{i \geq 1}$ be non-negative and non-zero solution of (1). If $r > d > 0$, then there exists a constant $c > 0$ such that for all $i \geq 1$,*

$$\lim_{t \to \infty} e^{(r-d)t} \cdot C_i(t) = c \cdot (\frac{d}{r})^i.$$

Hence for sufficiently large t the number $C_i(t)$ of i element gene families has the following asymptotics

$$C_i(t) \sim c \cdot e^{-(r-d)t} \cdot (\frac{d}{r})^i,$$

for all $i \geq 1$.

Corollary 1. *If $r > d$, then for sufficiently large t, the distribution of gene families in the genome at time t is close to geometric, i.e. if X is a random variable which measures the size of a gene family of paralogous genes, then*

$$Prob(X = i) = c \cdot \theta^i,$$

where $\theta = d/r$ and c is a normalizing constant.

2.2 Exponential Explosion of the Genome: $d > r$

We have the following result

Theorem 3. *Let $(C_i(t))_{i\geq 1}$ be non-negative and non-zero solution of (1). If $d > r > 0$, then there exist constants c_1 and c_2 such that for $i \geq 1$ and all $t \in \mathbb{R}$*

$$lim_{t\to\infty} e^{(d-r)t} \cdot C_i(t) = c_1.$$

In particular

$$lim_{t\to\infty} \frac{C_i(t)}{C_1(t)} = 1.$$

Moreover

$$e^{(r-d)t} \sum_{i\geq 1} i \cdot C_i(t) = c_2.$$

The above result shows a peculiar behavior of the system during genome explosion: each $C_i(t)$ vanishes exponentially fast (with the same speed, independent of i), while the total mass shows an exponential increase.

3 Discrete Time Model

In the previous section we introduce formal definition of all entities required in our discrete model. Now, we start with a Markov chain which is going to model the genome evolution with discrete time. States are infinite sequences $(s_i)_{i\geq 1}$ of non-negative integers. A state $(s_i)_{i\geq 1}$ represents a genome which for every $i \geq 1$ has s_i i-element gene families. As in the continuous time approach we introduce two parameters $p_R > 0$ and $p_D > 0$ which express the influence of removal and duplication, respectively. Let us assume that $p_R + p_D \leq 1$. Each gene of the genome is subject in one step of the Markov chain to:

- removal, with probability p_R;
- duplication, with probability p_D.

The gene remains unchanged with probability $p_U = 1 - p_R - p_D$.

For $i, j \geq 1$ let $q_{i,j}$ be the probability that a gene family of size i gets size j, after every gene of the family has been subject to the above events. It can be shown that

$$q_{i,j} = \sum_{k=0}^{\lfloor j/2 \rfloor} \binom{i}{j-2k, k} p_U^{j-2k} p_D^k p_R^{i-j+k}. \tag{2}$$

It follows that if $(s_i)_{i\geq 1}$ is a current state of the Markov chain, then the expected number of j-element gene families in the next state is

$$E_j = \sum_{i=1}^{\infty} q_{i,j} \cdot s_i.$$

Let $\widehat{Q} = (q_{i,j})_{i,j\geq 1}$. Hence, for $n \geq 0$

$$(C_1^{(n)}, \ldots) = (1, 0, \ldots) \cdot \widehat{Q}^n$$

is an infinite sequence, where $C_i^{(n)}$ is the expected number of i-element families of the Markov chain after n steps, assuming that the chain has started with the initial state consisting of just one gene. Again, we are interested in the observed distribution of gene families as we run the Markov chain a sufficient number of steps, and assuming that we started initially with a sufficient number of one element gene families.[2]

It is useful to extend the concept of a state $(s_i)_{i\geq 1}$ by introducing the quantity s_0 which represents the number of genes which have been removed. Thus an *extended state* is a sequence $(s_i)_{i\geq 0}$, where $(s_i)_{i\geq 1}$ is the ordinary state and s_0 is the number of genes which have been removed. Then equation (2) makes sense for $i \geq 1$ and $j \geq 0$, where $q_{i,0}$ is the probability that a family of size i disappears completely from the genome. We also set $q_{0,0} = 1$ and $q_{0,j} = 0$, for $j > 0$, which again satisfies equation (2).

It follows that the extended matrix $Q = (q_{i,j})_{i,j\geq 0}$ is a stochastic matrix, i.e. $\sum_{j\geq 0} q_{i,j} = 1$, for all $i \geq 0$. Moreover, if we define for $n \geq 0$,

$$(C_0^{(n)}, C_1^{(n)}, \ldots) = (0, 1, 0, 0, \ldots)Q^n,$$

then, as before, for $i \geq 1$, $C_i^{(n)}$ is the expected number of i-element gene families after n steps, and $C_0^{(n)}$ is the expected number of genes which have been removed after n steps. The above holds under proviso that the initial state of the Markov chain is $(0,1,0,0,\ldots)$. It also follows that $(C_i^{(n)})_{i\geq 0}$ is a probability distribution, i.e. $\sum_{i\geq 0} C_i^{(n)} = 1$.

It is easy to show the following conservation property. If $(C_1^{(n)}, \ldots) = (1, 0, \ldots)\widehat{Q}^n$ and $(D_0^{(n)}, D_1^{(n)}, \ldots) = (0, 1, 0, 0, \ldots)Q^n$, then $D_i^{(n)} = C_i^{(n)}$, for all $i \geq 1$.

It follows from the above remarks that for every $n \geq 0$, Q^n is a stochastic matrix, hence each of its rows is a probability distribution. A *generating function* for probability distribution $(p_i)_{i\geq 0}$ is the function $f(x) = \sum_{i\geq 0} p_i x^i$. This function is defined at least on the closed interval $-1 \leq x \leq 1$.

Let

$$\varphi(x) = p_R + p_U x + p_D x^2. \tag{3}$$

[2] If $p_R > p_D$, then we have to start simulations with suffcient supply of genes to ensure a sufficient number of steps before the genome vanishes.

The next theorem is a standard result in the theory of branching processes (see [2], chapter XII).

Theorem 4. *For all $n, i \geq 0$, the i-th row in Q^n has generating function $[\varphi^n(x)]^i$, where $\varphi^n(x)$ is n-fold composition of φ with itself.*

It follows from the above result that

$$\varphi^n(x) = \sum_{i=0}^{2^n} C_i^{(n)} x^i, \tag{4}$$

and $C_i^{(n)} = 0$, for $i > 2^n$. In particular we have $C_0^{(n)} = \varphi^n(0)$.

Instead of working directly with distribution (p_R, p_U, p_D) it will be sometimes more convenient to represent this distribution as $(p, 1-p-ap, ap)$, where $0 < p < 1$ is the probability of removal and $0 < a < (1-p)/p$ is a constant which expresses the ratio $a = \frac{p_D}{p_R}$. We will use both notations interchangeably as it never should cause a confusion.

Let us briefly look at the asymptotic behavior of $C_i^{(n)}$, when n tends to infinity. We have the following easy observation.

$$\lim_{n\to\infty} C_0^{(n)} = \begin{cases} 1, & \text{if } p_R > p_D, \\ a^{-1}, & \text{if } p_D > p_R. \end{cases} \tag{5}$$

Indeed, since $C_0^{(n)} = \varphi^n(0)$, it follows that the limit must be the smallest fixed point of $\varphi(x)$. Since $\varphi(x) - x = ap(x-1)(x-a^{-1})$, (5) follows.

Hence we obtain

$$\lim_{n\to\infty} \sum_{i\geq 1} C_i^{(n)} = \begin{cases} 0, & \text{if } p_R > p_D, \\ \frac{1-a}{a}, & \text{if } p_D > p_R. \end{cases} \tag{6}$$

It follows from the above that when $p_R > p_D$, then for every $i \geq 1$ we have $\lim_{n\to\infty} C_i^{(n)} = 0$.

Let us assume now that $p_D > p_R$. We show that

$$\lim_{n\to\infty} C_1^{(n)} = 0 \tag{7}$$

Indeed, it follows from (4) that for all $n \geq 0$ we have

$$C_1^{(n+1)} = p_U + C_1^{(n)} + 2p_D C_0^{(n)} C_1^{(n)},$$

and $C_1^{(0)} = 0$. Thus, if we let

$$\alpha_n = p_U + 2p_D C_0^{(n)},$$

then

$$C_1^{(n)} = \alpha_0 \alpha_1 \cdots \alpha_{n-1}.$$

Since $(C_1^{(n)})_{n\geq 1}$ is monotone increasing, it follows from (5) that

$$\alpha_i < p_U + 2p_D a^{-1} = p_U + 2p_R < 1.$$

This proves (7).

It follows from (7) and from Theorem 5 (see below) that for $i \geq 1$,

$$\lim_{n\to\infty} C_i^{(n)} = 0.$$

Thus we have argued that the sequence of $(C_1^{(n)}, C_2^{(n)}, \ldots)$ converges componentwise to $(0,0,\ldots)$, when $n \to \infty$. In order to capture the asymptotic behavior of the distribution of $(C_1^{(n)}, C_2^{(n)}, \ldots)$ we introduce the ratio

$$\xi_n(i) = \frac{C_i^{(n)}}{C_1^{(n)}}.$$

We are interested in

$$\xi(i) = \lim_{n\to\infty} \xi_n(i).$$

Observe that the information about function ξ completely determines the asymptotic distribution, provided $\sum_{i\geq 1} \xi(i) < \infty$. In particular this distribution is geometric iff $\xi(i) = \theta^{i-1}$, for some $0 < \theta < 1$.

The next result shows that ξ is well defined.

Theorem 5. *If $p_R \neq p_D$, then for every $i \geq 1$,*

$$0 < \xi(i) < \infty.$$

We call ξ the *asymptotic ratio.* Clearly it depends merely on the initial probabilities p_R and p_D.

The main technical result used in the proof of Theorem 5 is the following property. Let

$$\beta_n = p_D \frac{C_1^{(n)}}{\alpha_n}.$$

Then it can be proved that for every $i \geq 2$ and for all $n \geq 0$ we have

$$\xi_n(i) = \sum_{0<j<i} \sum_{m<n} \xi_m(j) \cdot \xi_m(i-j) \cdot \beta_m. \tag{8}$$

The above formula can be used to design an algorithm for computing the functions ξ_n. In contrast to Corollary 1 and to the impression one can get from Figure 1(b), it follows from (8) that the asymptotic distribution described by ξ is never geometric.

Proposition 1. *For no p_R and p_D with $p_R \neq p_D$ the asymptotic distribution described by ξ is geometric.*

Proof. Using equation (8), it is easy to show that $\xi(2)^2 > \xi(3)$. Hence the distribution cannot be geometric. □

Next, we will study the shape of the asymptotic distribution for case $p_R > p_D$ and $p_D > p_R$, separately. It will turn out that the nature of the process is completely different in each of the cases, yet, as we will see, there is a nice isomorphism between distributions which arise in each of the cases.

3.1 Collapse of the Genome: $p_R > p_D$

Let us look at the shape of the graph of function ξ depicted in Figure 1. We notice that for small values of p_R the graph fits very well geometric distribution (see Fig. 1(b)). However, by Proposition 1, for no value $p_R > 0$ it coincides with the geometric distribution. The difference can be better seen for larger values of p_R (see Fig. 1(a)).

Recall that it follows from Theorem 4 that $\varphi^n(x)$ is the generating function for the row number 1 in Q^n. Hence it is a generating function for the probability distribution $(C_i^{(n)})_{i\geq 0}$. It follows that the generating function for the distribution which is obtained from $(C_i^{(n)})_{i\geq 0}$ by restricting i to range over positive integers is

$$f_n(x) = \frac{\varphi^n(x) - \varphi^n(0)}{1 - \varphi^n(0)}.$$

Hence the generating function for the asymptotic distribution is the limit of $\{f_n\}_{n\geq 0}$. We have the following result.

Theorem 6. *Let $p_R > p_D$. The limit*

$$f(x) = \lim_{n\to\infty} f_n(x)$$

exists for all $x \in (-\frac{1-ap}{p}a^{-1}, a^{-1})$. The above convergence is almost uniform in complex plane C on the open disk $|x| < a^{-1}$, i.e. it is uniform on every closed subdisk contained in $\{x \in C : |x| < a^{-1}\}$. Hence f is an analytic function with the radius of convergence a^{-1}.

Moreover f satisfies the following equation

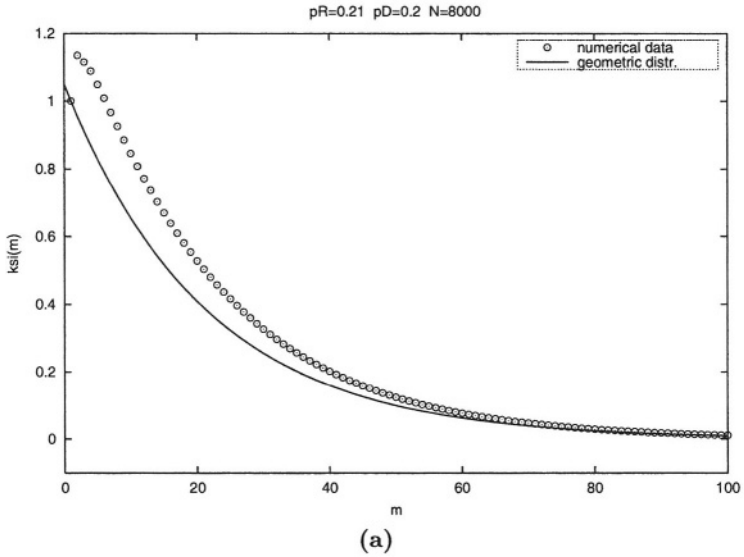
$$f(\varphi(x)) = \mu \cdot f(x) + 1 - \mu, \tag{9}$$

where $\mu = \varphi'(1) = p_U + 2p_D$.

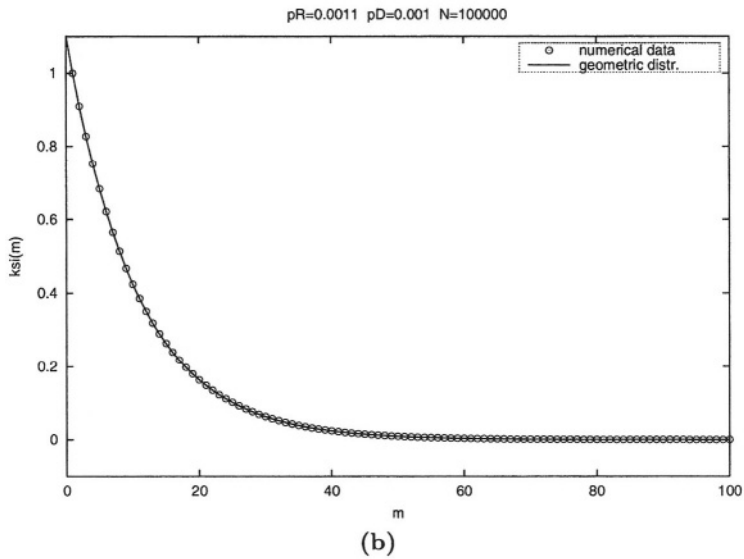
It follows from the theory of analytic functions[3] that f is a unique analytic function which satisfies (9) and the two constraints: $f(0) = 0$ and $f(1) = 1$). In this sense the above theorem gives a complete characterization of the generating function for the distribution of interest.

Relationship between function ξ and the generating function f is given in the next result.

[3] Two analytic functions which coincide on set containing a point of convergence are equal.



(a)



(b)

Fig. 1. Plots of function ξ for the case $p_R > p_D$. In (a) probability p_R is relatively large. The continuous line in this plot shows the best fit of the geometric distribution to the observed data. In (b) p_R is very small. The observed data fits very well in this case a geometric distribution.

Proposition 2. *Let $p_R > p_D > 0$. Then $f'(0) \neq 0$ and*

$$\sum_{i \geq 1} \xi(i) \cdot x^i = \frac{f(x)}{f'(0)}$$

holds for all $|x| < a^{-1}$. In particular we have

$$\sum_{i \geq 1} \xi(i) = \frac{1}{f'(0)}.$$

We conclude this subsection with a result which explains the observed shape of ξ during computer simulations for small values of p_R. In order to stress dependence on $p = p_R$ of functions under consideration we make this dependence explicit. Thus instead of writing $\varphi(x)$ we shall write $\varphi(p, x)$. The same applies to f and f_n. The main result of this section is the following theorem.

Theorem 7. *Let $a < 1$ be fixed. We have*

$$\lim_{p \to 0^+} f(p, z) = \frac{(1-a)z}{1-az},$$

for all z from open disk $|z| < a^{-1}$ in complex numbers.

Hence, by Proposition 2, for all $i \geq 1$, we have

$$\lim_{p \to 0^+} \xi(p, i) = a^{i-1}.$$

3.2 Exponential Explosion of the Genome: $p_D > p_R$

In this subsection we discuss the case $p_D > p_R$ and we shall explain the observed shape of ξ. Let us first look at computer simulations depicted in Figure 2. The striking feature is a completely different shape of the graph of ξ with respect to the previous case $p_R > p_D$. It looks now more like a uniform distribution for small values of p_D, but the domain is infinite! Corollary 2 explains exactly this behaviour.

Fortunately we do not have to repeat all the work done in the previous subsection since there a nice isomorphism which enables us to reduce the present situation to the former case. This isomorphism is given in the next result.

Theorem 8. *Let $p, q > 0$ be reals such that $p + q < 1$. Let $Q_{p,q}$ be the extended stochastic matrix for the initial probabilities $p_R = p$, $p_D = q$. Then*

$$Q_{q,p} = \Delta_a \cdot Q_{p,q} \cdot \Delta_a^{-1},$$

where $a = q/p$ and Δ_a is a diagonal matrix with $\delta_{i,i} = a^i$, for $i \geq 0$.

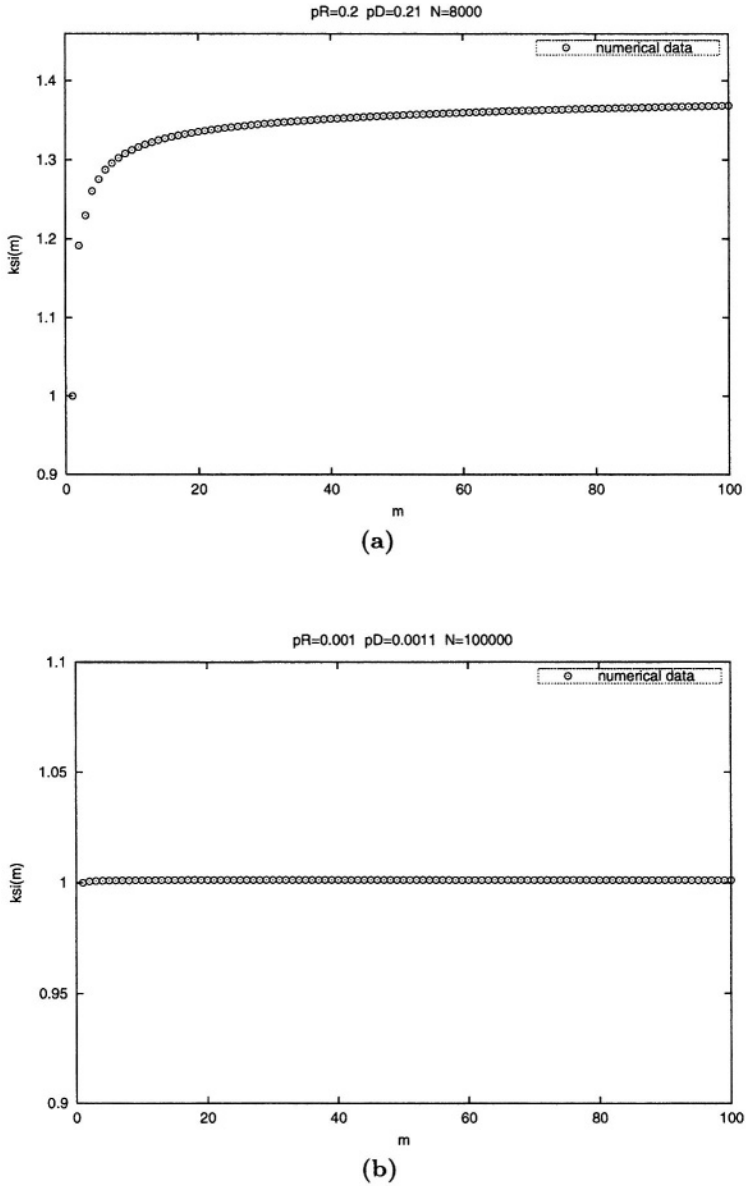


Fig. 2. Plots of function ξ for the case $p_D > p_R$. In (a) probability p_D is relatively large. In (b) p_D is very small.

Proof. Since the proof is very short we give it here. Let

$$\varphi_{p,q}(x) = p + (1 - p - q)x + qx^2$$

and let $a = q/p$. It is easy to check that we have

$$\varphi_{q,p}(x) = a \cdot \varphi_{p,q}(a^{-1} \cdot x).$$

By Theorem 4 the i-th row in $Q_{q,p}$ has the generating function

$$[\varphi_{q,p}(x)]^i = a^i \cdot [\varphi_{p,q}(a^{-1} \cdot x)]^i.$$

Hence

$$Q_{q,p} = \Delta_a \cdot Q_{p,q} \cdot \Delta_a^{-1},$$

as required. □

Let $\xi_{p,q}$ be the asymptotic ratio for the initial probabilities $p_R = p, p_D = q$. It follows immediately from Theorem 8 that for all $i \geq 1$,

$$\xi_{q,p}(i) = a^{i-1}\xi_{p,q}(i), \tag{10}$$

where $a = q/p$. Hence, if we apply Theorem 7 to (10), we easily obtain the following result:

Corollary 2. *Let $a > 1$ be fixed and let for all $i \geq 1$, $\xi(p,i)$ be the asymptotic ratio for the initial probabilities $p_R = p$ and $p_D = a \cdot p$. Then*

$$\lim_{p \to 0+} \xi(p,i) = 1.$$

4 Conclusions

Here we presented a complete mathematical description of the frequency distribution of paralog families in genomes for a simple model of evolution. Our model includes two types of events: gene duplication and removal. Moreover, we analysed and compared two approaches to the time in the model: continous and discrete.

This work is our initial step toward modeling of genome evolution using most important evolutionary processes/events. Actually, we should extend our model by some concept of e.g. gene mutation. Nevertheless, even this simple and crude model seems to reveal several interesting aspects of genome evolution.

Acknowledgements. JT thanks Piotr Slonimski for many discussions on various Markov chain models for genome evolution and Stanisław Kwapień and Krzysztof Oleszkiewicz for fruitful discussions on mathematical properties of the model.

References

1. Devaney, R.L., "An Introduction to Chaotic Dynamical Systems.", The Benjamin/Cummings Publishing CO., INC., 1985.
2. Feller W., "An introduction to probability theory and its applications.", Volume 1, John Wiley and Sons, Inc. New York , London 1961,
3. Huynen, M.A., van Nimwegen, E. "The Frequency Distribution of Gene Family Size in Complete Genomes." *Molecular Biology Evolution* 15(5), pp. 583–589, 1998.
4. Jordan, K., Makarova, K.S., Spouge, J.L., Wolf, Y.I., Koonin, E.V. "Lineage-Specific Gene Expansions in Bacterial and Archeal Genomes." *Genome Research* 11, pp. 555–565, 2001.
5. Karev, G.P., Wolf, Y.I., Rzhetsky, A.Y., Berezovskaya, F.S., Koonin, E.V., "Birth and death of protein domains: A simple model of evolution explains power law behavior.", *BMC Evolutionary Biology* 2:18, 2002.
6. Karev, G.P., Wolf, Y.I., Koonin, E.V.,"Simple stochastic birth and death models of genmome evolution: was there enough time for us to evolve?", *Bioinformatics* 19:15, pp. 1889–1900, 2003.
7. Karev, G.P., Wolf, Y.I., Rzhetsky, A.Y., Berezovskaya, F.S., Koonin, E.V., "Mathematical Modeling of the Evolution of Domain Composition of Proteomes: A Birth-and-Death Process with Innovation.", *Frontiers in Computational Genomics,* Caister Academic Press, 2003.
8. Koonin, E.V., Wolf, Y.I., Karev, G.P., "The structure of the protein universe and genome evolution.", *Nature* 420, 2002.
9. Koonin, E., Galperin, M. "Sequence - Evolution - Function: Computational Approaches in Comparative Genomics.", Kluwer Academic Publishers, 2003.
10. Norris, J., "Markov Chains", Cambridge Series on Statistical and Probabilistic Mathematics, Cambridge University Press, Cambridge, 1997.
11. Rudnicki, R., "On asymptotic stability and sweeping for Markov operators", *Bull. Pol. Ac.: Math.* 43, pp. 245–262, 1995.
12. Slonimski, P.P., Mosse, M.O., Golik, P., Henaût, A., Diaz, Y., Risler, J.L., Comet, J.P., Aude, J.C., Wozniak, A., Glemet, E., Codani, J.J. "The first laws of genomics." *Microbial and Comparative Genomics* 3:46, 1998.
13. Slonimski, P.P., "Comparision of complete genomes: Organization and evolution." *Proceedings of the Third Annual Conference on Computational Molecular Biology,* RECOMB'99 Stanislaw Ulam Memorial Lecture, pp. 310, ACM Press, 1999.
14. Tiuryn, J., Radomski, J.P., Slonimski, P.P. "Striking properties of duplicating DNA molecules. A Markov chain model demonstrates the convergence of amplified molecules to regular series of multiplies of two." *Comptes rendus de l'Académie des sciences – Série. III – Sciences de la vie* 322, pp. 455–459, 1999.
15. Tiuryn, J., Radomski, J.P., Slonimski, P.P. "A formal model of genomic DNA multiplication and amplification." *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families* (edited by Sankoff, D., Nadeau, J.H.), pp. 503–513, Kluwer Academic Publishers, 2000.
16. Yanai, I., Camacho, C.J., DeLisi, C. "Predictions of Gene Family Distributions in Microbial Genomes: Evolution by Gene Duplication and Modification." *Physical Review Letters* 85(12), pp. 2641–2644, 2000.

A Appendix: Proofs of Some Statements

A.1 Proof of Theorem 1

We change variables in equation (1) to obtain a system which will be easier in analysis. Let

$$x_i(t) = e^{(r-d)t} i C_i(t). \tag{11}$$

Then, we obtain

$$\begin{aligned} x_i'(t) &= (r-d)x_i(t) + e^{(r-d)t} i C_i'(t) \\ &= (r-d)x_i(t) + e^{(r-d)t} i[d(i-1)C_{i-1}(t) - (d+r)iC_i(t) + r(i+1)C_{i+1}(t)] \end{aligned}$$

and therefore

$$x_i'(t) = idx_{i-1}(t) - [(i+1)d + (i-1)r]x_i(t) + irx_{i+1}(t), \quad i = 1, 2, \ldots, \tag{12}$$

where we formally assume that $x_0(t) = 0$ for all $t \geq 0$.

Let l^1 denote the space of absolutely summable sequences. We check that system (12) generates a Markov semigroup on l^1. Recall that a linear mapping $P : l^1 \to l^1$ is called a *Markov* or *stochastic operator* if $P(D) \subset D$, where

$$D = \{x \in l^1 : x_n \geq 0 \text{ for all } n \geq 1 \text{ and } \sum_{n=1}^{\infty} x_n = 1\}.$$

A family $\{P(t)\}_{t \geq 0}$ of Markov operators which satisfies conditions:

(a) $P(0) = \mathrm{Id}$,
(b) $P(t+s) = P(t)P(s)$ for $s, t \geq 0$,
(c) for each $x \in l^1$ the function $t \mapsto P(t)x$ is continuous with respect to the l^1 norm

is called a *Markov* or *stochastic semigroup.*

We recall that an abstract equation $x'(t) = Ax(t)$, $x(0) = x_0$ generates a Markov semigroup $\{P(t)\}_{t \geq 0}$ if $P(t)x_0 = x(t)$.

Let $a_{i,i} = -(i+1)d - (i-1)r$, $a_{i+1,i} = (i+1)d$ for $i \geq 1$, $a_{i-1,1} = (i-1)r$ for $i \geq 2$, and $a_{i,j} = 0$ in other cases. System (12) can be written in the following way.

$$x'(t) = Ax(t), \quad x(0) = x_0, \tag{13}$$

where $A = (a_{i,j})_{i,j \geq 1}$. Let A be the $A^* = (A^*_{i,j})_{i,j \geq 1}$ where $a^*_{i,j} = a_{j,i}$ for $i, j \geq 1$. The matrix $A = (a_{i,j})_{i,j \geq 1}$ is a *Kolmogorov matrix,* i.e. it satisfies conditions: $a_{i,j} \geq 0$ for $i \neq j$ and $\sum_{k=1}^{\infty} a_{k,j} = 0$ for all $j \geq 1$. We need the following result

Theorem 9. *Let A be a Kolmogorov matrix. Then system (13) generates a Markov semigroup on l^1 if and only if for some positive θ there is no non-zero solution of the equation $A^*x = \theta x$, where $x \in l^\infty$.*

Recall that l^∞ is the space of bounded sequences. The proof of Theorem 9 can be found in [10]. Consider our case. If $A^*x = \theta x$, then

$$(i-1)rx_{i-1} - [(i+1)d + (i-1)r]x_i + (i+1)dx_{i+1} = \theta x_i, \quad i = 1, 2, \ldots, \tag{14}$$

where $x_0 = 0$. Then

$$x_{i+1} = \left(1 + \frac{(i-1)r + \theta}{(i+1)d}\right) x_i - \frac{(i-1)r}{(i+1)d} x_{i-1}. \tag{15}$$

We can assume that $x_1 > 0$. Using simple induction arguments one can check that the sequence (x_i) is increasing and

$$x_{i+1} = \left(1 + \frac{\theta}{(i+1)d}\right) x_i. \tag{16}$$

Thus

$$x_n > x_1 \prod_{i=2}^{n} \left(1 + \frac{\theta}{id}\right) \tag{17}$$

and therefore

$$\lim_{n\to\infty} x_n \geq x_1 \prod_{i=2}^{\infty} \left(1 + \frac{\theta}{id}\right) = \infty. \tag{18}$$

Consequently, $x \notin l^\infty$. Thus we have proved the following result:

Corollary 3. *For each $x \in l^1$ and $t \geq 0$ we denote by $P(t)x$ the solution $x(t)$ of equation (13) with the initial condition $x(0) = x$. Then $\{P(t)\}_{t\geq 0}$ is a Markov semigroup on l^1.*

Let $x \in l^1$ be a non-zero and non-negative sequence. Since $a_{i,i+1} > 0$ and $a_{i+1,i} > 0$ for all $i \geq 1$ we have $(P(t)x)_i > 0$ for all $t > 0$, $i = 1, 2, \ldots$. Now, we can return to equation (1). Let (C_i) be a non-zero and non-negative sequence such that $\sum_{i=1}^{\infty} iC_i < \infty$ Let $x = (x_i)$ be a sequence defined by $x_i = iC_i$. Then $x \in l^1$. From Corollary 3 it follows that equation 13 has a unique solution for every $x_0 \in l^1$ which is given by the formula $x(t) = P(t)x_0$. The unique solution of equation (1) such that $C_i(0) = C_i$ is given by the formula

$$C_i(t) = e^{(d-r)t} i^{-1} (P(t)x)_i, \tag{19}$$

where $x_i = iC_i$. This completes the proof of Theorem 1.

A.2 Proof of Theorem 2

Since the semigroup $\{P(t)\}_{t\geq 0}$ (see Appendix A.1) is defined on the space l^1, it is an *integral* or *kernel semigroup*. Moreover, $\{P(t)\}_{t\geq 0}$ is an irreducible semigroup because $(P(t)x)_i > 0$ for all $t > 0$ if $x \in l^1$ is a non-zero and non-negative sequence. If $\{P(t)\}_{t\geq 0}$ is an integral and irreducible Markov semigroup and there exists $x^* \in D$ such that $P(t)x^* = x^*$ for all $t > 0$, then $P(t)x \to x^*$ for every

$x \in D$ as $t \to \infty$ (see e.g. [11]). If $r > d > 0$, then system (12) has a stationary solution $x_i^* = ci(d/r)^i$ for $i = 1, 2, \ldots$. We can choose the constant c such that $x^* \in D$. Then $P(t)x \to x^*$ for every $x \in D$ as $t \to \infty$. Therefore

$$\lim_{t\to\infty} e^{(r-d)t} \cdot C_i(t) = \frac{x_i^*}{i} = c \cdot (\frac{d}{r})^i.$$

This completes the proof of Theorem 2.

A.3 Proof of Theorem 3

First observe that since $\{P(t)\}_{t\geq 0}$ (see Appendix A.1) is a Markov semigroup we have $\sum_{i=1}^{\infty} x_i(t) = const$ for any solution of system (12). From equation (11) it follows that

$$e^{(r-d)t} \sum_{i=1}^{\infty} iC_i(t) = const.$$

In order to prove the main part of Theorem 3 we substitute $z_i(t) = e^{(d-r)t} \left(\frac{r}{d}\right)^i iC_i(t)$. Then

$$z_i' = irz_{i-1} - [(i+1)r + (i-1)d]z_i + idz_{i+1}. \tag{20}$$

Equation (20) has a form of equation (12) with r and d interchanged. This implies that the Markov semigroup generated by (20) is asymptotically stable and the sequence $z^* = (z_i^*)_{i\geq 1}$ with $z_i^* = i\left(\frac{r}{d}\right)^i$ is stationary. It means that if $z(0) \in l^1$ then there is a constant c_1 such that and $\lim_{t\to\infty} z_i(t) = c_1 i \left(\frac{r}{d}\right)^i$. Therefore $\lim_{t\to\infty} e^{(d-r)t} C_i(t) = c_1$.

A.4 Proof of Theorem 5

We start by proving (8). It follows immediately from (4) that

$$C_i^{(n+1)} = p_U C_i^{(n)} + p_D \sum_{j=0}^{i} C_j^{(n)} C_{i-j}^{(n)}, \qquad \text{for } i \geq 1. \tag{21}$$

We can rewrite (21) as follows for $i \geq 1$

$$C_i^{(n+1)} = \alpha_n C_i^{(n)} + \sum_{j=0}^{i-1} p_D C_j^{(n)} C_{i-j}^{(n)}.$$

Hence, for $i \geq 2$ we have

$$\begin{aligned}
C_i^{(n)} &= \sum_{j=1}^{i-1} \sum_{m=0}^{n-1} p_D \alpha_{m+1} \cdots \alpha_{n-1} C_j^{(m)} C_{i-j}^{(m)} \\
&= \sum_{j=1}^{i-1} \sum_{m=0}^{n-1} p_D \frac{\alpha_m \alpha_{m+1} \cdots \alpha_{n-1}}{\alpha_m} C_j^{(m)} C_{i-j}^{(m)} \\
&= \sum_{j=1}^{i-1} \sum_{m=0}^{n-1} p_D \frac{C_1^{(n)}}{C_1^{(m)} \alpha_m} C_j^{(m)} C_{i-j}^{(m)} \\
&= \sum_{j=1}^{i-1} \sum_{m=0}^{n-1} p_D^2 \frac{C_1^{(n)}}{\beta_m \alpha_m^2} C_j^{(m)} C_{i-j}^{(m)} \\
&= C_1^{(n)} \sum_{j=1}^{i-1} \sum_{m=0}^{n-1} \frac{p_D C_j^{(m)}}{\beta_m \alpha_m} \frac{p_D C_{i-j}^{(m)}}{\beta_m \alpha_m} \beta_m \\
&= C_1^{(n)} \sum_{j=1}^{i-1} \sum_{m=0}^{n-1} \xi_m(j) \xi_m(i-j) \beta_m.
\end{aligned}$$

This yields (8).

It follows from (8) that for every $i \geq 2$ and for all $n \geq 0$,

$$\xi_n(i) \leq \xi_{n+1}(i).$$

Hence it suffices to show that for every $i \geq 2$, the sequence $(\xi_n(i))_{n=0,1,2,\ldots}$ is bounded from above by a certain constant B_i. We prove this statement by induction on i. In order to prove the statement for $i = 2$ let

$$\gamma = \begin{cases} p_U + 2p_D & \text{if } p_D \leq p_R, \\ p_U + 2p_R & \text{if } p_R < p_D. \end{cases}$$

Clearly we have $\gamma < 1$ iff $p_D \neq p_R$. Moreover, it follows from (5) that $\alpha_n \to \gamma$. Since $\alpha_n \leq \alpha_{n+1}$, we have $\alpha_n \leq \gamma$, for all n. Hence

$$\begin{aligned}
\xi(2) &= \sum_{m<\infty} \beta_m \\
&\leq \frac{p_D}{\alpha_0} \sum_{m<\infty} \gamma^m \\
&= \frac{p_D}{p_U(1-\gamma)}.
\end{aligned}$$

Assume the statement for all $i' < i$. Without loss of generality we may assume that $B_{i'} \leq B_{i''}$ holds for all $i' \leq i'' < i$. By (8) we have

$$\begin{aligned}\xi_n(i) &= \sum_{j=1}^{i-1} \sum_{m=0}^{n-1} \xi_m(j)\xi_m(i-j)\beta_m \\ &\leq (i-1)(B_{i-1})^2 \sum_{m=0}^{n-1} \beta_m \\ &\leq (i-1)(B_{i-1})^2 \xi(2).\end{aligned}$$

Hence, we have $\xi(i) < \infty$ for all $i \geq 1$.

Since $\beta_k > \beta_{k+1}$ for all $k \in \mathbb{N}$, thus we have

$$\xi(i) \geq \xi_i(i) \geq (\beta_i)^{i-1} > 0\ ,$$

for $i \geq 1$.

This completes the proof of Theorem 5.

A.5 Proof of Theorem 6

We first need the following two lemmas, where the former is a well known result in dynamical systems [1]. For the sake of completeness we give the proof of this result too.

Lemma 1. *Let $\varphi : (\alpha, \beta) \to \mathbb{R}$ be a C^2-function and $x_0 \in (\alpha, \beta)$. Assume that $\varphi(x_0) = x_0$, $|\varphi'(x_0)| = \mu$, $0 < \mu < 1$, $|\varphi(x) - x_0| < |x - x_0|$ and $\varphi(x) \neq x_0$ for $x \neq x_0$. Then for every $x \in (\alpha, \beta)$ there exists the limit*

$$\lim_{n\to\infty} \frac{|\varphi^n(x) - x_0|}{\mu^n} = g(x) \tag{22}$$

and $0 < g(x) < \infty$.

Proof. Without loss of generality we can assume that $x_0 = 0$ and $\varphi(x)x > 0$ for $x \neq 0$. From the asumption $|\varphi(x)| < x$ it follows immediately that $\lim_{n\to\infty} \varphi^n(x) = 0$ for every $x \in (\alpha, \beta)$ and therefore it is sufficient to check (22) for sufficiently small x. We consider the case $x > 0$ (the case $x < 0$ is analogous). We have

$$\log \frac{\varphi^n(x)}{\mu^n x} = \log \frac{\varphi^n(x)}{\mu\varphi^{n-1}(x)} + \cdots + \log \frac{\varphi(x)}{\mu x}.$$

Since $\varphi(x) = \mu x + \frac{1}{2}\varphi''(0)x^2 + o(x^2)$ we have $\varphi(x)/(\mu x) = 1 + cx + o(x)$, where $c = \frac{1}{2}\varphi''(0)/\mu$. Therefore $\log \frac{\varphi(x)}{\mu x} = cx + o(x)$. Observe that for sufficiently small x we have

$$(c-1)\left(\tfrac{\mu}{2}\right)^k x \leq (c-1)\varphi^k(x) \leq \log \frac{\varphi^{k+1}(x)}{\mu\varphi^k(x)} \leq (c+1)\varphi^k(x) \leq (c+1)\left(\tfrac{1+\mu}{2}\right)^k x.$$

and this implies that the series $\sum_{n=1}^{\infty} \log \frac{\varphi^n(x)}{\mu\varphi^{n-1}(x)}$ is convergent. Thus the limit

$$\lim_{n\to\infty} \log \frac{\varphi^n(x)}{\mu^n x}$$

exists. It follows that for sufficiently small x the limit (22) exists and it is a positive real number. This completes the proof. □

Lemma 2. *If $p_D < p_R$, then for every x satisfying $-(p_U+p_R)/p_D < x < p_R/p_D$ we have $\lim_{n\to\infty} \varphi^n(x) = 1$.*

Proof. We have the following identities.

$$\varphi(x) - p_R/p_D = p_D(x - p_R/p_D)(x + (p_U + p_R)/p_D) \tag{23}$$

$$\varphi(x) - 1 = p_D(x - 1)(x + (p_U + p_D)/p_D) \tag{24}$$

$$\varphi(x) - x = p_D(x - 1)(x - p_R/p_D). \tag{25}$$

Hence if $-(p_U + p_D)/p_D < x < 1$, then by (24) and (25) we have $x < \varphi(x) < 1$. In this case $\varphi^n(x)$ has a limit which is the closest fixed point of φ, i.e. 1.

If $1 < x < p_R/p_D$ then again by (24) and (25) we have $1 < \varphi(x) < x$ and by the same argument $\lim_{n\to\infty} \varphi^n(x) = 1$.

Finally, if $-(p_U + p_R)/p_D < x < -(p_U + p_D)/p_D$, then by (23) and (24) we have $1 < \varphi(x) < p_R/p_D$. Thus again $\lim_{n\to\infty} \varphi^n(x) = 1$. This completes the proof of Lemma 2. □

Now we start proving Theorem 6.

We have

$$\frac{\varphi^n(x) - \varphi^n(0)}{1 - \varphi^n(0)} = \frac{(\varphi^n(x) - 1)/\mu^n + (1 - \varphi^n(0))/\mu^n}{(1 - \varphi^n(0))/\mu^n}.$$

Hence by Lemma 1 and Lemma 2 the above limit exists and is finite whenever x satisfies $-(p_U + p_R)/p_D < x < p_R/p_D$. On the other hand it is easy to show that the limit is $+\infty$ for all other x's.

To show that the convergence is almost uniform in the open disk $|z| < a^{-1}$ in complex numbers take any $0 < r < a^{-1}$. We have for all $|z| < r$ the following inequalities

$$|f_n(z)| \leq f_n(|z|) \leq f_n(r) \tag{26}$$

The first inequality follows from non-negative coefficients of f_n and the second from its monotonicity. Since the sequence $f_n(r)$ is converging, the sequence of functions $f_n(z)$ is uniformly bounded on the disk $|z| < r$. According to the Vitali theorem a uniformly bounded sequence of analytic functions which is converging in some set with a limit point in this disk, coverges uniformly to an analytic function $f(z)$ on each compact subsets of this disk. This impies that the sequence $f_n(z)$ coverges uniformly to an analytic function $f(z)$ on each compact subsets of the disk $|z| < a^{-1}$.

Now we prove (9).

$$\frac{\varphi^n(x)-\varphi^n(0)}{1-\varphi^n(0)} = \frac{\varphi^{n-1}(\varphi(x))-\varphi^{n-1}(0)+\varphi^{n-1}(0)-\varphi^n(0)}{1-\varphi^n(0)}$$
$$= \frac{\varphi^{n-1}(\varphi(x))-\varphi^{n-1}(0)}{1-\varphi^{n-1}(0)}\frac{1-\varphi^{n-1}(0)}{1-\varphi^n(0)} + \frac{\varphi^{n-1}(0)-\varphi^n(0)}{1-\varphi^n(0)}.$$

It follows from Lemma 1 that

$$\lim_{n\to\infty}\frac{1-\varphi^{n-1}(0)}{1-\varphi^n(0)} = \frac{1}{\mu}$$

and

$$\lim_{n\to\infty}\frac{\varphi^{n-1}(0)-\varphi^n(0)}{1-\varphi^n(0)} = 1-\frac{1}{\mu}.$$

Hence, by taking limit on both sides we obtain the equality

$$f(x) = f(\varphi(x))/\mu + 1 - 1/\mu,$$

which is equivalent to (9).

A.6 Proof of Proposition 2

It is obvious that for a fixed n the sequence $(\xi_n(i))_{i\geq 1}$ uniquely determines (up to a constant factor) the sequence $(C_i^{(n)})_{i\geq 1}$. Thus if we normalize these sequences we obtain the same probability distribution.

It follows from Theorem 6 that the limit of the generating functions $f_n(x)$ for the distributions determined by the sequences $(C_i^{(n)})_{i\geq 1}$ exists. We have the same for the sequence $(\xi_n(i))_{i\geq 1}$. Moreover, we have to obtain the same limit. Hence we have

$$f(x) = \sum_{i\geq 1}\xi(i)x^i \Big/ \sum_{i\geq 1}\xi(i).$$

In particular we also have

$$f'(0) = 1\Big/\sum_{i\geq 1}\xi(i).$$

This completes the proof of Proposition 2.

A.7 Proof of Theorem 7

We prove the first part of the Theorem. The second part follows easily from the first part. First we show that the limit exists.

Lemma 3. *Let $a = p_D/p_R$ and let $0 < p < 1/(3-a)$. For all $n \geq 1$ we have*

$$\frac{\partial f_n}{\partial p}(p,x) \leq 0, \text{ if } 0 \leq x < a^{-1},$$

and

$$\frac{\partial f_n}{\partial p}(p,x) \geq 0, \text{ if } -a^{-1} < x \leq 0.$$

Proof. Consider the polynomial

$$\psi(p,x) = 1 - \varphi(p, 1-x).$$

Clearly we have

$$\psi^n(p,x) = 1 - \varphi^n(p, 1-x).$$

Thus

$$f_n(p,x) = 1 - \frac{\psi^n(p,1-x)}{\psi^n(p,1)}.$$

Let

$$F_n(p,x) = \frac{\partial \psi^n}{\partial p}(p,x) \cdot \frac{1}{\psi^n(p,x)}.$$

We prove the following property, for all $n \geq 1$, $|x| \leq a^{-1}$, and $0 < p < 1/(3-a)$,

$$\frac{\partial F_n}{\partial x}(p,x) \leq 0 \tag{27}$$

Let $\mu = 1 - p + ap$. We have $\psi(p,x) = \mu x - apx^2$. Thus $\psi^{n+1} = \mu\psi^n - ap[\psi^n]^2$. Hence we have

$$\frac{\partial \psi^{n+1}}{\partial p}(p,x) = [(\mu - 2ap\psi^n(p,x)) \cdot F_n(p,x) - 1 + a - a\psi^n(p,x)] \cdot \psi^n(p,x).$$

Hence

$$F_{n+1}(p,x) = \frac{\mu - 2ap \cdot \psi^n(p,x)}{\mu - ap \cdot \psi^n(p,x)} \cdot F_n(p,x) + \frac{-1 + a - a \cdot \psi^n(p,x)}{\mu - ap \cdot \psi^n(p,x)}.$$

Since for all $p > 0$,

$$\frac{\partial}{\partial x}\left(\frac{\mu - 2apx}{\mu - apx}\right) < 0$$

and since $\psi(p,x)$ is monotone increasing with respect to x for $|x| < a^{-1}$ and $0 < p \leq 1/(3-a)$, we have for all $n \geq 1$

$$\frac{\partial \psi^n}{\partial x}(p,x) \geq 0.$$

It follows that

$$\frac{\partial}{\partial x}\left(\frac{\mu - 2ap \cdot \psi^n(p,x)}{\mu - ap \cdot \psi^n(p,x)}\right) \leq 0.$$

In a similar way we show that

$$\frac{\partial}{\partial x}\left(\frac{-1+a-a\cdot\psi^n(p,x)}{\mu-ap\cdot\psi^n(p,x)}\right)\leq 0.$$

It is easy to show that

$$F_{n+1}=A_n\cdot A_{n-1}\cdots A_1\cdot B_0+A_n\cdot A_{n-1}\cdots A_2\cdot B_1+\cdots+A_n\cdot B_{n-1}+B_n,$$

where

$$A_i=\frac{\mu-2ap\cdot\psi^i(p,x)}{\mu-ap\cdot\psi^i(p,x)}$$

and

$$B_i=\frac{-1+a-a\cdot\psi^i(p,x)}{\mu-ap\cdot\psi^i(p,x)}.$$

Since each A_i and B_i is non-negative and both have non-positive derivatives with respect to x, it follows that for all $n\geq 1$, for all $|x|\leq a^{-1}$, and $0<p<1/(3-a)$,

$$\frac{\partial F_{n+1}}{\partial x}(p,x)\leq 0.$$

This proves (27).

Now we complete the proof of the lemma.

$$\frac{\partial f_n}{\partial p}(p,x)=-\frac{\frac{\partial\psi^n}{\partial p}(p,1-x)\cdot\psi^n(1)-\frac{\partial\psi^n}{\partial p}(p,1)\cdot\psi^n(p,1-x)}{[\psi^n(1)]^2}.$$

It follows from (27) that for all $0<p<1/(3-a)$ we have $F_n(p,1-x)\geq F_n(p,1)$, whenever $0\leq x<a^{-1}$; and $F_n(p,1-x)\leq F_n(p,1)$, whenever $-a^{-1}<x\leq 0$. This completes the proof of Lemma 3. □

It follows from Lemma 3 that the limit in Theorem 7 exists for $x\in(-a^{-1},a^{-1})$. Using once more inequalites (26) and the Vitali theorem we prove that if $p\to 0$ then $f(p,z)$ coverges uniformly to an analytic function $g(z)$ on each compact subsets of the disk $|z|<a^{-1}$. Denote this limit by $g(z)$. Since $f(p,z)$ are analytic functions we have also

$$\lim_{p\to 0}\frac{\partial f}{\partial z}(p,z)=g'(z) \tag{28}$$

uniformly on compact sets in $|z|<a^{-1}$.

Consider equation (9), subtract from both sides $f(x)$ and divide both sides by $\varphi(x)-x$. Writing explicitly the parameter p we obtain the equation

$$\frac{f(p,\varphi(p,x))-f(p,x)}{\varphi(p,x)-x}=\frac{(1-\mu)\cdot(1-f(p,x))}{\varphi(p,x)-x}.$$

Transforming the right hand side of the above equation leads to the following equation:

$$\frac{f(p,\varphi(p,x))-f(p,x)}{\varphi(p,x)-x}=\frac{(1-a)\cdot(1-f(p,x))}{(1-x)\cdot(1-ax)}. \tag{29}$$

We claim that

$$\lim_{p\to 0^+} \frac{f(p,\varphi(p,x)) - f(p,x)}{\varphi(p,x) - x} = g'(x). \tag{30}$$

Indeed, by the mean value theorem we know that there exists $x < \rho_p < \varphi(p,x)$ such that

$$\frac{f(p,\varphi(p,x)) - f(p,x)}{\varphi(p,x) - x} = f'(p,\rho_p).$$

Since $\lim_{p\to 0} \varphi(p,x) = x$, it follows that $\lim_{p\to 0} \rho_p = x$. From (28) it follows that $\lim_{p\to 0^+} f'(p,\rho_p) = f'(0,x) = g'(x)$. Thus we obtain from (29) and (30) the following differential equation

$$g'(x) = \frac{(1-a)\cdot(1-g(x))}{(1-x)\cdot(1-ax)}. \tag{31}$$

This is an equation with split variables and therefore it has a unique solution which satisfies the boundary condition $g(1) = 1$. It is easy to check that the function $g(x) = (1-a)x/(1-ax)$ satisfies (31). This completes the proof of Theorem 7.

Multicoloring: Problems and Techniques

Magnús M. Halldórsson[1] and Guy Kortsarz[2]

[1] Department of Computer Science, University of Iceland, IS-107 Reykjavik, Iceland.
mmh@hi.is
[2] Department of Computer Science, Rutgers University, Camden, NJ 08102.
guyk@camden.rutgers.edu

1 Multicoloring Graphs: Problems, Measures, Applications

A *multicoloring* is an assignment where each vertex is assigned not just a single number (a "color") but a set of numbers. The number of colors assigned to the vertex is specified by the *length* (or *color requirement*) parameter of that vertex in the input. As usual, adjacent vertices cannot receive the same color; thus here, the sets of colors they receive must be disjoint. Multicolorings are therefore proper generalizations of ordinary graph colorings. The purpose of this paper is to summarize some of the techniques that have been developed specifically for obtaining good approximate multicolorings in different classes of graphs.

The multicoloring problem is reserved for the case where there is no restriction on the set of colors that each vertex can receive (except its size) and the objective is to minimize the number of colors used. A different problem is obtained when we require the colors assigned to a vertex to form a contiguous interval – we refer to such an assignment as a *non-preemptive multicoloring*. Yet a different family of problems occurs when we change the objective function; in particular, we will be interested in the minimizing the *sum* of the multicolorings, or the sum of the largest color assigned to each vertex (assuming the colors correspond to the natural numbers).

Applications. Graph coloring has great many applications. One of the classic examples are in timetabling, where we want to assign courses (nodes) to time slots (colors) so that classes that cannot be taught simultaneously (e.g., that share a student/teacher population) are assigned different time slots. If courses are all of the same length, we have an ordinary coloring problem (let us ignore the issue of the number of classrooms). Only in special cases do we have an ordinary graph coloring problem; with lectures of different length, we have a non-preemptive multicoloring problem, while with lectures of identical length but several occurrences within the scheduling time frame, we have a preemptive problem.

Another application is frequency allocation or channel assignment in wireless communication. In a cellular network, communication between mobiles and a base station in a cell is across a narrow frequency channel. Two base stations cannot use the same frequency if it causes interference due to geographic locality.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 25–41, 2004.
© Springer-Verlag Berlin Heidelberg 2004

This is modeled by a graph where the nodes correspond to the base stations and edges represent geographic adjacency [42]. Each node needs to be allocated as many channels, or colors, as there are calls connecting to that base station, resulting in a multicoloring problem.

For many classes of graphs, the multicoloring problem can be translated to the ordinary coloring problem. A vertex v of length $x(v)$ is replaced by a clique of $x(v)$ vertices (connecting a copy of v to a copy of u if u and v are connected in G). This reduction is polynomial if p is polynomial in n, but can often be done implicitly for large values of p. This is one reason why multicolorings appear less often in the literature.

As the timetabling example indicates, practical applications of graph coloring often relate to scheduling. The graph then represents some constraints or conflicts between the jobs that disallow simultaneous execution. One difference with typical scheduling problems is that they tend to involve a fixed number of "machines", rather than allowing for an unbounded number of vertices of the same color. Another difference is that constraints on jobs in scheduling tend to be either non-existent or based on precedence instead of conflicts. Yet, there are several exceptions to these restrictions/differences.

Viewing a problem as a scheduling or as a graph theory problem is not as trivial a issue as it may seem; these are two (overlapping but) different communities with widely different vocabulary and different perspective. It may even be frowned upon to mix metaphors or borrow different concepts. We, however, advocate freedom from denominational canons in order to benefit from the best of both worlds. As we shall see, this will allow us to map a technique from one area to the other and back. We shall intermix the vocabulary, talking equally of vertices and jobs, colors and rounds (or steps), schedules and colorings.

Measures. The possibility of considering different objective functions is one eye-opening product of the scheduling perspective. The second most common objective function is the *sum of completion times,* or its weighted version. This has been considered for (unit-length) graph coloring as the *sum coloring problem:* the colors are positive integers, and the objective is to minimize the sum of the colors assigned to the vertices. In the multicoloring versions, we sum over the vertices the *finish times,* or the last color assigned to that vertex. In the context of dependent jobs in a system, the sum measure has been seen to favor the users (that own the jobs), while the makespan measure is the favorite of the system (that wants to get done quickly). We use the following notation for the different problems:

SC	Minimum sum coloring
pMC	Multichromatic number of G, or preemptive makespan multicoloring
npMC	Non-preemptive makespan, or fewest colors in a contiguous multicoloring
npSMC	Non-preemptive sum multicoloring
pSMC	Preemptive sum multicoloring

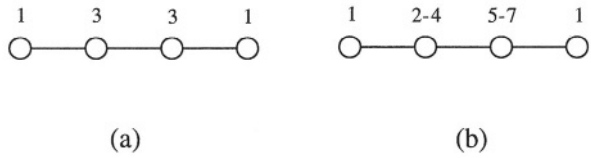


Fig. 1. An example of a path whose optimal sum coloring uses more than the minimum number of colors. (a) Graph with vertex lengths; (b) A minimum sum coloring.

Multicoloring vs. ordinary coloring problems. How different is multicoloring from ordinary graph coloring? We mention some classes of graphs where the difference becomes significant.

Paths. Almost anything is trivial for paths, including all makespan problems and all unit-length coloring problems. However, it is not at all easy to derive a polynomial time algorithm for preemptive sum multicoloring (as can be attested by many false starts by the authors). The current best algorithm due to Kovács runs in time $O(n^3p)$ [31]; a strongly polynomial algorithm is yet to be found.

Trees. Preemptive sum multicoloring has been shown to be strongly NP-hard for trees, even binary trees with polynomially bounded weights [38]. On the other hand, sum coloring yields an easy (but not greedy) linear time algorithm.

Interval graphs. Non-preemptive makespan multicoloring on interval graphs is the Dynamic Storage Allocation problem, which is NP-hard and APX-hard (i.e., hard to approximate within some $c > 1$). The ordinary coloring problem is, however, easily solvable by a greedy method. Sum coloring and sum multicoloring problems are all approximable within some constant; however, the best ratio differs by a factor of as much as 4 (see table).

Perfect graphs. For this class of graphs, we see a difference between preemptive and non-preemptive problems. Preemptive sum multicoloring is approximable within a small constant factor, while no constant factor approximation is known for the non-preemptive case.

1.1 Known Results

One of the most celebrated conjectures of mathematics for a long time was whether all planar graphs could be colored with at most 4 colors. This was finally proved by Appel and Haken [2] with a computer-aided proof, more recently refined in [47]. However, determining whether a planar graph requires 4 colors or not is NP-complete [13]. Exact coloring algorithms have been derived for numerous classes of graphs, one of the most general is that of perfect graphs, due to Grötschel, Lovász, and Schrijver [22]. Another important class of graphs is that of line graphs; coloring line graphs is equivalent to finding an edge coloring

of the underlying graph. This is NP-hard [23] but can be done within an additive one of the trivial lower bound of $\Delta(G)$, the maximum degree of the graph [49].

Minimum preemptive multicoloring is NP-hard to approximate on planar graphs within better than 4/3; this follows from the fact that it is NP-hard to tell if a planar graph is 3-colorable. The problem is known to be hard even on the special class of *hexagon graphs* [40], which are of particular importance for their applications for cellular networks. A 4/3-approximation for minimum multicoloring of hexagon graphs is given in [43]. The coloring algorithm of [22] for perfect graphs extends to multicoloring, and hence it is solvable on all of its subclasses. For line graphs, minimum multicoloring is equivalent to edge coloring multigraphs, which is approximable within a factor of 1.1 [45].

Non-preemptive makespan problems have been considered for different classes of graphs, but under names unrelated to colorings. The npMC problem for interval graphs is better known as *dynamic storage allocation.* Gergov gave an algorithm that uses at most $3\omega(G)$ colors [17]. Buchsbaum et al. [8] recently gave an algorithm with a performance ratio of $2 + \epsilon$, for any $\epsilon > 0$. Non-preemptive makespan of line graphs was studied by Coffman et al. [11] under the name *file transfer problem,* with applications to efficient movement and migration of data on a network.. They showed that a class of greedy algorithms yields a 2-approximation and gave a $(2+\epsilon)$-approximation for a version with more general resource constraints.

The sum coloring problem was first studied by Kubicka [33]. Efficient algorithms have been given for trees [33], partial k-trees [29], and regular bipartite graphs [37]. NP-hardness has been shown for general graphs [35], interval graphs [48], bipartite [6], line [4], planar [25], and cubic planar graphs [37]. Approximation algorithms were studied for sparse graphs [34,4], bounded-degree graphs [4], bipartite graphs [6,18], interval graphs [44,27], comparability graphs [27], perfect graphs [4], planar graphs [25], line graphs [4], while results on hardness of approximation have been shown for general [12,4], bipartite [6], and interval graphs [20]. See Table 1 for a summary of best results known.

Exact and approximate algorithms for multicoloring sum problems have been given for various classes of graphs, as indicated in Table 1. There are hardness results specific to sum multicoloring; the case to date is a recent NP-hardness result of Marx [38] of pSMC on trees.

Results on sum multicoloring problems are all fairly recent and in many cases there are large gaps between the best upper and lower bounds on approximability. Several successes are however prominent:

- Approximation preserving reductions to the maximum independent set problem on any hereditary graph class [5]: within a factor of 4 for sum coloring, and factor 16 for preemptive sum multicoloring.
- Polynomial time approximation schemes (PTAS) for planar graphs (pSMC and npSMC),
- Constant factor approximations for npSMC of line graphs and interval graphs [27], [16].
- Very small factor approximations of sum coloring bipartite graphs [18].

Table 1. Known results for sum (multi-)coloring problems

	SC		SMC	
	u.b.	*l.b.*	pSMC	npSMC
General graphs	$n/\log^2 n$ [4]	$n^{1-\epsilon}$ [4]	$n/\log^2 n$ [5]	$n/\log n$ [5]
Perfect graphs	3.591 [16]	$c > 1$ [6]	5.436 [16]	$O(\log n)$ [5]
Interval graphs	1.796 [27]	$c > 1$ [20]	[16]	$7.682 + \epsilon$ [16]
Bipartite graphs	27/26 [18]	$c > 1$ [6]	1.5 [5]	2.8 [5]
Partial k-trees	1 [29]		PTAS [25]	FPTAS [25]
Planar graphs	PTAS [25]	NPC [25]	PTAS [25]	PTAS [25]
Trees	1 [33]		PTAS [26]	1 [26]
Intersection of k-sets	k [4]		k [5]	$3.591k+.5$ [16]
Line graphs	2 [4]	NPC	2 [5]	7.682 [16]
Line graphs of trees	1		PTAS [39]	

Notation

We use the following symbols in the rest of the text:

$x(v)$	Length (or color requirements) of vertex v
$p = p(G)$	Maximum vertex length
$\chi(G)$	Chromatic number of graph G, ignoring vertex lengths

2 Length Partitioning Technique and npSMC of Planar Graphs

We will consider in this section the npSMC problem for planar graphs, in order to illustrate several of the techniques applicable to multicoloring problems. Unless where otherwise stated, the results are from [25]. We will be aiming towards a polynomial time approximation scheme (PTAS), but in order to get there, we shall be looking at progressively more general special cases. First, however, let us consider some of the more basic approaches.

The first approach might be to ignore the lengths to begin with, apply the quadratic algorithm behind the 4-color theorem [47], and then expand each color class as needed to fit the lengths of the vertices. This can lead to a multicoloring whose sum is arbitrarily worse than optimal. Consider the graph in Fig. 2. The only valid two coloring mixes the white vertices in color classes with the long dark vertices; then, at least half of the white vertices have to wait very long in order to start.

We see that we must give short vertices precedence over long vertices. A reasonable approach would be to color the vertices in groups, shortest-first.

> **Grouping by length:** Divide the vertices into groups of geometrically increasing lengths, and fully color the groups in order of length.

The most natural version is to use powers-of-two as breakpoints between groups, i.e., assign each vertex v to group $\lfloor \lg x(v) \rfloor + 1$. Each group is then colored into

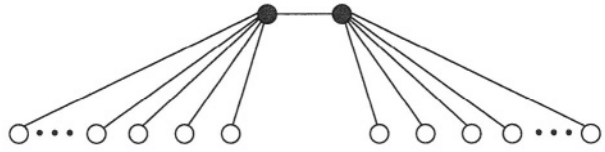


Fig. 2. Example of a planar graph (a tree) whose 2-coloring can lead to an arbitrarily poor sum multicoloring. The many white vertices are short, while the two dark vertices are very long.

$\chi(G)$ sets, each using at most $2^{\lceil \lg x(v) \rceil} - 1$ colors. For instance, vertices of lengths 4,5,6,7 are in group 3, and each of the $\chi(G)$ sets in that group are assigned 7 colors (the largest length of a vertex in the group).

This approach works reasonably well. Observe that group i will be fully colored after at most $\chi(G)[1 + 3 + 7 + \ldots + 2^i - 1]$ colors have been used. This amounts to less than $\chi(G)(2^{i+1} - 1)$. On the other hand, each vertex in group i is of length at least 2^{i-1}. The performance ratio is therefore at most $4\chi(G)$. A closer look can actually reduce this to $2\chi(G)$ [5]. A further improvement is obtained by selecting the base of the geometric sequence *randomly*; this gives the best ratio known of e for non-preemptive sum multicoloring bipartite graphs [5].

A planar graphs are 4-colorable, this length grouping approach gives us a constant factor approximation. We have, however, higher expectations for planarity. We now turn our attention to the unit-length case, the SC problem, as a first step on the road to an approximation ratio arbitrarily close to 1.

Sum Coloring Planar Graphs. The primary technique for approximately solving optimization problems, especially subgraph and partitioning problems, on planar graphs is the *decomposition* technique of Baker [3]. The decomposition theorem says that for any integer k, we can partition the vertices of a planar graph with n vertices into two sets inducing subgraph H_1 and H_2, where H_1 is *k-outerplanar* and H_2 is *outerplanar* with at most n/k vertices. A plane graph is said to be outerplanar if all the vertices lie on the outer (i.e., infinite) face. Outerplanar graphs are also the 1-outerplanar graphs, while a graph is said to be *k-outerplanar* if after removing all vertices on the outer face the graph is $k-1$-outerplanar.

Figure 3 illustrates a planar graph with the vertices on the outer face being emphasized.

The advantage with this decomposition is that outerplanar and k-outerplanar graphs are frequently easy to solve optimally. Baker gave explicit dynamic programming algorithms for many optimization problems on k-outerplanar graphs [3]; e.g., the algorithm to find maximum independent sets runs in time $O(8^k n)$. A more general indirect approach is to use the observation of Bodlaender that k-outerplanar graphs have treewidth at most $3k - 2$ [7], tapping into the vast resource of algorithms on partial k-trees.

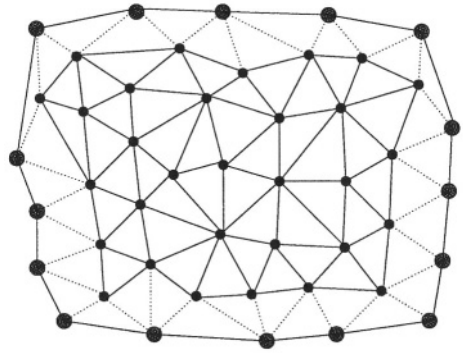


Fig. 3. Planar graph and the outerplanar graph induced by its outer face.

Baker's decomposition proceeds as follows. Let V_1 be the set of vertices on the outer face of the graph; remove this set from the graph. Now recursively apply this rule to obtain sets $V_2, V_3, \ldots$, each inducing an outerplanar graph. Figure 4 illustrates this "peeling of onion skins". It is easy to see that a vertex in a set V_i is adjacent only to vertices in its current set, previous set, or the following set. We now form set $U_i, i = 1, 2, \ldots, k$, as $\cup_t V_{tk+i}$, the union of the layers modulo k. Each of the U_i also induces an outerplanar graph, and at least one of them, say U_p, contains at most n/k vertices. The remaining vertices, $V - U_p$, then form a k-outerplanar graph.

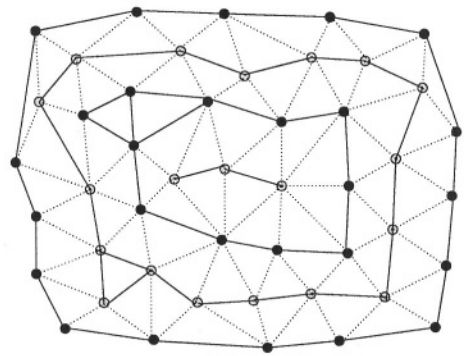


Fig. 4. Partition of a planar graph into a sequence of outerplanar graphs.

We can fairly easily solve sum coloring problems on partial k-trees, using traditional dynamic programming on the tree decomposition, as shown by Jansen [29]. When processing a supernode (a node in the tree decomposition), we want to compute for each possible coloring of the up to k vertices in the supernode, the minimum cost coloring of the subtree of the tree decomposition.

If the maximum color value of a vertex is c, then the total time complexity will be $O(kc^k n)$. We see that it is crucial to bound the number of colors needed.

Kubicka and Schwenk [35] observed that an optimum sum coloring of a tree may require $\Omega(\log n)$ colors, but $O(\log n)$ colors also suffice. A similar bound of $\theta(k \log n)$ was obtained for partial k-trees by Jansen [29]. We shall prove later a more general result for multicolorings. This graph measure, the minimum number of colors in a minimum sum coloring, has been studied more extensively recently as the *strength* of a graph.

We now see that it is easy to apply Baker's decomposition and solve each part optimally in quasi-linear time. The problem is: How do we combine these solutions into a single coloring with of low sum? Intuitively, since H_1 contains the great majority of the vertices, we would want to color those vertices first, and the vertices of H_2 afterwards. However, consider what happens if we wait until H_1 is fully colored for coloring H_2. The cost of coloring H_1 is the minimum chromatic sum of H_1, which is at most the minimum sum of the whole graph G, which is good. However, H_2 may now start to be colored at color $k \log n + 1$. Thus, the cost of coloring H_2 is as much as $n/k \cdot (k \log n + 1) \geq n \log n$. This is certainly much more than the optimal sum (for one thing, 5-coloring G has sum of at most $3n$).

Lemma 1. *After $t\chi(G)$ colors, an optimal sum coloring of graph G has colored all but at most $n/2^t$ vertices.*

> **Truncating a coloring:** If a good sum coloring uses too many colors, it may be preferable to stop that coloring earlier, and revert to a minimum-makespan coloring for the remainder.

The combined strategy is then the following:

1. Apply Baker's decomposition on input graph G, obtaining a k-outerplanar graph H_1 and a smaller graph H_2 with at most n/k vertices.
2. Solve H_1 optimally, using dynamic programming.
3. Use the first $4 \lg k$ colors of the optimal coloring of H_1, leaving at most $n/2^{\lg k} = n/k$ vertices uncolored.
4. Color the remaining at most $n/k + n/k$ vertices using the 5 colors $4 \lg k + 1, \ldots, 4 \lg k + 5$.

The cost of the vertices colored in the first $4 \lg k$ colors is at most the optimal value for G. The cost of the vertices colored later is at most $(4 \lg k + 3)2n/k$. For any given ϵ, choose $k = \theta(\epsilon^{-1} \lg \epsilon^{-1})$ such that this is at most $\epsilon n \leq \epsilon \cdot \text{npSMC}(G)$.

Improved time complexity of sum coloring planar graphs. We indicate how we can improve slightly the time complexity of the PTAS for sum coloring planar graphs of [25].

Instead of finding an optimal sum coloring of H_1, we might as well take into account that we shall only be using the first $4 \lg k$ color classes. Thus, instead, we can search for an optimal *truncated pseudocoloring* with $4 \lg k + 1$ colors; this

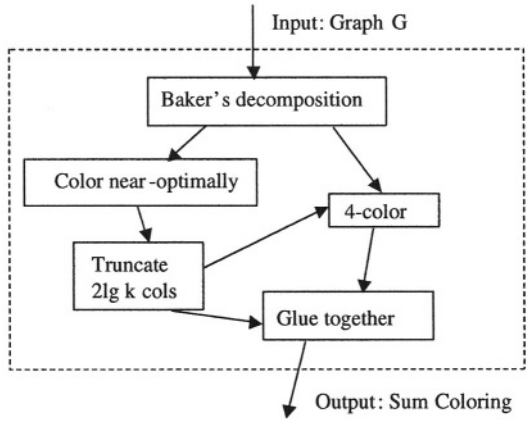


Fig. 5. The schema for sum coloring planar graphs.

is a proper coloring of all of the vertices except those in the last class. Lemma 1 applies as before, for any $t \leq \lg k$, so the last color class contains at most n/k vertices. The cost of such a coloring is at most the optimum chromatic sum of H_1, since any coloring is also a valid truncated pseudocoloring.

The advantage of using a truncated coloring is immediate from our observation of the time complexity of the DP approach being $O(nc^k)$, where c is the value of the largest color used. Here, $c = O(\lg k)$, so the resulting complexity is $O(n2^{k \lg \lg k})$.

Multicoloring with small lengths. What hinders us from applying the same strategy to the multicoloring case? Let us see how far the techniques used so far will take us.

First, we need to bound the number of colors used. A straightforward extension of Lemma 1 shows that after $O(tp\chi(G))$ colors, at most $n/2^t$ vertices remain, and the total number of colors used in an optimal multisum coloring is at most $O(p\chi(G) \log n)$.

The dynamic programming solution of partial k-trees can be applied with minimal changes to non-preemptive multicolorings. The only change needed is to assign each vertex an *interval* of colors instead of a single color. The primary effect is on the complexity, since the number of possible colors is larger. Thus, we can handle combinations of p and k such that $(p \log n)^k$ is polynomial.

This gives us a PTAS for npSMC of planar graphs with polynomially bounded lengths (although not very efficient).

Multicoloring with "almost identical" lengths. What if all the vertex lengths are the same? Is that the same as the unit-length case? In the non-preemptive case, that is indeed true; this can be shown in various ways, e.g., by taking a valid solution, and turning into one where jobs never overlap. (In the preemptive case,

it is not true, although it holds for several special cases like bipartite graphs and cliques.)

More generally, if the lengths are all multiple of a common factor q, we can *scale* the instance by this factor q, i.e. reduce the problem to the instance where all lengths are smaller by a factor of q.

Lemma 2 (Exact non-preemptive scaling). *Let $I = (G, x)$ be a non-preemptive multicoloring instance where for each v, $x(v)$ is divisible by q. Then, $q \cdot \mathtt{npSMC}(I/q) = \mathtt{npSMC}(I)$.*

We can argue by induction that in any optimal coloring of I, all changes happen at colors that are multiples of q. This shows that optimal colorings of I is equivalent to *stretching* an optimal coloring of I/q by a factor of q by repeating each color class q times in order.

What if the vertex lengths are fairly similar? We can then turn to a classical approximation technique from scheduling:

> **Rounding-and-scaling:** If all lengths are greater than r and we round them upwards to a multiple of q, then the increase in the objective function is at most a factor of $1 + q/r$.

This holds independent of the graph and for any convex objective function of the lengths (including multisum and makespan).

Partitioning by length. We have now seen how to handle unit-length instances, and certain restricted kind of multicoloring instances, most generally the case when the ratio between the minimum and maximum length is bounded (by a term that could be a small polynomial in n).

This suggests that we would want to divide the instance into groups of according to length, coloring the "short" vertices before the "long" vertices. We place the vertices on the scale according to length, divide the instance into groups of similar length, color each of them separately, and then "paste" them together in order of length. In order for this to work, we need to ensure that earlier groups do not "delay" the later groups. Basically, if a group starts receiving colors late, it may not matter how efficiently we color it; the resulting coloring will already have become too expensive.

[An aside: One may suggest that instead of coloring the groups in sequence – thus effectively delaying all the vertices in a group until all previous groups have been completed – that we try to color the vertices in the group as early as possible, intermixed with the colorings of the earlier groups. This may well be a good heuristic, but can be hard to analyze; in particular, it would destroy the independence of the solutions of the individual groups. We shall not attempt to pursue that direction here.]

Consider what could happen to a naive partition. A group of maximum length x, requires $\Omega(x)$ colors; indeed, if even if it is 3-colorable, if it contains a triangle with each vertex of length x, we will have to use at least $3x$ colors. This may prove too much for the next group, which may have most of its vertices with

lengths $x+1$. If we start that group at color $3x+1$ – not even accounting for the still earlier groups – that effectively precludes the possibility of a PTAS. Thus, what we need to ensure is that the cost of coloring the earlier groups is small in comparison with the *average length* of vertices in the current group.

We want to find a sequence of breakpoints $b_0 = 1, b_1, \ldots,$, that induce subsets $V_1, V_2, \ldots, V_t$ by $V_i = \{v \in V : x(v) \in (b_{i-1}, b_i]\}$. We find that we can save a logarithmic factor on any arbitrary choices of breakpoints.

Lemma 3 (Length partitioning). *For any $q = q(n)$, we can partition the vertex set into length groups so that the average length of vertices in V_i is at least $(\ln \sqrt{q})b_i$. Further, the groups differ by a factor of at most q, i.e. $b_i \leq b_{i-1} \cdot q$.*

This lemma has an interesting relationship with the classic inequality of Markov from probability theory. Consider any collection of positive numbers $x_1, \ldots, x_n$. Markov inequality shows that at most $1/\ell$ fraction of the elements of the set $X = \{x_1, x_2, \ldots, x_n\}$ are greater than ℓ times the average value $\overline{x}$ (cf. [41]). It is easy to show that this is tight for any fixed value of t; but it cannot be tight for more than one value of t simultaneously. If we are free to choose t from a range of values, the resulting bound on the tail is better; as our lemma shows, it is improved by a logarithmic factor.

Putting together the pieces. This becomes the missing puzzle in our quest for a PTAS.

The combined strategy is illustrated in Figure 6. The length partitioning lemma breaks the vertex set into groups with lengths in a compact interval; the groups are processed independently using a variation of the sum coloring

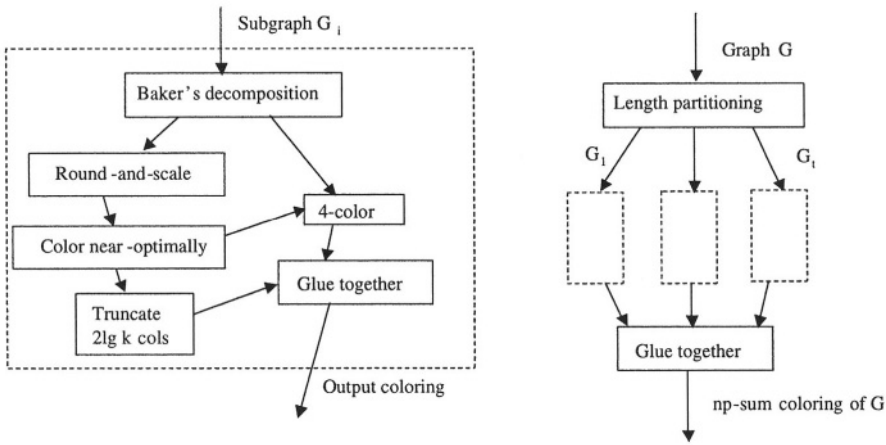


Fig. 6. The schema for non-preemptive sum multicoloring planar graphs. Each small dotted box on the right is an instance of the schema on the right applied to a length-constrained subgraph.

approximation scheme. Finally, the individual solutions are pasted together, in order of the group lengths.

The cost of the multicoloring is derived from two parts: the sum of the costs of the subproblems, and the delays incurred by the colorings of the earlier subproblems. The former is at most $1+\epsilon$ times the optimal cost of coloring each of the subgraphs separately, which is at most $(1+\epsilon)\mathbf{npSMC}(G)$. The main issue is therefore accounting for the contribution of the delays. The number of colors used in each subproblem G_j is at most σb_j, where $\sigma = O(\log k) = O(\log \epsilon^{-1})$. The sum of these is dominated by a geometric series with base of $\sqrt{q}$; thus, the sum of the colors used on the subproblems preceding G_i is at most $(1+1/(\sqrt{q}-1))\sigma b_i$. Here it becomes crucial that the average weight of vertices in each subproblem G_i, and thus the multicolor sum as well, is at least $\ln \sqrt{q} b_i$. Thus the cost incurred by the delays of earlier subproblems are at most $O(\mathbf{npSMC}(G) \cdot \sigma / \ln \sqrt{q})$. In our case we choose $q = e^{(\epsilon^{-1} \ln \epsilon^{-1})^2}$, to make this quantity at most $\mathbf{npSMC}(G) \cdot O(\epsilon)$. The total cost of the solution is therefore $1 + O(\epsilon)$ times optimal, which can be made arbitrarily close to 1.

3 The Delay Technique, npSMC on Line Graphs, and Open-Shop Scheduling

We now describe another general technique applicable in several scenarios. In particular, it is used to approximate the npSMC problem on line graphs, and for approximating open-shop scheduling.

Let $L(v)$ and $S(v)$ denote the sets of shorter and longer neighbors of v, respectively. Let $x(U) = \sum_{u \in U} x(u)$ be the sum of the lengths of vertices in a set U.

Note that for every edge $e = (u, v) \in E$ and in any legal schedule the jobs corresponding to u and v are performed in disjoint rounds. In particular, if the job of v starts (and thus ends) before the job u starts, u has to "wait" $x(v)$ time units before it can start being executed. Say that $x(u) \leq x(v)$. In such a case, a "perfect" algorithm will "manage" to schedule u before v incurring $x(u)$ delay (rather than $x(v)$ delay). Admittedly, some of the delays can happen "together, namely, at the same round several of the neighbors of v are active together. On the other hand, these active neighbors of v must form an independent set. At most two neighbors of any node in a line graph can be active at the same time. This intuition is converted into the following claim [27]. Let $Q(G) = \sum_v x(S(v))$.

Proposition 1. *[27] For a line graph G,*

$$Q(G) \leq 2 \cdot (\mathbf{npSMC}(G) - S(G)).$$

As it turns out, Claim 1 does not suffice to give a "good" approximation. The problem is with the longer neighbors $L(v)$ of a vertex v. Say that $u \in L(v)$ and $x(u) \gg x(v)$. In the non-preemptive scenario, if u executes before v then v has to wait for u to end causing a large delay.

If we disallow $u \in L(v)$ to be executed before v this may cause the independent sets executed at given rounds not to be even maximal independent sets. We adopt an intermediate approach that can be summarized as follows.

1. Before a vertex v can become "active", namely, is executed non-preemptively for $x(v)$ rounds, it has to "pay" $\beta \cdot x(v)$ rounds, where β is some carefully chosen constant.
2. A vertex can only pay if it not executed and has no active neighbor.

Thus the paying of $\beta \cdot x(v)$ rounds is a way of disallowing long jobs to have early process starting times. In fact, the delay "paid" by a job is proportional to its length, hence long jobs wait more.

The algorithm used in [27] is implied by the following additional rules:

- At each round, the union of the active and paying vertices is a *maximal* independent set. Thus, two vertices paying at the same round cannot be neighbors.
- **The length rule:** A vertex v pays in a round if and only if it has neither an active neighbor or a shorter paying neighbor in that round.

Our goal is to prove that the application of this algorithm on a line graph gives a $O(x(S(v))$ finish time for a vertex v. Then, by Proposition 1 an $O(1)$-approximation algorithm for npSMC is implied. In given round, a vertex v can either be

1. active,
2. paying, or
3. neither paying nor active, in which case it is *delayed.*

It is easy to account for the contribution of rounds where v is active or paying. It remains to check rounds where v is delayed. By definition, v can be delayed because either the round contains a paying $S(v)$ vertex or the round contains an active neighbor of v.

Again, it is easy to account for rounds where v is delayed by a shorter neighbor (paying or active); see Proposition 1. Hence the only "problematic" rounds are rounds that contain a longer active neighbor.

Let $L'(v)$ be the vertices of $L(v)$ that became active before v. Observe that by definition, before becoming active, these $L'(v)$ vertices paid a total of $\beta \cdot x(L'(v))$ units (while some of those units were simultaneously paid). The following claim bounds $x(L'(v))$ by $O(x(S(v))$. Thus, we can use Claim 1 to get a constant factor approximation for npSMC.

Proposition 2. $x(L'(v)) = O(k \cdot x(S(v))$

Proof. Since the vertices of $L'(v)$ needed to pay for a total of $\beta \cdot x(L'(v))$ rounds before becoming active, and since at most 2 neighbors of a vertex in a line graph are active at the same time, there were at least $\beta \cdot x(L'(v))/k$ rounds in which the $L'(v)$ vertices were paying. Call such a round an "important" round.

Consider an important "problematic" round for v, namely, an important round for v in which no $S(v)$ vertex is paying or active. Since v was not chosen to be the paying vertex in an important round, it follows that v must have an active neighbor in such a round; otherwise, by the length rule, it has to be paying. Thus we get:

$$\frac{\beta \cdot x(L'(v))}{k} \leq x(L'(v)) + x(S(v)).$$

Remark: The main property used here is that vertices in $L(v) \setminus L'(v)$ cannot become active before v becomes active.

Hence, we get that $x(L'(v)) = O(k \cdot x(S(v)))$ as required.

A more detailed proof along these lines implies a 12-ratio for npSMC on line graphs.

Open-shop and simultaneous delay. In our context, it is best to describe the non-preemptive open-shop scheduling problem as follows. We are given a bipartite graph $G(M, J, E)$, (M for machines and J for jobs). Each edge e corresponds to a task and has length $x(e)$. A subset of the tasks (edges) has to be scheduled at every round. The edges scheduled at a given round must be "independent", namely, must induce a matching. We need to schedule non-preemptively all tasks (edges) so that every e is executed non-preemptively for $x(e)$ time units.

In this scenario, vertices $m \in M$ correspond to jobs. A job is completed if all its tasks (edges) complete. Formally, let $m \in M$. The finish time $f(m)$ is the maximum finish time of an edge touching m. The objective function is to minimize $\sum_{m \in M} f(m)$.

The open shop scheduling problem resembles the npSMC problem on line graphs (because a round is an independent set of edges). The main difference is that we sum the finish times of vertices (of the underlying graph, of which we take the line graph) and not of edges. In that respect, the open-shop problem resembles more the data migration problem (see [30]). In [15] the delay method is used combined with LP techniques to give improved approximation. The problem is relaxed to a fractional linear program. The fractional values are used in the delay function. Namely, if an edge e has fractional starting time $\mu(e)$, it is delayed by a function of $\mu(e)$ of rounds. Instead of a "combinatorial" lower bound lemma 1, the fractional LP value is used to prove a lower bound.

One important new idea is used here. In the line graph algorithm, adjacent vertices cannot simultaneously pay at a round (the paying vertices are an independent set). In [15] a simultaneous pay method is used: Adjacent vertices can pay at the same round. While this simultaneous pay method fails to give a good approximation for general line graphs, it succeeds for open shop scheduling, in part because open shop scheduling essentially corresponds to line graphs of bipartite graphs.

Theorem 1. [15] The non-preemptive open-shop scheduling problem admits a 5.055-ratio approximation.

This improves the previous best 5.83-approximation of [46].

References

1. F. Afrati, E. Bampis, A. Fishkin, K. Jansen, and C. Kenyon. Scheduling to minimize the average completion time of dedicated tasks. In *FSTTCS 2000,* LNCS, Delhi.
2. K. Appel, W. Haken. Every planar map is 4-colorable. *Contemporary Mathematics,* Volume 98, 1989.
3. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM* **41**:153–180, Jan. 1994.
4. A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, T. Tamir. On chromatic sums and distributed resource allocation. *Inf. Comp.* **140**:183–202, 1998.
5. A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, H. Shachnai, and R. Salman. Sum multicoloring of graphs. *J. Algorithms* **37**(2):422–450, 2000.
6. A. Bar-Noy and G. Kortsarz. The minimum color-sum of bipartite graphs. *J. Algorithms* **28**:339–365, 1998.
7. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science,* **209**:1–45, 1998.
8. A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold and M. Thorup. OPT versus LOAD in Dynamic Storage Allocation. *STOC'03.*
9. W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* Wiley, 1998.
10. S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved Scheduling Problems For Minsum Criteria. In *Proc. 23rd International Colloquium on Automata, Languages, and Programming (ICALP),* LNCS 1099, 646-657, 1996.
11. E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. LaPaugh. Scheduling File Transfers. *SIAM Journal on Computing,* **14**(3):744-780, 1985.
12. U. Feige and J. Kilian. Zero Knowledge and the Chromatic number. *Journal of Computer and System Sciences,* **57**(2):187-199, October 1998.
13. M. R. Garey, D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* W. H. Freeman, 1979.
14. H. Gabow and O. Kariv. Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM Journal of Computing,* **11**(1), February 1992.
15. R. Gandhi, M. M. Halldórsson, G. Kortsarz and H. Shachnai, Approximating non-preemptive open-shop scheduling and related problems. *ICALP '04.*
16. R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Improved Bounds for Sum Multicoloring and Weighted Completion Time of Dependent Jobs. Unpublished manuscript, 2004.
17. J. Gergov. Algorithms for compile-time memory allocation. *SODA'99.*
18. K. Giaro, R. Janczewski, M. Kubale and M. Malafiejski. A 27/26-approximation algorithm for the chromatic sum coloring of bipartite graphs. *APPROX,* 131–145, 2002.
19. M. K. Goldberg. Edge Coloring of Multigraphs: Recoloring Technique. *J. Graph Theory,* **8**:121-137, 1984.
20. M. Gonen. Coloring Problems on Interval Graphs and Trees. M.Sc. Thesis, School of Computer Science, The Open Univ., Tel-Aviv, 2001.
21. R. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal,* 45:1563–1581, 1966.
22. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, 1993.

23. I. Holyer. The NP-completeness of Edge-Colouring. *SIAM J. Comput.,* **10**(4), 1981.
24. L. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms. *Mathematics of Operations Research,* **22**:513–544, 1997.
25. M. M. Halldórsson and G. Kortsarz. Tools for multicoloring with applications to planar graphs and partial k-trees. *J. Algorithms,* **42**(2), 334–366, 2002.
26. M. M. Halldórsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai, and J. A. Telle. Multicoloring trees. *Inf. Computation,* **180**(2):113–129, 2003.
27. M. M. Halldórsson, G. Kortsarz, H. Shachnai. Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs. *Algorithmica* **37**:187–209, 2003.
28. H. Hoogeveen, P. Schuurman, and G. Woeginger. Non-approximability results for scheduling problems with minsum criteria. In *Proc. of the 6th International Conference on Integer Programming and Combinatorial Optimization (IPCO),* LNCS 1412, 353–366, 1998.
29. K. Jansen. The optimum cost chromatic partition problem. In *Proc. Third Italian Conference on Algorithms and Complexity (CIAC '97),* LNCS # 1203, pages 25–36, 1997.
30. Y. Kim, Data migration to minimize average completion time. SODA 2003. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA),* pages 97–98, 2003.
31. A. Kovács. Sum-multicoloring on paths. In STACS 2004.
32. M. Kubale. Preemptive versus non preemptive scheduling of biprocessor tasks on dedicated processors. *European J. Operational Research,* **94**:242–251, 1996.
33. E. Kubicka. The chromatic sum of a graph. PhD thesis, Western Michigan University, 1989.
34. E. Kubicka, G. Kubicki, and D. Kountanis. Approximation Algorithms for the Chromatic Sum. In *Proc. First Great Lakes Computer Science Conference,* LNCS 1203, pages 15–21, 1989.
35. E. Kubicka and A. J Schwenk. An Introduction to Chromatic Sums. In *Proc. 17th Annual ACM Computer Science Conference, "Computing trends in the 1990's",* pages 39–45, 1989.
36. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy-Kan, and D. B. Shmoys. Sequencing and Scheduling: Algorithms and Complexity. In S. C. Graves, A. H. G. Rinnooy-Kan, and P. Zipkin, eds., *Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory,* 445–522, 1993.
37. M. Malafiejski. The complexity of the chromatic sum problem on cubic planar graphs and regular graphs. *Electronic Notes in Discrete Mathematics,* **8**, May 2001.
38. D. Marx. The complexity of tree multicolorings. In *Proc. Mathematical Foundations of Computer Science (MFCS),* 532–542, LNCS 2420, Springer, 2002.
39. D. Marx. Minimum sum multicoloring on the edges of trees. In *Proc. 1st Workshop on Approximation and Online Algorithms (WAOA),* 214–226, LNCS 2909, Springer, 2004.
40. C. McDiarmid and B. Reed. Channel assignment and weighted coloring. *Networks,* **36**(2):114–117, 2000.
41. R. Motwani and R. Ragavan *Randomized algorithms.* Cambridge University Press, 1995.

42. L. Narayan. Channel assignment and graph multicoloring. In *Handbook of wireless networks and mobile computing,* pages 71 – 94, Wiley Series On Parallel And Distributed Computing, 2002.
43. L. Narayanan and S. Shende. Static frequency assignment in cellular networks. *Algorithmica,* **29**(3), 396–401, 2001.
44. S. Nicoloso, M. Sarrafzadeh and X. Song. On the sum coloring problem on interval graphs. *Algorithmica,* **23**:109–126,1999.
45. T. Nishizeki and K. Kashiwagi. On the 1.1 edge-coloring of multigraphs. *SIAM Journal on Discrete Mathematics,* **3**(3):391–410, 1990.
46. M. Queyranne and M. Sviridenko. Approximation Algorithms for Shop Scheduling Problems with Minsum Objective. *Journal of Scheduling,* **5**:287–305, 2002.
47. N. Robertson, D.P. Sanders, P. D. Seymour, and R. Thomas. A new proof of the four color theorem. *Electron Res. Announc. Amer. Math. Soc.,* **2**:17-25, 1996.
48. T. Szkaliczki. Routing with Minimum Wire Length in the Dogleg-Free Manhattan Model is NP-complete. *SIAM Journal on Computing,* **29**(1):274–287, 1999.
49. V. G. Vizing. On the estimate of the chromatic class of *p*-graphs. *Diskret. Analiz.,* **3**:23-30,1964.

Some Recent Progress in Algorithmic Randomness

Rod Downey*

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

1 Introduction

Recently there has been exciting progress in our understanding of algorithmic randomness for reals, its calibration, and its connection with classical measures of complexity such as degrees of unsolvability.

In this paper, I will give a biased review of (some of) this progress.

In particular, I will concentrate upon randomness for *reals*. In this paper "real" will mean a member of Cantor space 2^ω. This space is equipped with the topology where the basic clopen sets are $[\sigma] = \{\sigma\alpha : \alpha \in 2^\omega\}$. Such clopen sets have measure $2^{-|\sigma|}$. This space is measure-theoretically identical with the rational interval (0,1), without being homeomorphic spaces.

An important program which began in the early 20th Century was to give a proper mathematical foundation to notion of randomness. In terms of understanding this for probability theory, the work of Kolmogorov and others provides an adequate foundation.

However, another key direction is to attempt to answer this question via notion of randomness in terms of *algorithmic randomness*. Here we try to capture the nature of randomness in terms of algorithmic considerations. (This is implicit in the work on *Kollektivs* in the fundamental paper of von Mises [88].)

There are three basic approaches to algorithmic randomness. They are to characterize randomness in terms of algorithmic predictability ("a random real should have bits that are hard to predict"), algorithmic compressibility ("a random real should have segments that are hard to describe with short programs"), and measure theory ("a random real should pass all reasonable algorithmic statistical tests").

A classic example of the relationship between these three is given by the emergence of what is now called Martin-Löf randomness. For a real $\alpha = .a_1a_2\cdots \in 2^\omega$, a consequence of the law of large numbers is that if α is to be random then $\lim_s \frac{a_1+\cdots+a_s}{s} = \frac{1}{2}$. Consider the null set of reals that *fail* such a test. Then Martin-Löf argued that a real α can only be random if it was not in such a null set. He argued that a random real should pass all such "effectively presented" statistical tests. Thus we define a Martin-Löf test as a computable collection

* Research supported by the Marsden Fund of New Zealand.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 42–83, 2004.
© Springer-Verlag Berlin Heidelberg 2004

$\{U_n : n \in \mathbb{N}\}$ of computably enumerable open sets such that $\mu(U_n) \leq 2^{-n}$. We say that α passes the test iff $\alpha \notin \cap_{n\in\mathbb{N}} U_n$. A real α is called Martin-Löf random or 1-random iff it passes all Martin-Löf tests. (Martin-Löf [77])

It turns out that there are *equivalent* definitions of this notion of randomness in terms of the other paradigms. For instance, Schnorr proved that α is 1-random iff there is a constant c such that for all n, $K(\alpha \upharpoonright n) \geq n - c$. Here K denotes prefix-free Kolmogorov complexity[1].

In this paper we will report on some recent work centered around a program attempting to understand when one real is more random than another and what consequences this has for computability theoretical-aspects of the real. For instance, what does it mean for a real to be "highly random"? If a real is "highly random" what can be said about its Turing degree? Should it be complicated or should it be relatively simple? Does it have high or low information content, etc.

We have seen that we can characterize 1-randomness in terms of initial segment. This suggests natural measures of relative complexity. One real is more random than another if, but some reasonable measure, the initial segment complexity of the first is at least as high as the second on initial segments. We will look at various methods of calibration by initial segment complexity such as those introduced by Solovay [112], Downey, Hirschfeldt, and Nies [30], Downey, Hirschfeldt, and LaForte [27], Downey [21].

These methods are related to lowness notions such as those of Kučera and Terwijn [64], Terwijn and Zambella [119], Nies [91,93,92], Downey, Griffiths and Reid [25]. Here a lowness notion is one that says the real does not help as an oracle. For instance, a real A is Martin-Löf low if the collection of reals 1-random *relative to A* are the *same* as the original 1-randoms. Miller and Yu [86] have demonstrated that this notion also gives rise to natural notions of calibration of randomness such as Miller and Yu's von Lambalgen reducibility. This is intimately related to our earlier work on initial segment complexity.

Finally, this is also related to higher level randomness notions going back to the work of Kurtz [66], Kautz [54], and Solovay [112], and other calibrations of randomness based on definitions along the lines of Schnorr.

These notions all have complex interrelationships, and relationships with classical computability notions such as relative computability and enumerability. For instance, a very exciting by-product of the program is a more-or-less natural

[1] We assume that the reader is somewhat with basic Kolmogorov complexity, and the notion of a prefix-free machine. The reader should recall that a prefix-free machine is a Turing machine M whose domain is a prefix-free set of strings; that is for all σ if $M(\sigma)\downarrow$, then for all strings τ, with $\sigma \prec \tau$, $M(\tau)\uparrow$. Of course, such machine have measurable domain, $\mu(\text{dom } (M)) = \sum_{M(\sigma)\downarrow} 2^{-|\sigma|}$. Prefix-free machines are used in the algorithmic information theory of reals. There is a minimal universal such machine U, in the sense that for all M there is a constant c_M such that for all $K_U(\sigma) \leq K_M(\sigma) + c_M$. Here $K_D(\sigma)$ denotes the Kolmogorov complexity of a string σ relative to a machine D. That is the length of shortest string τ with $D(\tau) = \sigma$, and ∞ if no τ exists. We let $K(\sigma)$ denote $K_U(\sigma)$.

requirement-free solution to Post's problem, much along the lines of the Dekker deficiency set.

There is a large number of open problems in this area many of which are very basic. We will focus on some of these problems.

Given the space restrictions, I will not include any proofs. I refer the reader to Downey, Hirschfeldt, Nies and Terwijn [32] for an expanded version of this article, and to the forthcoming book of Downey and Hirschfeldt [26] for full details.

I will be concentrating upon algorithmic information theory for *reals* and hence be ignoring a lot of new combinatorial results, particularly from the Moscow school, on the Kolmogorov complexity of *strings.* Also ignored is the complexity theoretical work of Lutz and others relating effective Hausdorff and packing dimension to complexity classes. Here, we refer the reader to Ambos-Spies and E. Mayordomo [4] and Lutz [75]. Finally I will not talk about the recent work of Allender and others looking at efficient reductions to the collection of nonramdom strings. (See e.g. Allender et. al. [1])

2 Three Approaches to Randomness

Historically[2], there were three main approaches to the definition of an algorithmically random sequence. They are via what we call

(i) The measure-theoretical paradigm,
(ii) The incompressibility paradigm, and
(iii) The unpredictability paradigm.

2.1 The Measure-Theoretic Paradigm and Stochasticity

The first author to address a possible "definition" of randomness was von Mises. In his remarkable paper [88], von Mises defined a notion of randomness based on such "admissible" stochastic properties, such a random real should have 1's and 0's equally likely, and noted that for any *countable* set of such properties a nonempty notion of randomness resulted. This was early in the 20th century and well before the development of the notion of a computable function. He did not have a canonical choice of such a countable set at hand. After the development of the notion of a computable function, Church made the connection with the theory of computability by suggesting that one should take all *computable* stochastic properties.

Jumping forward, Martin-Löf noted that sets coding such stochastic properties are a special kind of measure zero sets, and that a more general and smooth definition could be obtained by considering all effectively measure zero sets.

[2] In these notes, we will try to avoid discussion of the history of the evolution of the notion of algorithmic randomness. There is a thorough discussion in the monograph of Li and Vitányi [71], the paper of Zvonkin and Levin [129], and van Lambalgen's thesis [67].

The measure-theoretic paradigm is that random reals should be those with no *effectively rare* properties. If a property constitutes an effective null set, then a random real should not have such a property.

A collection of reals that is effectively enumerated is a Σ^0_1-class. We can represent any Σ^0_1-class of reals as $\{[\sigma] : \sigma \in W\}$ for some prefix-free c.e. set W. Now a "test" is a series $\{\mathcal{U}_i\}_{i\in\omega}$ of such Σ^0_1-classes that are shrinking in size. A real passes a test $\{\mathcal{U}_i\}_{i\in\omega}$ if it is not in the intersection $\bigcap_i \mathcal{U}_i$. The main idea is that a random real should pass all effective tests. This leads to the following definition.

Definition 1 (Martin-Löf [77]). *A set of reals $\mathcal{A} \subseteq 2^\omega$ is Martin-Löf null (or Σ_1-null) if there is a uniformly c. e. sequence $\{\mathcal{U}_i\}_{i\in\omega}$ of Σ^0_1-classes (called a* Martin-Löf test*) such that $\mu(\mathcal{U}_i) \leq 2^{-i}$ and $\mathcal{A} \subseteq \bigcap_i \mathcal{U}_i$. $A \in 2^\omega$ is Martin-Löf random, or Σ_1-random, or 1-random, if $\{A\}$ is not Σ_1-null.*

Solovay [112] observed that the definition of 1-randomness is equivalent to saying that a real α is 1-random iff for all c.e. collections of intervals $\{[\sigma_i] : i \in \mathbb{N}\}$, if $\sum_i 2^{-|\sigma_i|} < \infty$ then α is in at most finitely many $[\sigma_i]$.

One very interesting fact, due to Martin-Löf, is that there is a *universal* Martin-Löf test $\mathcal{U} = \{U_n : n \in \mathbb{N}\}$. That is, a real α is Martin-Löf random iff $\alpha \notin \bigcap_n U_n$. To see this, given an enumeration $\{V^m_i : i, m \in \mathbb{N}\}$, of partial Martin-Löf tests, with $\mathcal{V}^m = \{V^m_i : i \in \mathbb{N}\}$, let $\mathcal{U} = \{U_n : n \in \mathbb{N}\}$, with $U_n = \bigcup_k V^k_{n+k}$.

2.2 The Incompressibility Paradigm

The most celebrated approach to the notion of a random real is that essentially due to Kolmogorov [58]. Here we regard a *string* as random iff there is no short program to generate the string, meaning that the only way to generate the string is essentially to hardwire the string into the machine. (Consider, 101010 repeated 1000 times could be generated by a short program).

Plain Kolmogorov complexity. Fix a universal Turing machine U. Given a string $\sigma \in 2^{<\omega}$, define the *plain Kolmogorov complexity* of σ as $C(\sigma) = \min\{|\tau| : U(\tau) = \sigma\}$. The basic facts concerning C are that : (i) the choice of U matters only an additive constant in the theory and (ii) for all σ, $C(\sigma) \leq |\sigma| + O(1)$.

We can define σ to be *k-random* if $C(\sigma) \geq |\sigma| - k$. An easy counting argument shows that random strings exist: Given length n, there are $\sum_{i=0}^{n-k-1} 2^{-i} = 2^{n-k} - 1$ programs of length $< n-k$, so there are $2^n - 2^{n-k} + 1$ k-random strings of length n. For every k, the set of k-random strings is an *immune* Π^0_1 set, i.e. it does not contain any infinite c.e. subsets. As a function, C is not computable. If $m(x) = \min\{C(y) : y \geq x\}$, then m is unbounded (because we run out of short programs), but m grows slower than any partial computable function.

We would like to extend the definition of randomness for finite strings to a definition for infinite strings. Naively, we could define a real A to be random

if and only if for some k every $\sigma \preceq A$ would be k-random. However, Martin-Löf showed that such reals do not exist! This can be seen using the following argument of Katseff [53]. Suppose $A\lceil m$ is the n-th binary string. Then from the *length* of $A\lceil n$ we can recover $A\lceil m$, which is of length $\log n$. This gives

$$C(A\lceil n) \leq C(A\lceil^{n}_{m+1}) \leq n - \log n + O(1).$$

Namely first generate $C(A\lceil^{n}_{m+1})$. Compute n from its length $n - \log n$. Then recover $A\lceil m$.

The basic intuition is that the Kolmogorov complexity should be the length of the shortest string σ used to generate τ from the *bits* of σ. The *length* of σ seems to give an additional $\log n$ many bits of information. This idea is explicitly used to demonstrate that using plain Kolmogorov complexity we will always get complexity oscillations.

First Levin and then later Chaitin introduced methods to get around this problem, as we will see in section 2.2.

Prefix-free complexity. Call a Turing machine T a *prefix machine* if $\mathrm{dom}(T) \subseteq 2^{<\omega}$ is prefix-free. It is easy to construct a *universal* prefix machine U:

$$U(\langle m, \sigma\rangle) = T_m(\sigma),$$

for an effective enumeration $T_0, T_1, \ldots$ of all prefix machines. Here $\langle m, \sigma\rangle$ denotes a prefix encoding of the pair (m, σ).

Now we can define the *prefix-free complexity* of a string σ as $K(\sigma) = \min\{|\tau| : U(\tau) = \sigma\}$. Again, the choice of U matters only an additive constant in the theory. The prefix encoding $\widehat{\sigma} = 1^{|\sigma|}0\sigma$ gives $K(\sigma) \leq 2|\sigma| + O(1)$, and the prefix encoding $\widehat{|\sigma|}\sigma$ gives $K(\sigma) \leq |\sigma| + 2\log|\sigma| + O(1)$. Unlike C, K is *subadditive*: $K(x, y) := K(\langle x, y\rangle) \leq K(x) + K(y) + O(1)$.

The fundamental result about K is the following.

Theorem 1 (Counting Theorem, Chaitin [16]).

(i) $\max\{K(x) : |x| = n\} = n + K(n) + O(1)$.

(ii) $|\{x : |x| = n \wedge K(x) \leq n + K(n) - r\}| \leq 2^{n-r+O(1)}$, *where the constant* $O(1)$ *does not depend on* n *and* r.

We are now in a position to define randomness for reals in terms of initial segment complexity.

Definition 2 (Levin [70], Chaitin [16]). *A real* A *is* Levin-Chaitin random *(or Kolmogorov-Levin-Chaitin random) if there is a constant* c *such that* $K(A\lceil n) \geq n - c$ *for every* n.

Again we arrive at the same concept.

Theorem 2 (Schnorr, see Chaitin [16]). $A \in 2^{\omega}$ *is Martin-Löf random if and only if it is Levin-Chaitin random, i.e there is a constant* c *such that* $K(A\lceil n) \geq n - c$ *for every* n.

The proof of Theorem 2 uses a basic tool of the area called Kraft's inequality which Chaitin observed was effective. Suppose that U is a prefix-free machine. Then the domain of U is measurable. Kraft's inequality is a kind of inverse to this.

Theorem 3 (Kraft [59] or Kraft-Chaitin). *Let $d_1, d_2, d_3, \ldots$ be a computable sequence of natural numbers, possibly with repetitions. Suppose that τ_i is a set of target strings, and we effectively enumerate the pairs (which we call Kraft-Chaitin axioms) $\{\langle d_i, \tau_i\rangle : i \in \mathbb{N}\}$. Then there is a prefix-free machine M and strings σ_i of length d_i, with $M(\sigma_i) = \tau_i$, if and only if $\sum_i 2^{-d_i} \leq 1$.*

Schnorr's Theorem, which can be interpreted to mean that *not only is a real Martin-Löf random iff its initial segment complexity is above n, but in fact its complexity will be "well above" n most of the time.*

Theorem 4 (Miller and Yu [86]).

(i) A real α is Martin-Löf random iff

$$\sum_{n\in\mathbb{N}} 2^{n-K(\alpha\upharpoonright n)} < \infty.$$

(ii) If α is Martin-Löf random, then $\sum_{n\in\mathbb{N}} 2^{K(\alpha\upharpoonright n)-n-K(n)} = \infty$.

A consequence of the above and some more analysis, is the following theorem. This theorem generalizes earlier results of Solovay (which only applied to computable functions f).

Theorem 5 (Miller and Yu [86]). *Suppose that f and g are arbitrary functions from $\mathbb{N}$ to $\mathbb{N}$, such that $\sum_{n\in\mathbb{N}} 2^{-f(n)} = \infty$ and $\sum_{n\in\mathbb{N}} 2^{-g(n)} < \infty$. Then*

(i) If α is random, $(\exists^\infty n)[K(\alpha \upharpoonright n) > n + f(n)]$.
(ii) If α is random, $(\exists^\infty n)[K(\alpha \upharpoonright n) > n + K(n) - g(n)]$.

A very longstanding question was whether there was a characterization of 1-randomness in terms of plain complexity C. It was known to Martin-Löf that if a real α had the property that

$$\exists^\infty n(C(\alpha \upharpoonright n) > n - \mathbb{O}(1)),$$

then α was 1-random. Reals that infinitely often hit the maximal plain complexity are now called *Kolmogorov random.*[3] We know that Kolmogorov randomness

[3] There are some problems with terminology here. Kolmogorov did not actually construct or even name such reals, but he was the first more or less to define randomness for *strings* via initial segment plain complexity. The first person to actually construct what we are calling Kolmogorov random strings was Martin-Löf, whose name is already associated with 1-randomness. Schnorr was the first person to show that the notions of Kolmogorov randomness and Martin-Löf randomness were distinct. Again we can't use Schnorr randomness since Schnorr's name is associated with a randomness notion using tests of computable measure. Similar problems occur later with

and 1-randomness are different, although with probability 1, a real is 1-random iff it is Kolmogorov random. As we will later see, work of Nies, Stephan and Terwijn, and of Miller has shown that Kolmogorov randomness coincides with another natural notion of randomness, 2-randomness. In April, 2004, Joe Miller and Liang Yu finally solved the plain complexity question:

Theorem 6 (Miller and Yu [86]). *A real α is 1-random iff for all computable functions g with $\sum_{n\in\mathbb{N}} 2^{-g(n)} < \infty$ there is a constant d such that for all n,*

$$C(\alpha \upharpoonright n) > n - g(n) - d.$$

2.3 The Unpredictability Paradigm

Perhaps the most common intuition of randomness for a sequence such as coin tosses, is that the sequence is "unpredictable". In particular, a real $\alpha = .a_0a_1\ldots$ would be random if we could not predict any bits of it given others. One way to implement this idea is to use martingales. To formalize this intuition, Schnorr effectivized a notion of betting strategy introduced by Ville to describe Lebesgue measure.

Definition 3. *A* martingale *is a function $d : 2^{<\omega} \to \mathbb{R}^+$ that satisfies for every $\sigma \in 2^{<\omega}$ the averaging condition*

$$2d(\sigma) = d(\sigma 0) + d(\sigma 1). \tag{1}$$

Similarly, d is a supermartingale *if d satisfies*

$$2d(\sigma) \geq d(\sigma 0) + d(\sigma 1). \tag{2}$$

A (super)martingale d succeeds on *a set A if $\limsup_{n\to\infty} d(A\upharpoonright n) = \infty$. We say that d succeeds on, or* covers, *a class $\mathcal{A} \subseteq 2^\omega$ if d succeeds on every $A \in \mathcal{A}$. The* success set *$S[d]$ of d is the class of all sets on which d succeeds.*

The reader should think of a martingale as a betting strategy. The function d is assigning a proportion of the capital to bet on the string σ. The success set is thus the collection of reals upon which we get to make infinite capital.

The following classical result shows how the concept of a martingale relates to measurability.

what we call strongly Chaitin random reals. These were never defined by Chaitin, nor constructed by him. They were first constructed by Solovay who has yet another well-known notion of randomness associated with him which is equivalent to 1-randomness. However, again Chaitin *did* look at the associated notion for *finite* strings, where he proved the fundamental lemma that $K(\sigma) \leq n + K(|\sigma|) + \mathbb{O}(1)$ which allows for the definition of the reals. It is also known that Loveland in his 1969 ACM paper proposed equivalent notions via uniform Kolmogorov complexity. Again, Loveland's name is commonly associated with yet another notion of complexity Kolmogorov-Loveland stochasticity which we look at in Section 9. Even there, there is a randomness notion Kolmogorov-Loveland randomness which is associated with Kolmogorov-Loveland nonmonotonic betting, and neither Kolmogorov nor Loveland considered such martingale-type operations, the randomness notion apparently first articulated by An. A. Muchnik, A. Semenov, and V. Uspensky in [83].

Theorem 7 (Ville [121]). *For any class $\mathcal{A} \subseteq 2^\omega$ the following statements are equivalent:*

(i) *$\mathcal{A}$ has Lebesgue measure zero,*
(ii) *There exists a martingale that succeeds on $\mathcal{A}$.*

Martingales will prove important when we look at refinements of classical Martin-Löf randomness. Martingales are the key to looking at measure and Hausdorff dimension in small classes such as polynomial time, a fact first realized by Lutz [73]. (See also Lutz [76].) Roughly speaking, the reason for this is that there is no analog of Kraft-Chaitin in such classes.

Schnorr's idea was to effectivize the notion above to look at *effective betting strategies.*

Definition 4 (Schnorr). *We will define a (super-)martingale d as being effective, Σ_1, or* computably enumerable *if $d(\sigma)$ is uniformly a c. e. real, and at every stage we have effective approximations to d in the sense that $d(\sigma) = \lim_s d_s(\sigma)$, with $d_s(\sigma)$ a uniformly computable increasing sequence of rationals.*

We remark that the reader might have expected that an effective martingale would be one with f a computable function rather than one with computable *approximations.* We will soon return to this very important point when we consider Schnorr and computable randomness.

Schnorr proved the following effective version of Ville's Theorem 7:

Theorem 8 (Schnorr [104], Satz 5.3). *$\mathcal{A} \subseteq 2^\omega$ is Martin-Löf null if and only if there is a Σ_1-supermartingale d such that $\mathcal{A} \subseteq S[d]$.*

Now Σ_1-supermartingales are Σ_1 objects, so it comes as no surprise that we can effectively list all of them in an enumeration $d_0, d_1, \ldots$ The sum $d = \sum_i 2^{-i} d_i$ is then again a Σ_1-supermartingale, and the success set $S[d]$ is the maximal Martin-Löf null set, so d is a *universal* Σ_1-supermartingale. Additionally, this is a *minimal* Σ_1-supermartingale, in the sense that for any other nontrivial Σ_1-supermartingale d', there is a c, such that for all σ, $cd'(\sigma) \geq d(\sigma)$. This result is also implicit in Levin's construction of a universal c.e. semimeasure (see Zvonkin and Levin [129]). We remark (see Downey and Griffiths [22]) that there is no minimal universal *martingale.*

Thus we see that 1-randomness is a mathematically robust notion in that it has 3 natural definitions. Reals which are random are effectively unpredictable, their segments are incompressible, and they avoid all effective null sets.

John Hitchcock made an interesting observation here. Computably enumerable (super-)martingales are monotonically non-decreasing in the amount you bet at any node σ. But this does not mean that the *side* the better might favour cant' change. To wit, suppose that the beginning of time we might have $f_0(\sigma) = 1$ so there is no preference. Then as we go along, we might raise $f_s(\sigma 0)$ to be above $f_s(\sigma 1)$ at some stage. Perhaps we set $f_s(\sigma 0) = 2$ whilst keeping $f_s(\sigma 1) = 1$. However, at some even later stage $t > s$ we *might* eventually raise $f_t(\sigma 1) > f_t(\sigma 0)$. For example, we might keep $f_t(\sigma 0) = 2$ but raise $f_t(\sigma 1) = 4$. Of course this has a

consequential effect on a lower bound for the value of $f_t(\sigma)$, but the point is, *we can change our minds where we think we should play our capital. At some stage we might favour the left branch and at some later stage the right, or vise versa.* Suppose we define a *Hitchgale* to be a Σ_1-supermartingale f, where the *ratios* are Σ_1, so that such mind changes are now not possible. That is, we suppose that there is an additional computably enumerable function r such that for all s and all σ, and $i \in \{0,1\}$,

$$r_s(\sigma\hat{}i) = \frac{f_s(\sigma\hat{}i)}{f_s(\sigma)}.$$

Note that in a Hitchgale, once we decide that we will be more than half of our capital on one side we are forced to do that henceforth. We say a real is Hitchcock random iff no Σ_1-Hitchgale succeeds upon it.

Question 1. **(Hitchcock)**[4] Is every Hitchcock random real Martin-Löf random?

Anticipating later definitions, we remark that if you replace supermartingale in the definition by martingale, then the definition coincides with computable randomness. Also Downey and Stephan have independently claimed that there are Hitchcock random reals that are not computably random, although no proofs have appeared.

3 Computably Enumerable Reals

In the same way that the domains of partial computable functions, the computably enumerable *sets,* are central players in classical computability theory, the *measures* of the domains of prefix-free machines are important in algorithmic information theory. These are precisely the reals α for which there exists a computable increasing sequence of rationals $\{q_i : i \in \mathbb{N}\}$ with $q_i \to \alpha$. In the literature, such reals are sometimes called *left computable.* An equivalent definition is that the left cut of α is a c.e. set of rationals.

A classical example of a computably enumerable real is the most famous explicit 1-random real, Chaitin's Ω:

$$\Omega = \sum_{U(\sigma)\downarrow} 2^{-|\sigma|} = \mu(\mathrm{dom}(U)),$$

where U is a universal prefix-free machine. This is called the *halting probability.*

Here is a short proof that Ω is random. Let us use the recursion theorem to build a prefix-free machine M which is coded in U with coding constant e. Thus if we enumerate $M(\sigma) = \tau$, then in U the effect will be that U enumerates $U(1^e0\sigma) = \tau$. At stage s, if we see $K_U(\Omega_s \upharpoonright n) < n - e$, say $U_s(\nu) = \Omega_s \upharpoonright n$ with $|\nu| < n - e$, then we will in M declare that $M_{s+1}(\nu) = \Omega_s \upharpoonright n$, causing $\Omega \upharpoonright n \neq \Omega_s \upharpoonright n$.

[4] Of course, Hitchcock did not use the term "Hitchcock random," or "Hitchgale."

The analog of Ω in classical computability is the halting set $K = \{i : \varphi_i(i)\downarrow\}$. Of course we should really say that Ω_U is *a* halting probability, rather than *the* halting probability. But in classical computability we usually talk about *the* halting problem. The reason for this is that we can define an appropriate reducibility m-reducibility, and show that all halting sets are the same up to m-degree. Indeed they are all of the same 1-degree. In the next section we will show that all 1-random c.e. reals are the same up to an analytic version of m-reducibility we call Solovay reducibility.

There has been quite a bit of interest in computability theoretical aspects of computably enumerable reals. For instance, we can look at the degrees of sets A such that $\{q_i : i \in A\}$ *represents* α in the sense that $q_i \to \alpha$. It turns out that such computably enumerable A more or less correspond to (up to m-degree) splittings of the lower cut of α. We refer the reader here to Calude [10], Calude, Coles, Hertling, and Khoussainov, [12], Downey [21], Downey and Hirschfeldt [26].

Another way to generate such reals is via their *presentations.* A prefix free c.e. set A is said to *present* α if $\mu(A) = \alpha$. Downey and LaForte [35] constructed a c.e. noncomputable real with only computable presentations. Downey and LaForte also established that the degrees of presentations was essentially related to the *weak truth table* degrees below that of α. Downey and Terwijn [39] clarified this situation by showing that any appropriate "Σ^0_3 ideal" in the c.e. wtt degrees could be realized as the wtt degrees of presentations.

Authors have also studied classes more general than the c.e. reals. For instance, the field generated by the c.e. reals is in fact the collection of reals of the form $\alpha - \beta$ with α and β c.e. reals. (Ambos-Spies, Weihrauch and Zheng [7].) This field is thus called the field of d.c.e. reals. Recently the following was shown.

Theorem 9 (Ng Keng Meng [90], Raichev [100]). *The field of d.c.e. reals is real closed*[5].

The Turing degrees containing d.c.e. reals include the ω-c.e. degrees (in the sense of the Ershov Hierarchy) (Downey, Wu, Zheng [41]), as well as some other Δ^0_2 degrees, (Zheng [128]) but there are Δ^0_2 degrees with no d.c.e. reals (Downey, Wu, Zheng [41]). Finally Ho [48] observed that the reals which are the limits (not monotone) of computable sequences of rationals are exactly the Δ^0_2 reals. Zheng and Wu have explored other notions of convergence of reals. Little else is known.

4 Solovay Reducibility and 1-Random c.e. Reals

In the last section we saw that the halting probability Ω is a c.e. real that it is that was 1-random. The analog of Ω in classical computability is the halting set $K = \{i : \varphi_i(i)\downarrow\}$. Of course we should really say that Ω_U is *a* halting

[5] Actually Raichev showed something stronger. Under the rK-reducibility we later define, he showed that the collection of reals $\leq_{rK} \beta$ for any β is always real closed, and that this $\leq_{rK}$ result implies the d.c.e. one.

probability, rather than *the* halting probability. But in classical computability we usually talk about *the* halting problem, and this situation is analogous. Of course underlying all of this is Myhill's Theorem which says that all halting problems are essentially the same. As we promised, in the present section, we will address this situation for c.e. reals and randomness.

Solovay [112] recognized the need for an analog of Myhill's Theorem for c.e. reals. He sought to introduce appropriate reducibilities to attempt to prove such an analog. Whilst Solovay's program was only recently realized by the joint work of several authors, Solovay reducibility will be our starting point.

Definition 5 (Solovay [112]). *We say that a real α is* Solovay reducible *to β (or β* dominates *α), $\alpha \leq_S \beta$ iff there is a constant c and a partial computable function f, so that for all $q \in \mathbb{Q}$, with $q < \beta$,*

$$c(\beta - q) > \alpha - f(q).$$

The intuition is that a sequence of rationals converging to β can be used to generate one converging to α at the same rate. The point is that if we have a c.e. sequence $\{q_n : n \in \omega\}$ of rationals converging to β then we know that $f(q_n)\downarrow$. Notice that if $r_n \to \alpha$ then for all m there is some k such that $\alpha > r_k > f(q_m)$. (The reals are not rational.) Noticing this yields the following characterization of Solovay reducibility.

Lemma 1 (Calude, Coles, Hertling, Khoussainov [12]). *For c.e. reals, $\alpha \leq_S \beta$ iff for all c. e. sequence or rationals $q_i \to \beta$ there exists a total computable g, and a constant c, such that, and c.e. rationals $r_i \to \alpha$, such that, for all m,*

$$c(\beta - q_m) > \alpha - r_{g(m)}.$$

Solovay [112] observed that this "analytic" version of m-reducibility was enough to establish many properties of Ω, for c.e. reals, in their sense that if $\Omega \leq_S \alpha$ then α is 1-random. The key property allowing us to see this is the following.

Lemma 2 (Solovay [112]). *$\leq_S$ is both a C- and a K- measure of relative randomness in the sense that for $I \in \{C, K\}$, we have*

$$\alpha \leq_S \beta \text{ implies } (\exists c)(\forall n)[I(\alpha \upharpoonright n) \leq I(\beta \upharpoonright n) + c].$$

Using Kraft-Chaitin, Calude, Hertling, Khoussainov, and Wang [13] proved that if α is *Ω-like* (in the sense that $\Omega \leq_S \alpha$ and α is c.e.) then α is the halting probability of a universal prefix-free machine:

Theorem 10 (Calude, Hertling, Khoussainov, Wang [13]). *Let α be a c.e. real such that $\Omega \leq_S \alpha$. Then α is a halting probability. That is, there is a universal machine $\widehat{U}$ such that $\mu(\mathrm{dom}(\widehat{U})) = \alpha$.*

Kučera and Slaman finished the story by proving the following.

Theorem 11 (Kučera and Slaman [63]). *Suppose that α is 1-random and c.e. Then α is Solovay complete meaning that if β is a c.e. real, then $\beta \leq_S \alpha$.*

Theorem 11 gives great insight into the structure of c.e. random reals. Notice that to "qualify" as being random all we need is that the real α has $K(\alpha \upharpoonright n) \geq n - O(1)$ for all n. But of course $K(\sigma)$ can be near $|\sigma| + \log|\sigma|$. If fact we know that Ω must have oscillations above this. And, indeed by the work of Miller and Yu, all 1-random reals have such oscillations. The Kučera-Slaman theorem says that for c.e. random reals, there is only *one* and everything oscillates above n at essentially the same rate!

We were motivated to understand the structure of c.e. reals under the $\leq_S$. Naturally, the reducibility gives rise to equivalence classes, called *degrees.* Despite the many attractive features of the Solovay degrees of c.e. reals, their structure is largely unknown. Recently progress has been made.

Theorem 12 (Downey, Hirschfeldt and Nies [30]).
The Solovay degrees of c.e. reals

(i) form a distributive upper semilattice, where the operation of join is induced by +, arithmetic addition (or multiplication) (namely $[x] \vee [y] \equiv_S [x + y]$.*)*
(ii) are dense,
(iii) If $\mathbf{a}$ *is incomplete and* $\mathbf{b} <_S \mathbf{a}$*, then there exist* $\mathbf{a_1}|_S\mathbf{a_2}$ *such that* $\mathbf{b} < \mathbf{a_1}, \mathbf{a_2}$*, and* $\mathbf{a} = \mathbf{a_1} \vee \mathbf{a_2}$*. That is every incomplete degree splits over all lesser ones.*
(iv) If $[\Omega] = \mathbf{a} \vee \mathbf{b}$ *then either* $[\Omega] = \mathbf{a}$ *or* $[\Omega] = \mathbf{b}$.

We point out that item (iv) above demonstrates that random c.e. reals are qualitatively different from all others in the sense that they cannot be split into two smaller c.e. reals. It is important to realize that this is only true of c.e. reals. The fact that + induces the join operation reveals another definition of Solovay reducibility on c.e. reals.

Theorem 13 (Downey, Hirschfeldt, Nies [30]). *For c.e. reals,* $\alpha \leq_S \beta$ *iff there exists a* $\mathbf{c}$ *and a c.e. real* γ *such that*

$$c\beta = \alpha + \gamma.$$

Before we leave the Solovay degrees of c.e. reals, we note that the structure is very complicated.

Theorem 14 (Downey, Hirschfeldt, LaForte [28]). *The first-order theory of the Solovay degrees of c.e. reals is undecidable.*

The proof of Theorem 14 uses Nies' method of interpreting effectively dense boolean algebras, together with a technical construction of a certain class of (strongly) c.e. reals. Calude and Nies [14] have proven that the c.e. 1-random reals are all *wtt*-complete. Earlier Kučera [60] has used Arslanov's completeness criterion to show that all 1-random c.e. reals are Turing complete. This also follows from a result in Downey, Hirschfeldt, LaForte [27] that $\alpha \leq_S \beta$ *implies* $\alpha \leq_{wtt} \beta$. If we combine this with the result of Demuth below, and the fact that if a 1-random has c.e. degree then it is Turing complete, we see that *whilst* Ω *is wtt-complete, it is tt-incompatible with all c.e. sets.*

Theorem 15 (Demuth). *If α is 1-random and $\beta \leq_{tt} \alpha$, then there is a 1-random set $\widehat{\beta} \equiv_T \beta$.*

A proof of Demuth's Theorem can be found in Kautz [54], and Downey and Hirschfeldt [26]. Very little else is known about the Solovay degrees of c.e. reals.

A major philosophical question about c.e. reals is whether they are *red herrings.* The material in the present section would suggest that 1-random reals are computationally complex in the sense that, like the halting problem, they encode a lot of knowledge. Papers by Chaitin suggest that "Ω is the number of knowledge."

However, recent work will show that for more highly random reals, and indeed a "random" random real they are computationally very weak. One might think that therefore c.e. reals are red herrings in the sense that they are completely atypical.

The following result says that this is in another sense completely false.

Theorem 16 (Downey, Hirschfeldt, Miller, Nies [24]). *With probability 1, for a real A there is a set B such that $A =^* \Omega^B$.*

That is almost all random reals are in fact halting probabilities relative to *some* oracle! This seems very strange indeed. The answer lies in the fact that whilst Ω is a c.e. operator, it is not a CEA (computably enumerable in *and above*) one!

5 Other Measures of Relative Randomness

A pre-ordering $\leq$ on reals is an initial segment *K-measure of relative randomness* if it satisfies the *Solovay property:*

$$\text{If } \beta \leq \alpha \text{ then } (\exists c)(\forall n)\big[K(\beta \upharpoonright n) \leq K(\alpha \upharpoonright n) + c\big].$$

Solovay reducibility is a C or K measure of relative randomness, but it is certainly not the only one. Moreover it is quite restrictive since it is too fine and uniform (as we see) and it really only makes sense for c.e. reals. It is quite easy to construct a real α and a *computable* set β with $\beta \not\leq_S \alpha$.

Downey, LaForte and Hirschfeldt [27] defined several new reducibilities which were measures of relative randomness.

Definition 6. *$\beta \leq_{sw} \alpha$ if there is a functional Γ such that $\Gamma^\alpha = \beta$ and the use of Γ is bounded by $x + c$ for some c.*

Again it is not difficult to prove that $\leq_{sw}$ is a measure of relative C– or K–complexity. This reducibility is quite close to one looked at by Soare, Nabutovsky and Weinberger see [109], the difference being that the constant must be 0 in their case. $\leq_{sw}$ agrees with $\leq_S$ on the c.e. *sets.* Furthermore if α is a c.e. real which is noncomputable, then there is a noncomputable strongly c.e. real $\beta \leq_{sw} \alpha$, and Downey, Hirschfeldt and LaForte [27] showed that this is *not* true in general, for $\leq_S$. By and large, however, *sw*-reducibility is very badly behaved, as witnessed by the next result.

Theorem 17 (Downey, Hirschfeldt, LaForte [27]).

(i) The sw-degrees of c.e. reals do not form an upper semilattice.
(ii) sw and S are incomparable in the sense that there exist c.e. reals α, β, γ, δ with $\alpha \leq_S \beta$ and $\alpha \not\leq_{sw} \beta$, and $\gamma \leq_{sw} \delta$ and $\gamma \not\leq_S \delta$.

The next theorem gives significant insight into the relative behavior of 1-random c.e. reals. It says that whilst the initial segment complexity of all random c.e. reals is the same, there is no natural uniform way to get the *bits* of one version of Ω from another.

Theorem 18 (Yu and Ding [124]). *There does not exist an sw-complete c.e. real. Thus the analog of the Kučera-Slaman theorem cannot hold for sw-reducibility.*

There is something we can say about *sw*-reducibility and c.e. *sets.*

Theorem 19 (Downey and Hirschfeldt [26]). *Suppose that α is a c.e. real whose initial segment complexity is (for example) ϵn for some $\epsilon > 0$ and for almost all n. Then for any c.e. set B, $B \leq_{sw} \alpha$ with use $\alpha \upharpoonright m$ for $B(m)$. Thus, not only is Ω wtt-complete, but it is sw-complete for c.e. sets.*

Both S-reducibility and sw-reducibility are uniform in a way that relative initial-segment complexity is not. Motivated by this idea, Downey, Hirschfeldt, and LaForte introduced the following.

Definition 7. *Let α and β be reals. We say that β is* relative K-reducible *(rK-reducible) to α, and write $\beta \leq_{rK} \alpha$, if there exist a partial computable binary function f and a constant k such that for each n there is a $j \leq k$ for which $f(\alpha \upharpoonright n, j)\downarrow = \beta \upharpoonright n$.*

Theorem 20 (Downey, Hirschfeldt, LaForte [27]).

(i) $\leq_{rK}$ is a K- and C- measure of relative randomness.
(ii) Let α and β be c.e. reals. If $\beta \leq_S \alpha$ or $\beta \leq_{sw} \alpha$, then $\beta \leq_{rK} \alpha$.
(iii) A c.e. real α is rK-complete if and only if it is random.
(iv) If $\beta \leq_{rK} \alpha$ then $\beta \leq_T \alpha$.

The most interesting characterization of rK-reducibility (and the reason for its name) is given by the following result, which shows that there is a very natural sense in which rK-reducibility is an exact measure of relative randomness. Recall that the prefix-free complexity $K(\tau \mid \sigma)$ of τ *relative to* σ is the length of the shortest string μ such that $M^{\sigma}(\mu)\downarrow = \tau$, where M is a fixed self-delimiting universal computer. (Similarly for C.)

Theorem 21 (Downey, Hirschfeldt, LaForte [27]). *Let α and β be reals. Then $\beta \leq_{rK} \alpha$ if and only if there is a constant c such that $K(\beta \upharpoonright n \mid \alpha \upharpoonright n) \leq c$ for all n. (And $C(\beta \upharpoonright n \mid \alpha \upharpoonright n) \leq c$.)*

The *rK*-degrees have nice structural properties.

Theorem 22 (Downey, Hirschfeldt, LaForte [27]).

(i) *The rK-degrees of c.e. reals form an upper semilattice with least degree that of the computable sets and highest degree that of* Ω.
(ii) *The join of the rK-degrees of the c.e. reals* α *and* β *is the rK-degree of* $\alpha + \beta$.
(iii) *For any rK-degrees* $\mathbf{a} < \mathbf{b}$ *of c.e. reals there is an rK-degree* $\mathbf{c}$ *of c.e. reals such that* $\mathbf{a} < \mathbf{c} < \mathbf{b}$.
(iv) For *any rK-degrees* $\mathbf{a} < \mathbf{b} < \deg_{rK}(\Omega)$ *of c.e. reals, there are rK-degrees* $\mathbf{c_0}$ *and* $\mathbf{c_1}$ *of c.* e. *reals such that* $\mathbf{a} < \mathbf{c_0}, \mathbf{c_1} < \mathbf{b}$ *and* $\mathbf{c_0} \vee \mathbf{c_1} = \mathbf{b}$.
(v) $deg_{rK}(\Omega)$ *is join inaccessible in the uppersemilattice of c. e. rK-degrees.*

We remark that we do not know if the *rK*-degrees are distributive. The theories of neither the sw- nor the rK- degrees has yet been proven undecidable whilst surely this must be the case.

5.1 The Basic Measures $\leq_K$ and $\leq_C$

Of course, our fundamental interest will be in the measures of relative complexity *defined* by the Solovay properties. Thus we can define e.g. $\alpha \leq_K \beta$ if there is a c such that for all n,

$$K(\alpha \upharpoonright n) \leq K(\beta \upharpoonright n) + c.$$

Note that $\leq_K$ is not really a *reducibility,* but simply a transitive pre-ordering measuring relative complexity. This is best seen by the following result.

Theorem 23 (Yu, Ding, Downey [126]). $|\{\alpha : \alpha \leq_K \Omega\}| = 2^{\aleph_0}$.

Joe Miller has proven that the *K*-degree of Ω, or any other 1-random real, is always countable. (This is consistent since there is no natural join operator.) Presumably this is also true for *C,* but we offer the following as a question:

Question 2. Is the *C*-degree of each 1-random real countable?

In spite of Theorem 23 the following question remains open.

Question 3. Is there a *K*- or a *C*- degree with uncountably many members?

In Theorem 23, we can replace Ω by any 1-random. Later we will see that $\leq_K$ does not imply $\leq_T$ even on the c.e. reals. For $\leq_C$ we have the following.

Theorem 24 (Stephan [114]). *Suppose that* α *and* β *are c.e. reals such that* $\alpha \leq_C \beta$. *Then* $\alpha \leq_T \beta$.

Stephan's Theorem generalizes an old result of Chaitin [16] which says that the sets $\leq_C 1^\omega$ are exactly the computable sets. (which generalizes an even older result of Loveland [72]).

Before we turn to the very interesting relationship of $\leq_K$ to $\leq_T$, we look at the structure of c.e. reals under $\leq_K$.

Theorem 25 (Downey, Hirschfeldt, LaForte [27]).

(i) *The K-degrees of c.e. reals form an upper-semilattice with highest degree that of* Ω.
(ii) *The join of the K-degrees of the c.e. reals* α *and* β *is the K-degree of* $\alpha+\beta$.
(iii) *For any K-degrees* $\mathbf{a} < \mathbf{b}$ *of c.e. reals there is an K-degree* $\mathbf{c}$ *of c.e. reals such that* $\mathbf{a} < \mathbf{c} < \mathbf{b}$.
(iv) *For any K-degrees* $\mathbf{a} < \mathbf{b} < \deg_K(\Omega)$ *of c. e. reals, there are K-degrees* $\mathbf{c_0}$ *and* $\mathbf{c_1}$ *of c.e. reals such that* $\mathbf{a} < \mathbf{c_0}, \mathbf{c_1} < \mathbf{b}$ *and* $\mathbf{c_0} \vee \mathbf{c_1} = \mathbf{b}$.
(v) $\deg_K(\Omega)$ *is join inaccessible in the uppersemilattice of c.e. K-degrees.*

The reader will note the obvious similarities between Theorems 12, 22, and 25. Downey and Hirschfeldt [26] have proven the following generalization.

Theorem 26 (Downey and Hirschfeldt [26]). *Let* $\leq_Q$ *be any measure of relative randomness which is* Σ^0_3 *and having the properties that the* 0*-degree contains the computable sets, + induces the join operation, and the top degree is that of* Ω. *The the following hold for* $\leq_Q$.

(i) *The Q-degrees of c.e. reals form an uppersemilattice with highest degree that of* Ω.
(ii) *For any Q-degrees* $\mathbf{a} < \mathbf{b}$ *of c.e. reals there is an K-degree* $\mathbf{c}$ *of c.e. reals such that* $\mathbf{a} < \mathbf{c} < \mathbf{b}$.
(iii) *For any Q-degrees* $\mathbf{a} < \mathbf{b} < \deg_Q(\Omega)$ *of c.e. reals, there are Q-degrees* $\mathbf{c_0}$ *and* $\mathbf{c_1}$ *of c.e. reals such that* $\mathbf{a} < \mathbf{c_0}, \mathbf{c_1} < \mathbf{b}$ *and* $\mathbf{c_0} \vee \mathbf{c_1} = \mathbf{b}$.
(iv) $\deg_Q(\Omega)$ *is join inaccessible in the uppersemilattice of c.e. Q-degrees.*

At a talk by the first author in Heidelberg, in May 2003, Alexander Shen pointed out that a natural measure of relative randomness would be to replace the constant in the definition by $O(\log n)$. That is, $\alpha \leq \beta$ iff for all n, $K(\alpha \upharpoonright n) \leq K(\beta \upharpoonright n) + O(\log n)$. The reason he suggests this is that, since all the reducibilities are within a log factor of one another it would be independent. We have not looked at this at all, but point at the paper Chernov et al. [18]. Notice that the complexity of the reducibility remains Σ^0_3.

Remark 1 (Monotone Complexity). Another measure not yet explored is the following. Levin introduced *monotone* complexity. Here a machine M is called monotone if its action is continuous: $M(\sigma)\downarrow$ and $M(\tau)\downarrow$ and $\sigma \prec \tau$ imply that $M(\sigma) \prec M(\tau)$. We can prove a characterization of 1-randomness via monotone complexity in the same way. α is 1-random iff its monotone complexity is $n - \mathbb{O}(1)$ on $\alpha \upharpoonright n$. This is, again, a Σ^0_3 measure the top degree is that of Ω and the bottom is the computable sets. It is not very discerning for calibrating the complexity of random reals since all random reals have the same degree! However, the structure of the monotone degrees of c.e. reals is more or less open.

The measures $\leq_C$ and $\leq_K$ are pretty difficult to deal with directly. In view of Theorem 23, it is not even clear whether there are uncountably many K-degrees. This was recently solved by showing that whilst the cardinality of the set of reals K below a given one can be large, the measure is small.

Theorem 27 (Yu, Ding, and Downey [126]). $\mu(\{\alpha : \alpha \leq_K \beta\}) = 0$. *Hence there are uncountably many K-degrees of random reals.*

Using Theorem 27, Yu and Ding established the following.

Theorem 28 (Yu and Ding [125]). *There are* $2^{\aleph_0}$ *many K-degrees of random reals.*

In [31], it is shown that for c.e. reals we have the following.

Theorem 29 (Downey, Hirschfeldt, Nies, and Stephan [31]). *On the K- and C-degrees of c.e. reals, + is a join.*

Notice that since $\leq_K$ and $\leq_C$ are Σ^0_3, this implies that the degree structures restricted to the c.e. reals form a dense uppersemilattice.

Question 4. Are there minimal pairs of $\leq_K$ degrees? Is the structure a lattice on the c.e. reals?

Question 5. Given any α is there a random β with $\alpha \leq_K \beta$?

Question 6. Are there minimal *K*-degrees? The answer is surely no.

A truly fundamental question is the following which remains open in spite of the best efforts of several authors.

Question 7. Do there exist *random* α, β with $\alpha <_I \beta$ with *either* $I \in \{C, K\}$?

I regard this as a *central* question. The reason is that a *negative* answer says something very significant about initial segment complexity *characterizing* randomness. I find it hard to believe that the answer is "no."

There are a number of exciting recent results on $\leq_K$ and $\leq_C$ due to Liang Yu and Joe Miller that we will discuss in the next section.

6 Relativized Randomness

A classical way to calibrate the level of randomness is to increase the complexity of the tests in terms of the arithmetical hierarchy. This material will be of particular interest when we study the relationship between randomness and Turing degrees. The basic result connecting 1-randomness to Turing reducibility is the celebrated result often attributed only to Gács, but actually first proved by Kučera.

Theorem 30 (Kučera [60], Gács [44]). *Every set is wtt-reducible to a Martin-Löf random set.*

The easiest proof of Theorem 30 is the recent one of Merkle and Mihailovic [81].

Other results on the Turing degrees of 1-random sets include Kučera's result [60] that all degrees above $\mathbf{0}'$ contain 1-random reals. Since the collection of reals with initial segment complexity $n - c$ for all n is a Π^0_1 class, there are 1-random reals of low Turing degree. Kučera [60] proved that Δ^0_2 1-random degrees are not closed upwards. For the Δ^0_2 degrees they are closed upwards at least in terms of the jump operator: using a new basis theorem for Π^0_1 classes with no computable members, Downey and Miller [36] proved that for every S which is computably enumerable in, and above $\emptyset'$, there is a Δ^0_2 1-random real A with $A' \equiv_T S$. (This had earlier been claimed in a remark following a construction of a high incomplete 1-random real in each Π^0_1 class of sufficient measure, by by Kučera in [61], Theorem 12 and Remark 8.)

Kučera also observed that 1-randomness was connected to the PA-degrees, the degrees of complete extensions of Peano arithmetic. The picture was recently clarified by Frank Stephan who proved the following.

Theorem 31 (Stephan [115]). *Suppose that* $\mathbf{a}$ *is PA-complete and 1-random. Then* $\mathbf{0}' \leq_T \mathbf{a}$.

Theorem 31 establishes that there are two varieties of randoms. The first variety are those that are computationally rich and can compute the halting problem. The second are those that are computationally feeble and cannot even compute a {0,1} valued fixed-point free function. As we see below, this means all 2-randoms (see below) are computationally weak.

The basic definition of 1-randomness can be generalized quite easily. For our purposes, we have the following noticed by many such as Solovay, Kurtz, and Martin-Löf:

Definition 8.

(i) *A* Σ^0_n–test *is a computable collection* $\{V_n : n \in \mathbb{N}\}$ *of* Σ^0_n *classes such that* $\mu(V_k) \leq 2^{-k}$.
(ii) *A real* α *is* Σ^0_n*-random or* n*-random iff it passes all* Σ^0_n *tests.*
(iii) *One can similarly define* Π^0_n, Δ^0_n *etc tests and randomness.*
(iv) *A real* α *is called* arithmetically random *iff for any* n, α *is* n*-random.*

The reader might notice the subtle difference between the notion of Σ^0_1 test and Martin-Löf test, especially in the analogs used in Definition 8. The former is defined in terms of *classes* of *reals* and the latter in terms of c.e. *open sets* of *strings*. This is immaterial for $n = 1$ since every Σ^0_1 class C is equivalent to $\{[\sigma] : \sigma \in W\}$ for some c.e. set W. However, consider the case $n = 2$. Consider the Σ^0_2 class consisting of reals that are always zero from some point onwards. This Σ^0_2 class is *not* equivalent to one of the form $\{[\sigma] : \sigma \in W\}$ for some Σ^0_2 set W.

The use of open sets is basic in most arguments. Fortunately, this can be resurrected in the $n > 1$ cases also, as we now see.

Theorem 32 (Kurtz [66], Kautz [54]). *Let $q \in \mathbb{Q}$.*

(i) For S a Σ^0_n class, we can uniformly compute the index of a $\Sigma^{\emptyset^{(n-1)}}_1$ class which is also an open Σ^0_n class $U \supseteq S$ and $\mu(U) - \mu(S) < q$.

(ii) For T a Π^0_n class T, we can uniformly compute the index of a $\Pi^{\emptyset^{(n-1)}}_1$ class which is also a closed Π^0_n class $V \subseteq T$ and $\mu(T) - \mu(V) < q$.

(iii) For each Σ^0_n class S we can uniformly in $\emptyset^{(n)}$ compute a closed Π^0_{n-1} class $V \subseteq S$ such that $\mu(S) - \mu(V) < q$. Moreover, if $\mu(S)$ is a real computable from $\emptyset^{(n-1)}$ then the index for V can be found computably from $\emptyset^{(n-1)}$.

(iv) For a Π^0_n class T we can computably from $\emptyset^{(n)}$ obtain an open Σ^0_{n-1} class $U \supseteq T$ such that $\mu(U) - \mu(T) < q$. Moreover, if $\mu(S)$ is a real computable from $\emptyset^{(n-1)}$ then the index for U can be found computably from $\emptyset^{(n-1)}$.

Using the above, we can easily show that for instance, $n+1$-randomness coincides with 1-randomness relative to $\emptyset^{(n)}$, a theorem of Kurtz. Let U be the standard universal prefix-free machine. Then U^X will be a universal machine relative to any X. (The question of relativization in this setting can be a bit vexed, and is discussed in the next section.) We then get the following natural $n+1$-random sets.

$$\Omega^{(n)} =_{\text{def}} \sum_{U^{\emptyset^{(n)}}(\sigma)\downarrow} 2^{-|\sigma|}.$$

Becher and Santiago have looked at other natural examples of n-random sets using versions of Post's Theorem.

We return to our comment about Kolmogorov random reals. Recall that we defined a real to be Kolmogorov random iff its plain complexity hit $n - \mathbb{O}(1)$ infinitely often. Whilst we have seen in section 2.2 that no real can have its C-complexity always high, the set of Kolmogorov random reals has measure 1. The next theorem shows that *Kolmogorov randomness is characterized by 2-randomness.* Yu, Ding, and Downey [126] proved that each 3-random set is Kolmogorov random. They also observed that no Kolmogorov random set is in Δ^0_2. This is also implied by Theorem 33 since 2-random sets cannot be Δ^0_2.

Theorem 33 (Nies, Stephan, Terwijn [96]). *The following are equivalent for any set Z:*

(I) *Z is 2-random*
(II) *Z is Kolmogorov random*

The most surprising implication (I) $\Rightarrow$ (II) in Theorem 33 was proven independently and earlier by Joseph Miller [84].

Another characterization of 2-randomness can be given if we consider sets that are low for Ω:

Definition 9. *A is* low for Ω *if Ω is Martin-Löf random relative to A.*

Theorem 34 (Nies, Stephan, Terwijn [96]). *A set A is 2-random if and only if A is 1-random and low for Ω.*

Theorem 34 can also be obtained from material of van Lambalgen [67]. There is an apparently difficult open question here. By Chaitin's Counting Theorem, we know that the highest K-complexity a string can have is $|\sigma| + K(|\sigma|) - \mathbb{O}(1)$. We define a real α to be *strongly Chaitin random* iff

$$\exists^{\infty} n (K(\alpha \upharpoonright n) > n + K(n) - \mathbb{O}(1).)$$

It is known that if α is 3-random then it is strongly Chaitin random, and that every strongly Chaitin real is already 2-random. The following is open.

Question 8. Does strong Chaitin randomness coincide with Kolmogorov randomness= 2-randomness?

There is a lot of interesting and unpublished material here about n-randomness. For $n \geq 2$, n-random reals seem to be "more typically random". They all have the property that $A \oplus \emptyset^{(n-1)} \equiv_T A^{(n-1)}$, (in particular $A \oplus \emptyset' \equiv_T A'$) and hence we see that randomness is a *lowness property.*

The initial segment complexity of $n+1$-random reals and in particular $\Omega^{(n)} = \Omega^{\emptyset^{(n)}}$, the natural $n+1$-random real.

Theorem 35 (Yu, Ding, Downey [126]). *For all c and $n < m$,*

$$(\exists^{\infty} k)\big[K(\Omega^{(n)} \upharpoonright k) < K(\Omega^{(m)} \upharpoonright k) - c\big].$$

For $n = 0, m = 1$ Theorem 35 was proven by Solovay [112], using totally different methods.

More powerful results have recently been obtained by Miller and Yu. We need the following classical theorem of van Lambalgen. (The easy direction of the following was also proven by Kurtz.)

Theorem 36 (van Lambalgen [67]). *B n-random and A is B-n-random iff $A \oplus B$ is n-random.*

Motivated by this theorem, Miller and Yu suggested the following.

Definition 10 (Miller and Yu [86]). *We say that $\alpha \leq_{vL} \beta$, α is* van Lambalgen[6] *reducible to β if for all $x \in 2^{\omega}$, $\alpha \oplus x$ is random implies $\beta \oplus x$ is random.*

Theorem 37 (Miller and Yu [86]). *For all α, β,*

(i) α n-random and $\alpha \leq_{vL} \beta$ implies β is n-random.
(ii) If $\alpha \oplus \beta$ is random then α and β have no upper bound in the vL-degrees.
(iii) If $\alpha \leq_T \beta$ and α is 1-random, then $\beta \leq_{vL} \alpha$.
(iv) There are random $\alpha \equiv_{vL} \beta$ of different Turing degrees.

[6] This is closely related to a relation introduced by Nies: He defined $A \leq_{LR} B$ if for all Z, Z is 1-B-random implies Z is 1-A-random. If A and B are both random then $A \leq_{LR} B$ iff $B \leq_{LR} A$. Reals with $A \leq_{LR} \emptyset$ are Martin-Löf low. $\leq_{vL}$ is really only interesting on the randoms, but there are a number of possible extensions which also make sense on the non-randoms. These have yet to be explored.

(v) There are no maximal, minimal random vL-degrees, and no join.
(vi) If $\alpha \oplus \beta$ is random then $\alpha \oplus \beta <_{vL} \alpha, \beta$.
(vii) The Σ_1^0 theory of the vL-degrees is decidable.

Furthermore, Miller and Yu show that $\Omega^{(n)}$ and $\Omega^{(m)}$ have no upper bound in the *vL* degrees for $n \neq m$. One consequence of the above of the following very attractive result.

Corollary 1 (Miller and Yu [86]). *Suppose that the real α is n-random and that $\beta \leq_T \alpha$ is 1-random. Then β is n-random.*

All of this is filters through an interesting relationship between $\leq_{vL}$ and $\leq_C, \leq_K$.

Lemma 3 (Miller and Yu [86]). *For random α, β,*

(i) Suppose that $\alpha \leq_K \beta$. Then $\alpha \leq_{vL} \beta$.
(ii) Suppose that $\alpha \leq_C \beta$. Then $\alpha \leq_{vL} \beta$.

We state the following for $\leq_K$ but they hold equally for $\leq_C$.

Corollary 2 (Miller and Yu [86]).

(i) Suppose that $\alpha \leq_K \beta$, and α is n-random. Then β is n-random.
(ii) If $\alpha \oplus \beta$ is 1-random, then $\alpha|_K \beta$ and have no upper bound in the K-degrees.
(iii) For all $n \neq m$, the K-degrees of $\Omega^{(n)}$ and $\Omega^{(m)}$ have no upper bound.
(iv) If $\alpha \oplus \beta$ is 1-random, then $\alpha|_K \alpha \oplus \beta$.

Using new techniques and extensions of the above, Miller has proven the following.

Theorem 38 (Miller [85]).

(i) If α is 1-random and $\alpha \leq_{rK} \beta$, then $\beta \leq_{rK} \alpha$. As a consequence, $\alpha \equiv_T \beta$.
(ii) If α, β are random, and $\alpha \equiv_K \beta$, then $\alpha' \equiv_{tt} \beta'$. As a consequence, every K-degree of a random real is countable.
(iii) If $\alpha \leq_K \beta$, and α is 3-random, the $\beta \leq_T \alpha \oplus \emptyset'$. Furthermore, $\beta' \leq_{tt} \alpha'$.

Finally Miller has shown that if α is 3-random then its initial segment complexity is often very low indeed. We will call α *pseudo-low* if $(\exists^\infty n)[K(n) \leq K^\alpha(n) + O(1)]$. Thus in a pseudo-low real, the information in α is so useless that it cannot help to compress n. The following result echoes the theme articulated by Stephan that most random reals have little *usable* information in them.

Theorem 39 (Miller [85]).

(i) If α is 3-random it is pseudo-low.
(ii) If α is pseudo-low, then α is strongly Chaitin random in that

$$(\exists^\infty n)\big[K(\alpha \upharpoonright n) \geq n + K(n) - O(1)\big].$$

7 K-Antirandomness, Post's Problem, and Generalizing Kučera-Slaman

7.1 K-Trivial Reals

There have been really amazing developments relating $\leq_K$ and $\leq_T$.

We will call a real α *K-trivial* if $\alpha \leq_K \emptyset$. Solovay was the first to construct Δ_2^0 K-trivial reals. In [31], Downey, Hirschfeldt, Nies, and Stephan gave a new construction of a K-trivial real, and this time the real was strongly c.e. (Independently, this had been done by Kummer in an unpublished manuscript.) Earlier, Calude and Coles had indicated a construction of a K-trivial real. As we will later see this is a priority-free solution to Post's problem.

Theorem 40. (Downey, Hirschfeldt, Nies, and Stephan [31], Calude and Coles [11], Kummer (unpubl.) after Solovay [112]) *There is a noncomputable c.e. set A such that $K(A \upharpoonright n) \leq K(n) + O(1)$.*

The proof of Downey et. al. is relatively easy and looks like the Dekker deficiency set construction. One remarkable fact about the K-trivials is that for any witnessing constant there are *few* of them.

Theorem 41 (Zambella [127]). *For each d there are $O(2^d)$ many reals α such that for all n, $K(\alpha \upharpoonright n) \leq K(n) + d$.*

We remark that *K-trivial* reals are necessarily Turing incomplete, and not of high degree, and hence form a somewhat natural solution to Post's Problem.

Theorem 42 (Downey, Hirschfeldt, Nies, Stephan [31]). *If a real α is K-trivial then α is Turing incomplete. Indeed, α is not even high.*

The proof of Theorem 42, and its remarkable later improvements by Nies (which we discuss below) is tricky. It involves a kind of "pinball" or "decanter" construction, with what we call "quanta pushing." I think that there should be some easy proof. For example, by the work of Chaitin we know that every K-trivial real is Δ_2^0. Moreover, for a fixed d the K-trivial reals lie on a Δ_2^0 tree with exactly $O(2^d)$ many paths by Theorem 41. It seems that some oracle type basis result might be usable here to give an analysis of such trees.

Question 9. Is there a straightforward and easy proof that K -trivials are Turing incomplete?

We remark that Downey, Hirschfeldt, Nies, and Stephan [31] proved that the K-trivials are closed under *wtt*-reduction and +. The former will be improved to Turing reduction soon. As Nies remarks this gives the only "natural" Σ_3^0 ideal in the Turing degrees.

Note also that, since we have seen $\leq_C$ implies $\leq_T$ for c.e. reals, $\leq_K$ does not imply $\leq_C$.

Question 10. Does $\leq_C$ imply $\leq_K$ (for the c.e. reals)? We would suggest that the answer is no.

7.2 Lowness

We have already looked at relativized randomness. M. van Lambalgen and D. Zambella asked whether there exist noncomputable sets A such that every Σ_1-random set is already Σ_1-random relative to A. (The question was first stated in Zambella [127].) Note that every computable set is low for the Σ_1-random sets. The existence of nontrivial Martin-Löf low sets is given in the following.

Theorem 43 (Kučera and Terwijn [64]). *There exists a noncomputable c.e. set that is low for the Martin-Löf random sets.*

Downey, Hirschfeldt, Nies, and Stephan [31] noted that the construction of Theorem 43 bears much resemblance to their construction of a K-trivial set. The following definitive results shows that this is no coincidence.

Theorem 44 (Nies [92]). *A is low for the Martin-Löf random sets if and only if A is K-trivial.*

Notice that this means the class of K-trivials is closed under Turing reducibility, and earlier result of Nies and Hirschfeldt. Nies also shows that this class of K-antirandom sets consists of only *superlow* sets (that is $A' \equiv_{tt} \emptyset'$). The proofs are more intricate and nonuniform variations of the original Downey, Hirschfeldt, Nies, Stephan Theorem. Again we wonder if there might be easier proofs. Nies remarks that the proofs are in some sense intrinsically nonuniform. Nies, Stephan and Terwijn have shown that any c.e. set which keeps Ω random (i.e. low for Ω) is already K-trivial. The hypothesis that A be c.e. is essential since those authors also showed that each 2-random set is low for Ω, so the class of low for Ω sets has measure 1!

We mention last lowness results here. Andrei Muchnik (1998) worked on the class which as oracles do not help to reduce the prefix complexity of a string. He showed that $\mathcal{M}$ contains a non-computable set.

Definition 11. *A is* low for *K if* $\forall y\ K(y) \leq K^A(y) + \mathcal{O}(1)$.

Theorem 45 (Nies [92], Nies and Hirschfeldt [92]). *Each K-trivial real is low for K.*

The second says that K-antirandomness is somehow tied to computable enumerability.

Theorem 46 (Nies [92]). *For each K-trivial set A, there is a c.e. K-trivial set D such that $A \leq_{tt} D$.*

8 Schnorr, Computable, and Kurtz Randomness

Having proved that a real is Martin-Löf random iff its segments are K-incompressible iff no computably enumerable martingale succeeds on the real, Schnorr argued that the result demonstrated a clear *failure* of the intuition behind the notion of Martin-Löf randomness. He argued that randomness should be

concerned with defeating *computable* strategies or computably given tests rather than computably enumerable ones, since the latter are fundamentally asymmetric, in the same way that a computably enumerable set is semi-decidable rather than decidable.

Definition 12 (Schnorr randomness, Schnorr [104]).

(i) *We say that a Martin-Löf test* $\{V_n : n \in \mathbb{N}\}$ *is a* Schnorr test *if for all* n, $\mu(V_n) = 2^{-n}$. *In this case we call* $\bigcap_n V_n$ *a* Schnorr null set.
(ii) *We say that a real* α *is* Schnorr random *iff for all Schnorr tests,* $\alpha \notin \bigcap_n V_n$.

Definition 13 (computable randomness, Schnorr [104], Lutz [74]).

(i) *A martingale* f *is called computable iff* $f : 2^{<\omega} \mapsto \mathbb{R}^+ \cup \{0\}$ *is a computable function with* $f(\sigma)$ *(the index of functions representing the effective convergence of) a computable real. (That is, we will be given indices for a computable sequence of rationals* $\{q_i : i \in \mathbb{N}\}$ *so that* $f(\sigma) = \lim_s q_s$ *and* $|f(\sigma) - q_s| < 2^{-s}$.*)*
(ii) *A real* α *is called* computably random *iff no computable martingale succeeds on* α.

Schnorr observed that $\mathbb{R}^+ \cup \{0\}$ could be replaced by $\mathbb{Q}^+ \cup \{0\}$ in the definition of computable randomness, and 2^{-n} by any computable bound effectively shrinking to zero in the definition of Schnorr randomness.

The notion of computable randomness has inspired a number of notions in complexity theory since it naturally allows for time bounds. We refer the reader to Lutz [73,74], Ambos-Spies and Kučera [3], and Ambos-Spies and Mayordomo [4].

It is known that Martin-Löf random implies computably random implies Schnorr random. That there are Martin-Löf random sets that are not computably random was proved by Schnorr [104], and that there are computably random sets that are not Schnorr random was proved by Wang [122]. In some sense these concepts were not really explored much after the work of Schnorr, perhaps for various technical reasons. For instance, lacking was a machine definition of either of them. Perhaps part of the reason for this was because the Martin-Löf randomness was *enough* for many results. Another reason was that the notion is much less tractable than Martin-Löf randomness. For instance, Schnorr proved that *there is no universal Schnorr test.* The cornerstone of Martin-Löf's version of randomness is that the three characterizations (i) machine incompressibility (ii) tests and (iii) martingales, all coincide, modulo some criticisms; and hence the notion of 1-randomness is mathematically very robust.

Schnorr gave the following martingale characterization of Schnorr randomness. The crucial notion is that of an order.

Definition 14 (Schnorr [104]). *An* order *is a nondecreasing unbounded function* $h : \omega \to \omega$. *(N.B. An "Ordnungsfunktion" in Schnorr's terminology is always computable whereas we prefer to leave the complexity of orders unspecified in general.) For a martingale* d *and an order* h *we define*

$$S_h[d] := \{X : \limsup_{n\to\infty} \frac{d(X\upharpoonright n)}{h(n)} \geq 1\}.$$

Theorem 47 (Schnorr [104], Satz 9.4, 9.5). *$\mathcal{A} \subseteq 2^\omega$ is Schnorr null if and only if there are computable d and h such that $\mathcal{A} \subseteq S_h[d]$.*

It had been a longstanding open question to provide a machine characterization for Schnorr randomness. Terwijn [116] had made some progress in this area. Downey and Griffiths gave the following machine characterization of Schnorr randomness:

Definition 15 (Downey and Griffiths [22]). *A prefix free machine M is called* computable *iff*

$$\mu(\mathrm{dom}M) = \sum_{\sigma\in\mathrm{dom}(M)} 2^{-|\sigma|}$$

is a computable real.

Theorem 48 (Downey and Griffiths [22]). *A real α is Schnorr random iff for all computable machines M, there is a constant c such that, for all n, $K_M(\alpha\upharpoonright n) \geq n - c$.*

The Kučera-Slaman theorem shows that all random c.e. reals are wtt-complete, being Solovay complete. (Kučera [60] was the first to prove that they are all Turing complete.) There also is a characterization of the Turing degrees of Schnorr random reals:

Theorem 49 (Downey and Griffiths [22]). *Let A be a Schnorr random c.e. real. Then A has high Turing degree.*

Later we will see a strong converse to this result by Nies, Stephan and Terwijn. The machine characterization of Martin-Löf randomness allows us to calibrate randomness via $\leq_K$, and we know that a c.e. real α is random iff $\beta \leq_K \alpha$ for all c.e. reals β.

Inspired by this we can calibrate the complexity of c.e. reals in terms of their Schnorr complexity.

Definition 16 (Downey and Griffiths [22]). *We say that $\alpha \leq_{Sch} \beta$ iff for all* computable *machines M, there is a constant c and computable machine $\hat{M}$ such that for all n,*

$$K_{\hat{M}}(\alpha\upharpoonright n) \leq K_M(\beta\upharpoonright n) + c.$$

Recently, Downey, Griffiths, and LaForte [23] proved that Schnorr trivials are quite different from K-trivials.

Theorem 50 (Downey, Griffiths, and LaForte [23]).

(i) There exist Turing complete c.e. sets A that are Schnorr trivial.
(ii) No Schnorr trivial set is wtt-complete.
(iii) There exist nonzero c.e. degrees containing no Schnorr trivial reals.

Little is known about $\leq_{Sch}$.

Extending earlier work of Downey, Griffith and LaForte [23] and Wang [122] we have the following.

Theorem 51 (Nies, Stephan, and Terwijn [96]). *For every set A, the following are equivalent.*

(I) *A is high.*
(II) $\exists B \equiv_T A$, *B is computably random but not Martin-Löf random.*
(III) $\exists C \equiv_T A$, *C is Schnorr random but not computably random.*

Furthermore, if A is a c.e. real then B and C can also be chosen to be c.e. reals.

There is also an (ugly) measure theoretical characterization of computable randomness:

Definition 17 (Downey, Griffiths, and LaForte [23]). *A Martin-Löf test* $\{V_n\}$ *is* computably graded *if there exists a computable map* $f : 2^{<\omega} \times \omega \to \mathbb{R}$ *such that, for any* $n \in \omega$, $\sigma \in 2^{<\omega}$, *and any finite prefix-free set of strings* $\{\sigma_i\}_{i \leq I}$ *with* $\cup_{i=0}^{I} [\sigma_i] \subseteq [\tau]$, *the following conditions are satisfied:*

1. $\mu(V_n \cap [\sigma]) \leq f(\sigma, n)$
2. $\Sigma_{i=0}^{I} f(\sigma_i, n) \leq 2^{-n}$
3. $\Sigma_{i=0}^{I} f(\sigma_i, n) \leq f(\tau, n)$

A real x withstands a computably graded test iff $x \notin \bigcap_n V_n$. The computably graded tests give an alternative to the martingale characterization of computable randomness:

Theorem 52 (Downey, Griffiths and LaForte [23]). *A real* x *is computably random iff it withstands all computably graded tests.*

A similar characterization was also found by Merkle, Mihailovic, and Slaman [82].

Question 11. (i) Is there any *C*-characterization of either Schnorr or computable randomness?
(ii) Is there any characterization using *K*- alone?

There are other variations on the notion of 1-randomness. In [66], Stuart Kurtz introduced a new notion of randomness which looks at the idea from another perspective. Namely, instead of thinking of a real as random if it *avoids* all effectively given null tests, Kurtz suggested that a real should be considered as random if it *obeys* every effectively given test of measure 1.

Definition 18 (Kurtz [66]).

(i) A Kurtz (positive) test *is a c.e. open set U such that* $\mu(U) = 1$.
(ii) A real is called Kurtz random *(or weakly* 1*-random) if for all Kurtz tests* U, $\alpha \in U$.

Kurtz originally called this notion *weak randomness,* and it is a weak notion in that, as shown by Wang[7] [122], the notion is not *stochastic.* In relativized form, Kurtz 2-randomness means being in every Σ^0_2 class of measure 1, is *equivalent* to passing every "generalized" Martin-Löf test $\{U_n : n \in \mathbb{N}\}$, where now we only know that $\mu(U_n) \to 0$, but we do not have a computable upper bound on the measure. There is an equivalent formulation of Kurtz randomness in terms of a null test. We remark that some care is needed here since it is *not* true that Kurtz 2-randomness is equivalent to being in every $\Sigma^{0'}_1$ (open) class of measure 1. In relativized form what is true is that for $n \geq 2$, Kurtz n-randomness is equivalent to Kurtz $\Sigma^{0^{(n-2)}}_2$ randomness.

Some authors would argue that Kurtz n+1 randomness should be called *strong* Martin-Löf n-randomness, and that perhaps Kurtz (1-)randomness should not be regarded as a randomness notion at all; rather a genericity notion.

Definition 19 (Wang [122]). *A* Kurtz null test *is a collection* $\{V_n : n \in \mathbb{N}\}$ *of c.e. open sets, such that*

(i) $\mu(V_n) \leq 2^{-n}$, *and*
(ii) There is a computable function $f : \mathbb{N} \mapsto (2^{<\omega})^{<\omega}$ *such that* $f(n)$ *is a canonical index for a finite set of* σ*'s, say,* $\sigma_1, \ldots, \sigma_n$ *and* $V_n = \{[\sigma_1], \ldots, [\sigma_n]\}$.

Theorem 53 (Wang [122], after Kurtz [66]). *A real* α *is Kurtz random iff it passes all Kurtz null tests.*

There is a martingale definition of Kurtz randomness:

Theorem 54 (Wang [122]). *A real* α *is Kurtz random iff there is no computable martingale F and nondecreasing computable unbounded function* h*, such that for* almost all n*,*

$$F(\alpha \upharpoonright n) > h(n).$$

Thus Schnorr randomness implies Kurtz randomness. In spite of the fact that Jockusch proved that all Kurtz random reals are bi-immune, degree-theoretical they have nice behaviour. Extending an earlier theorem of Kurtz we have the following.

Theorem 55 (Downey, Griffiths, Reid [25]). *Let* $\mathbf{a} \neq \mathbf{0}$ *be a computably enumerable degree. Then* $\mathbf{a}$ *contains a Kurtz random computably enumerable real.*

Question 12. Characterize the Kurtz random degrees.

Downey, Griffiths, and Reid were able to give a nice machine characterization of Kurtz randomness in the style of Theorem 48.

[7] Comments in both Kurtz' and Kautz' theses indicate that this fact was probably known to them, but not stated.

Definition 20 (Downey, Griffiths, and Reid [25]). *A prefix-free machine M is called* computably layered *if there is a related computable function*

$$f : \omega \to (2^{<\omega})^{<\omega}$$

such that

- $\bigcup_i f(i) = \mathrm{dom} M$.
- *If* $\gamma \in f(i+1)$, *then* $(\exists \tau \in f(i))$, *such that* $M(\tau) \preceq M(\gamma)$.
- *If* $\gamma \in f(i)$, *then* $|M(\gamma)| = |\gamma| + i + 1$.

The idea of a computably layered machine is that each layer of the *domain,* $f(i)$, provides a layer of the *range,* and the range elements just become more refined as i increases.

Theorem 56 (Downey, Griffiths, and Reid [25]). *α is Kurtz random iff for all computably layered machines M,*

$$(\exists d)(\forall n)(K_M(\alpha \upharpoonright n) \geq n - d).$$

Interestingly, there is yet another machine characterization; this time in terms of computable machines.

Theorem 48 says that a real x is *Schnorr random* iff for all computable machines M, there is a constant c such that for all n, $K_M(x \upharpoonright n) \geq n - c$. Computable machines also characterize Kurtz randomness according to the following theorem:

Theorem 57 ([Downey, Griffiths, and Reid [25]). *α is* not *Kurtz random iff there is a computable machine M and a computable function $f : \omega \to \omega$ such that:*

$$(\forall d)(K_M(\alpha \upharpoonright f(d)) < f(d) - d).$$

There are other possible notions we could look at. For instance, we say that a real is *finitely random* if it avoids all Martin-Löf tests $\{U_n : n \in \mathbb{N}\}$ which are finite, and *computably finitely random* iff additionally there is a computable function g such that $|U_n| < g(n)$. These notions have been explored by Downey, Miller and Reimann [37]. They showed that, for instance, the Δ^0_2 reals which are finitely random are exactly the Martin-Löf random reals; something that fails in general. Also all non-low$_2$ c.e. degrees bound computably finitely random c.e. reals, which all have "array noncomputable degrees."

The point here is that one can come up with a number of meaningful notions of randomness which have interesting relationships with the degrees.

Question 13. Is there a notion of randomness capturing, say, any given double jump class?

Lowness questions are very interesting here. For a notion like Schnorr randomness there axe two possible lowness notions. The first is that X is low for *tests* meaning that for any X-Schnorr test $\{U^X_n : n \in \mathbb{N}\}$, there is a Schnorr test

$\{\widehat{U}_n : n \in \mathbb{N}\}$, such that $\cap_n \widehat{U}_n \supseteq \cap_n U_n^X$. The second notion is that the collection of Schnorr randoms relative to X is the same as the collection of Schnorr randoms.

Terwijn and Zambella [119] proved that there are uncountably many reals that are low for Schnorr tests and they live in the hyperimmune free degrees.

Definition 21 (Terwijn and Zambella [119]). *A set X is* computably traceable *if there is a computable function p (called a* bound, *such that, for all functions $g \leq_T X$, there is a computable function h where, for all n,*

(i) $|D_{h(n)}| < p(n)$,
(ii) $g(n) \in D_{h(n)}$.

Theorem 58 (Terwijn and Zambella [119]). *A degree* **a** *is low for Schnorr tests iff it is computably traceable.*

Here is the situation for the low-for-Schnorr-randomness notion.

Theorem 59. [Kjos-Hanssen and Nies [56], Bedregal and Nies [9]] *A degree* **a** *is low for Schnorr randoms iff it is low for Schnorr tests.*

Nies also answered the question for computable randomness.

Theorem 60 (Nies [93]). *A degree* **a** *is low for computably random reals iff it is computable.*

For other randomness notions the questions are still open. Downey, Griffiths and Reid [25] showed that if a degree **a** is low for Kurtz tests then it is hyperimmune-free and if the degree is computably traceable, it is low for Kurtz tests.

Question 14. (i) Suppose that **a** is low for Kurtz tests. Is **a** Schnorr low?
(ii) Is low for Kurtz tests the same as being low for Kurtz random?

Notice that such questions can be asked for many other randomness notions such as polynomial time randomness, etc.

Actually relativization here is a vexed issue since there is no oracle A where Kurtz 1-A-random implies Martin-Löf randomness. Joe Miller has suggested that the correct thing to study would be relativization only for Kurtz-n-randomness for $n \geq 2$.

9 Selection

Actually, there is a possible answer to Schnorr's criticisms of Martin-Löf randomness using computable predictions which has attracted a lot of interest recently.

Before we discuss this possible answer, we first return to von Mises vision of "selection rules." The basic idea is that a selection rule is one that given some bits of a real predicts another bit. A *selection rule* is a partial function $r : 2^{<\omega} \mapsto \{yes, no\}$. The subsequence of a sequence A selected by a selection

rule r is that with $r(A \upharpoonright n) = yes$. The sequence of selected places are those i such that $r(A \upharpoonright i) = yes$. Then for a given selection rule r and a given real A, we generate a sequence $n_0, n_1, \ldots$ of selected places, and we say that a real is *stochastic* with respect to admissible selection rules iff for any such selection rule, either the sequence of selected places is finite, or

$$\lim_{n\to\infty} \frac{\{i \leq n : A(n_i) = 1\}}{n} = \frac{1}{2}.$$

We say that a real is *von-Mises- Church- Wald stochastic* iff it is stochastic for all partial computable selection rules.

Of course, if α is Martin-Löf random it is easily seen to be von-Mises-Church-Wald stochastic. The converse is well-known to fail (von Lambalgen [67]). Recently Merkle showed that being stochastic has some consequences.

Theorem 61 (Merkle [79]). *Suppose that α is von-Mises-Church-Wald stochastic. Then for all c there exist infinitely many n such that*

$$C(\alpha \upharpoonright n) > c \log n.$$

We remark that if only computable selections are allowed then the resulting stochasticity notion is called Church randomness since it follows a suggestion of Church [19]. Evidently, every computably random real is Church stochastic.

A strengthening of the notion of computable randomness is to additionally consider partial computable martingales. As best I can determine, these were first considered by Ambos-Spies and Yongge Wang. (see Ambos-Spies [2], just before the bibliography.) They are also discussed in Terwijn's Thesis [116] were they are attributed to Fortnow, Freivalds, Gasarch, Kummer, Kurtz, Smith, and Stephan [43].

Theorem 62 (Merkle [79]). *Let $\mathcal{F}$ be the collection of all computable functions $f : \mathbb{N} \mapsto \mathbb{N}$ that are nondecreasing and unbounded. Then*[8]

(i) There is a computably random real α such that for all $f \in \mathcal{F}$, and almost all n,

$$C(\alpha \upharpoonright n|n) \leq f(n).$$

(ii) There is a partially computably random real β such that for all $f \in \mathcal{F}$, and almost all n,

$$C(\beta \upharpoonright n|n) \leq f(n) \log n.$$

It follows that there is a computably random set with $C(\alpha \upharpoonright n) < 2 \log n + c$, and hence we can obtain the following result of Ambos-Spies.

Corollary 3 (Ambos-Spies [2]). *There is a computably random real that is not von-Mises-Church-Wald stochastic.*

[8] Actually, Merkle proved these for monotone complexity rather than C, which is stronger.

It is also known from the thesis of von Lambalgen that there are von-Mises-Church-Wald stochastic reals that are not computably random. It is known that computable randomness imples computable stochasticity where now in the definition of von-Mises-Church-Wald stochasticity we only allow computable selections. Ambos-Spies, Mayorodomo, Wang and Zheng [5] have shown that stochasticity can be viewed as a kind of randomness for *restricted* kind of martingales.

Now we will change the rules and begin to consider nonmonotonic *selection.* We are given a real α. Now we axe allowed to select the places in some nonmonotonic fashion. The result is another notion of stochasticity called Kolmogorov-Loveland stochasticity. Similarly there is a martingale-betting game associated with this notion of selection. To wit, the betting game will select a sequence $n_0, n_1, \ldots$ of distinct places where we are allowed to bet on whether the place will have value 1. The place n_{i+1} depends solely upon the previous scanned bits $\alpha(n_0), \alpha(n_1) \ldots \alpha(n_i)$. Thus the game is defined by *pairs* (r_i, b_i) where the game receives as information $(r_0, b_0) \ldots (r_i, b_i)$ about the places r_j and bets b_j, than outputs (r_{i+1}, b_{i+1}). r_{i+1} is the next position to bet on, and b_{i+1} is the portion of the current capital we will bet. (More details can be found in Muchnik, Semenov, Uspensky [83] where this notion was first articulated, and in Merkle et. al. [80].)

The crucial point is that unlike a martingale we are not constrained to bet on position 1, then 2, then 3 etc. Here we can bet *nonmonotonically.* We call a real *Kolmogorov-Loveland*[9] *random* if no nonmonotonic computable betting strategy succeeds on it.

Question 15. **(An. A. Muchnik, Semenov, Uspensky [83])** Is Kolmogorov-Loveland randomness the same as Martin-Löf randomness?

We note that the answer is known for *stochasticity.*

Theorem 63 (Shen [107]). *There is a Kolmogorov-Loveland stochastic real that is not Martin-Löf random.*

If the answer to Question 15 is yes then this is a strong counter argument to Schnorr's in that it is a computable notion of randomness which has the same strength as Martin-Löf randomness. The question seems hard. Merkle et. al. have made the best progress so far on this question. They have proven that the answer is "close to" yes, at least in terms of Hausdorff dimension and density.

Theorem 64 (Merkle et. al. [80]). *For any Kolmogorov-Loveland-random sequence R and any $\delta > 0$ there is a (monotonic) computable subsequence S of R of density at least $1 - \delta$ that is 1-random.*

[9] We remind the reader of the earlier footnote regarding naming of terms. Perhaps this notion should be Muchnik-Semenov-Uspensky randomness or more simply nonmonotonic randomness.

Theorem 65 (Merkle et. al. [80]).

(i) Under any computable splitting, one of the halves of a Kolmogorov-Loveland-random real is 1-random.
(ii) If the real is Δ^0_2, then both halves are 1-random

Merkle et. al. also show that this result does *not* imply that the real is 1-random.

Theorem 66 (Merkle et. al. [80]). *There is a non-random (not even computably random) real such that under any computable splitting both halves are 1-random.*

Theorem 67 (Merkle et. al. [80]). *All Kolmogorov-Loveland-stochastic reals α have Σ^0_1 Hausdorff dimension 1. That is,*

$$\liminf_n \frac{K(\alpha \upharpoonright n)}{n} \to 1.$$

Theorem 67 strengthens an earlier result or Muchnik [83] that a sequence cannot be Kolmogorov-Loveland-stochastic if there is some $\alpha < 1$ such that we can effectively compute infinitely many n such that the length n prefix of the sequence can be compressed (according to C or K) to a string of length $\leq \alpha n$.

We remark that there is also a lot of interest in the realtionship between these ideas and partial computable randomness. Recently Nies and Stephan have proven that partial computable random reals are *not* closed under computable permutations of $\mathbb{N}$. Miller and Nies call a real permutation random iff any computaable permutation of them remains partial computably random. The question is, are they distinct from Kolmogorov-Loveland random reals?

Mostly these concepts remain to be explored as do the higher analogs and their degrees. Progress seems hard.

10 Ω^A and Post's Problem

Relativization is a strange business in the setting of, say, universal prefix-free machines. The reason for this is that we are dealing with c.e. operators, but not CEA (computably enumerable in *and above)* operators. Indeed, if A is not GL_1, then no A-random set is Turing above A (Kučera [62]). In particular, whatever Ω^Ω is, $\Omega \not\leq_T \Omega^\Omega$.

One hope that we had was that perhaps for some choice of universal machine, Ω would be a degree-invariant operator. Being degree-invariant would entail $A \equiv_T B$ implies $\Omega^A \equiv_T \Omega^B$. Should this be possible, Ω would be a degree invariant operator which was not a jump or iterate of a jump; thereby resolving the longstanding conjecture of Martin.

As we will see this program to disprove Martin's conjecture fails dismally. However, there is a lot of very interesting information we can prove about Ω as an operator.

Clearly any reasonable definition must have that Ω^X is a X-c.e. real which is X-random. Also we want it to be relatively machine independent. Thus we want that a universal machine U for which U^X is universal (i.e. minimal and prefix-free) *and has coding locations for all machines: that is, for any prefix-free* U_e^X, *there is a string* σ_e *such that* $U_e^X(\nu) = U^X(\sigma_e\nu)$, for all oracles X. *Henceforth this will always be what we mean by "universal".*

For example, we recall the theorem of Kučera and Slaman that if α is 1–random and c.e. , then α is Solovay complete (Theorem 11). When combined with Theorem 10 we conclude that the only 1–random c.e. reals are versions of Ω.

The following theorem shows that in spite of the fact that the original proof does not seem to relativize, there is a way of doing so.

Theorem 68 (Downey, Hirschfeldt, Miller, Nies [24]). *Suppose that* α *is* X*-c.e. and* X*-Solovay complete. Then there is a universal* U *with* $\alpha = \Omega_U^X$.

As we mentioned earlier, in Theorem 16, the range of the Ω-operator is wide indeed:

Theorem 69 (Downey, Hirschfeldt, Miller, Nies [24]). *Suppose that* α *is 2-random. Then there is a universal* U *and an oracle* X *such that* $\alpha = \Omega_U^X$.

Of course not all random reals are Ω^X for some X. Consider the real $\alpha = 1 - \Omega$.

Question 16. Suppose that $\mathbf{a} > \mathbf{0}'$. Can $\mathbf{a}$ contain a real α which is Ω^X for some X?

How far Ω fails to be degree invariant is revealed by the following theorem.

Theorem 70 (Downey, Hirschfeldt, Miller, Nies [24]). *For all universal* U, *there exist* $A =^* B$ *such that* $\Omega^A|_T\Omega^B$.

There are many other interesting results from [24], but in the interests of space, I will only mention a few more.

Theorem 71 (Downey, Hirschfeldt, Miller, Nies [24]). *Suppose that* A *is* Δ_2^0. *Then* A *is K-trivial iff* Ω^A *is a c.e. real.*

A consequence of this result is that any K-trivial real is a d.c.e. real.

The final results look at Ω as an operator acting on 2^ω.

Theorem 72 (Downey, Hirschfeldt, Miller, Nies [24]). *The following are equivalent.*

(i) A *is 1-generic*
(ii) Ω_M^A *is continuous at* A *for every prefix-free oracle machine* M.
(iii) Ω_U^A *is continuous on* A *for some universal* U.

As a corollary if $\Omega^A = \sup(\mathrm{ra}\ (\Omega_U^A))$, then A is 1-generic. (Here ra and sup denote the range and supremum of the relevant operators and sets.) Finally we have the following.

Theorem 73 (Downey, Hirschfeldt, Miller, Nies [24]). *If* $X \in \overline{\mathrm{ra}\ (\Omega_U)} - \mathrm{ra}\ (\Omega_U)$ *for some universal* U, *then* X *is 2-random.*

11 Hausdorff Dimension and Partial Randomness

There has been a lot of very nice work on efectivizing dimension beginning with Schnorr, and more recently by Ambos-Spies and his group in Heidelberg, and Lutz' group in Iowa. Suppose that $\Omega = .a_1a_2\ldots$. Then we would expect that $\tilde{\Omega} = a_10a_20\ldots$ would be something like "$\frac{1}{2}$-random." We would like to make this more precise. A classical refinement of the notion of a measure zero set, is given by dimensions such as packing and Hausdorff dimensions. Recall from Definition 14 the null sets of the form $S_h[d]$. Schnorr also addressed null sets of the form $S_h[d]$ with $h(n) = 2^{\varepsilon n}$, $\varepsilon \in (0,1]$, of *exponential* order. Schnorr did not mention Hausdorff dimension, but one can arrive at that by looking at exponential orders.

The most well-cited approach is due to Lutz. Lutz constructivized Hausdorff dimension in [75,76], using what he called *s-gales*.

Definition 22. *Let $s \in [0,\infty)$. An* s-gale *is a function $d : 2^{<\omega} \to \mathbb{R}^+$ that satisfies the averaging condition*

$$2^s d(w) = d(w0) + d(w1) \tag{3}$$

for every $w \in 2^{<\omega}$. Similarly, d is an s-supergale *if (3) holds with $\geq$ instead of equality. The success set $S[d]$ is defined exactly as was done for martingales in section 2.3.*

The idea behind an s-gale is that inflation is at work. In a martingale, nothing bad happens if you decide not to favour $\sigma 0$ or $\sigma 1$ with your bet at σ. However, in the case of an s-gale, not betting means that you will definitely lose money. Though the following is really a Theorem of Lutz, we will take it as our definition of effective Hausdorff dimension.

Definition 23. *For a complexity class $\mathcal{C}$, $\mathcal{A}$ has* $\mathcal{C}$-dimension α *if*

$$\alpha = \inf\{s \in \mathbb{Q} : \exists d \in \mathcal{C}\ (\ d \text{ supermartingale and } \mathcal{A} \subseteq S_{2^{(1-s)n}}[d]\ )\}.$$

For Σ_1-dimension, $\dim(\beta)$, we have the following characterization.

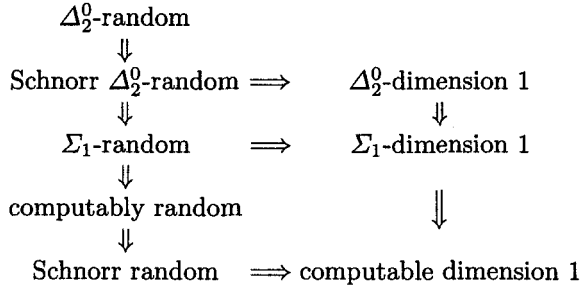
Theorem 74 (Lutz [75,76]). *For a real β,*

$$dim\ (\beta) = \liminf_n \frac{I(\beta \upharpoonright n)}{n},$$

for I either C or K.

Thus, a real has Σ_1-dimension 1 iff it is somehow a smoothed version of a 1-random real. For instance, if you take a very sparse set within a 1-random or even dimension 1 real, then you can code at will in that sparse set. Thus degrees containing dimension 1 sets are closed upwards. Note that this will also mean that there are dimension 1 degrees with no random sets.

The relations between the various notions are as follows:



No other implications hold than the ones indicated.

At this stage I would like to remark that using complexity theory, Lutz has shown that it is possible to give a meaningful dimension to an individual *string*! It is possible to show that our "stretched" Ω, $\tilde{\Omega}$, has dimension $\frac{1}{2}$. In fact, Tadaki [120] used an **s-variation** of measure on prefix free V:

$$\mu_s(V) = \sum_{\sigma \in V} 2^{-s|\sigma|},$$

to allow him to define **s-Martin-Löf** randomness, and show that $\tilde{\Omega}$ is $\frac{1}{2}$-**random.** There is a difficulty concerning the triad or definitions for Martin-Löf randomness when **s-Martin-Löf** randomness is used in place of randomness. The problem is that if you put $[\sigma 0]$ into some normal test set (a c.e. open set) and then later want to put the whole cone $[\sigma]$ in, then we only need to put $[\sigma 1]$ into the test. However, when you use **s-tests,** this does not work. These considerations lead to a new notion of test which allows for the characterization of **s-Marrtin-Löf** random. We refer the reader to Calude, Staiger and Terwijn [15].

This leads to a longstanding and fundamental question.

Question 17. Suppose a real β has positive Σ_1-**dimension.** Does it T-bound a 1-random real?

Other notions of effective Hausdorff dimension are less well studied. Terwijn in his thesis had a lot of nice material. One case study is on Schnorr and computable dimensions. Recently, Downey and Reimann [38] proved that Schnorr dimension has a machine characterization in terms of computable machines, akin to the characterization of Schnorr randomness of Downey and Griffiths.

Authors such as Lutz have also studied *packing dimension.* Without going through the classical definition, the characterization boils down to looking at, in the Σ_1-case, $\limsup_n \frac{I(\alpha \upharpoonright n)}{n}$. Thus limsup replaces liminf. This notion is mostly unexplored for other complexity classes, as is other dimensions such as box dimension and the like. Some of these are discussed in Jan Reimann's Thesis from Heidelberg.

Here is one example on computable packing dimension of computable enumerable *sets.* Computably enumerable sets are relatively simple in terms of their initial segment complexity. Barzdins' showed that they have complexity

$\leq 2\log n + \mathbb{O}(1)$, and Kummer proved the following lovely classification result. (The "array computable degrees" are a proper subclass of the $\mathbf{low_2}$ c.e. degrees.)

Theorem 75 (Kummer's Gap Theorem, Kummer [65]**).** *Exactly one of the following (i) or (ii) holds.*

(i) *A c.e. degree* $\mathbf{a}$ *contains a c.e. set A such that* $\exists^\infty n(C(A \upharpoonright n) \geq 2\log n - \mathbb{O}(1))$ *iff* $\mathbf{a}$ *is array noncomputable.*

(ii) *If A is c.e. and of array computable degree, then for every unbounded, nondecreasing, total computable function* f *there is a constant* c *such that*

$$C(A \upharpoonright n) \leq \log n + f(n) + c.$$

(iii) *Hence each c.e. degree* $\mathbf{a}$ *has the following gap phenomenon:*

(iiia) *Either, there is a c. e. set A in* $\mathbf{a}$ *such that* $C(A \upharpoonright n) \geq 2\log n - O(1)$ *for infinitely many* n.

(iiib) *Or there is* no *c.e. set* $A \in \mathbf{a}$ *and* $\epsilon > 0$ *such that* $(\exists^\infty n)[C(A \upharpoonright n) \geq (1+\epsilon)\log n - \mathcal{O}(1)]$.

Barzdins' upper bound fails dismally when you try to show its Schnorr dimension is something like $2\log n$. It is the totality of the domain of the prefix-free machine that causes problems.

Theorem 76 (Downey and Reimann [38]). *Suppose that* α *is a computable real* < 1. *Then there is a computably enumerable set whose computable (and hence Schnorr) packing dimension is above* α.

The point here is in some sense the *reason* that the Σ_1-dimension is of a c.e. set so small is because the measuring devise is simply too powerful. It is unknown if we can have a c.e. set of Schnorr packing dimension 1. It is unknown if Theorem 76 holds for Schnorr Hausdorff dimension. Merkle and Reimann have shown that it fails for computable Hausdorff dimension. Little else is known.

Degree theoretical aspects of, say, Σ_n dimension are not yet explored as are reducibilities like $\leq_C$ on them.

References

1. E. Allender, H. Buhrman, and Michal Kouck, *What Can be Efficiently Reduced to the K-Random Strings?, Proc. 21st International Symposium on Theoretical Aspects of Computer Science* (STACS), 2004. Lecture Notes in Computer Science 2996, pp. 584-595.
2. K. Ambos-Spies, *Algorithmic randomness revisited,* in *Language, Logic and Formailzation of Knowledge,* Bibliotheca, 1998, 33-52.
3. K. Ambos-Spies and A. Kučera, *Randomness in computability theory,* in: P. Cholak et al., Computability Theory: Current Trends and Open Problems, Contemporary Mathematics 257 (2000) 1–14, American Mathematical Society.
4. K. Ambos-Spies and E. Mayordomo, *Resource bounded measure and randomness,* in *Complexity, Logic and Recursion Theory,* (A. Sorbi, ed.) Marcel-Decker, New York, 1997, 1-48.

5. K.Ambos-Spies, E.Mayordomo, Y.Wang, and X. Zheng *Resource bounded balanced genericity, stochasticity and weak randomness,* In: Proc. 13rd STACS, pages 63-74. Lecture Notes in Computer Science 1046, 1996.
6. K. Ambos-Spies, W. Merkle, J. Reimann, and F. Stephan, *Hausdorff Dimension in Exponential Time,* Computational Complexity 2001, 210-217, IEEE Computer Society, 2001.
7. K. Ambos-Spies, K. Weihrauch and X. Zheng, *Weakly computable real numbers,* J. Complexity, Vol. 16 (4) (2000), 676-690.
8. J. Barzdins, *Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set,* Soviet Mathematics Doklady vol. 9 (1968), 1251-1254.
9. A.Ñies B. Bedregal. Lowness properties of reals and hyper-immunity. In *WOLLIC 2003,* volume 84 of *Electronic Notes in Theoretical Computer Science.* Elsevier, 2003. http://www.elsevier.nl/locate/entcs/volume84.html.
10. C. Calude, *Information and Randomness, An Algorithmic Perspective,* Springer, 1994. Second edition 2002.
11. C. Calude, and R. Coles, *Program size complexity of initial segments and domination relation reducibility,* in *Jewels are Forever,* (J. Karhümaki, H. Mauer, G. Paŭn, G. Rozenberg, eds.) Springer-Veralg, New York, 1999, 225-237.
12. C. Calude, R. Coles, P. Hertling, B. Khoussainov, *Degree-theoretic aspects of computably enumerable reals,* in *Models and Computability,* (ed. Cooper and Truss) Cambridge University Press, 1999.
13. C. Calude, P. H. Hertling, B. Khoussainov, and Y. Wang, *Recursively enumerable reals and Chaitin Ω numbers,* Proc. STACS 1998, Springer (1998) 596–606.
14. C. Calude and A. Nies, *Chaitin's Ω numbers and strong reducibilities,* Journal of Universal Computer Science, 1997.
15. C. Calude, L. Staiger, S. A. Terwijn, *On partial randomness,* in preparation.
16. G. Chaitin, *A theory of program size formally identical to information theory,* Journal of the ACM 22, (1975) 329–340.
17. G. Chaitin, *Algorithmic Information Theory,* Cambridge University Press, 1987.
18. A. Chernov, An. A. Muchnik, A. Romashchenko, A. Shen, N. Vereshchagin, *Upper semi-lattice of binary strings with the relation "x is simple conditional to y",* Theor. Comput. Sci. 271(1-2): 69-95 (2002)
19. A. Church, *On the concept of a random number,* Bulletin of the AMS, 46 (1940), 130–135.
20. R. Coles, P. Cholak, R. Downey and E. Herrmann, Automorphisms of the lattice of Π^1_0 classes: perfect thin classes and a.n.c. degrees, *Trans. Amer. Math. Soc.* Vol. 353 (2001), 4899-4924.
21. R. Downey, *Some computability-theoretical aspects of reals and randomness,* to appear, *Lecture Notes in Logic,* volume edited by Peter Cholak.
22. R. Downey and E. Griffiths, *On Schnorr randomness, Journal of Symbolic Logic.* Vol 69 (2) (2004), 533-554.
23. R. Downey, E. Griffiths, and G. LaForte, On Schnorr and computable randomness, martingales, and machines, to appear, *Mathematical Logic Quarterly.*
24. R. Downey, D. Hirschfeldt. J. Miller, and A. Nies, *On Ω^A,* in preparation.
25. R. Downey, E. Griffiths, and S. Reid, *On Kurtz randomness,* submitted.
26. R. Downey and D. Hirschfeldt, *Algorithmic Randomness and Complexity,* Springer-Verlag, to appear.

27. R. Downey, D. Hirschfeldt, and G. LaForte, *Randomness and reducibility,* to appear *Journal of Computing and System Sciences,* extended abstract appeared in *Mathematical Foundations of Computer Science, 2001* J. Sgall, A. Pultr, and P. Kolman (eds.), Mathematical Foundations of Computer Science 2001, Lecture Notes in Computer Science 2136 (Springer, 2001), 316–327.
28. R. Downey, D. Hirschfeldt, and G. LaForte, *Undecidability of Solovay and other degree structures for c.e. reals,* in preparation.
29. R. Downey, D. R. Hirschfeldt, S. Lempp, and R. Solomon, *A Δ^0_2 set with no infinite low subset in either it or its complement,* Journal of Symbolic Logic 66 (2001) 1371–1381.
30. R. Downey, D. Hirschfeldt, and A. Nies, *Randomness, computability and density* SIAM J. Comput. Vol. 31 (2002), 1169-1183. (Extended abstract appeared in *Symposium for Theoretical Aspects of Computer Science, STACS'01* January, 2001. Lecture Notes in Computer Science, Springer-Verlag, (Ed A. Ferriera and H. Reichel, 2001, 195-201)
31. R. Downey, D. R. Hirschfeldt, A. Nies, F. Stephan, *Trivial reals,* In Electronic Notes in Theoretical Computer Science (ENTCS), 2002. Final version appears in *Proceedings Asian Logic Conferences,* (edited Downey, et. al.) World Scientific, 2003.
32. R. Downey, D. R. Hirschfeldt, A. Nies, and S. Terwijn, *Calibrating randomness,* Bull. Symb Logic, to appear.
33. R. Downey, C. Jockusch, and M. Stob, *Array recursive sets and multiple permitting arguments,* in *Proceedings Oberwolfach 1989,* Springer Verlag, *Lecture Notes in Mathematics 1990,* 141-174.
34. R. Downey, C. Jockusch, and M. Stob, *Array nonrecursive degrees and genericity,* in *Computability, Enumerability, Unsolvability,* (ed. Cooper, Slaman, and Wainer), London Mathematical Society Lecture Notes Series Vol 224, Cambridge University Press (1996), 93-105.
35. R. Downey, and G. LaForte, *Presentations of computably enumerable reals,* Theoretical Computer Science, Vol. 284 (2002), 539-555.
36. R. Downey and J. Miller, *A basis theorem for Π^0_1 classes and jumps of random reals,* in preparation.
37. R. Downey, J. Miller, and J. Reimann, *Finite randomness,* in preparation.
38. R. Downey, and J. Reimann, *On Schnorr dimension,* in preparation.
39. R. Downey and S. A. Terwijn, *Computably enumerable reals and uniformly presentable ideals,* Mathematical Logic Quarterly 48(1) (2002) 29–40.
40. R. Downey and Yang Yue, Relative computability vs relative complexity, in preparation.
41. R. Downey, G. Wu and X. Zheng, *Degrees of d.c.e. reals,* to appear, Mathematical Logic Quarterly.
42. K. Falconer, *Fractal geometry, mathematical foundations & applications,* Wiley & Sons, 1992.
43. L. Fortnow, R. Freivalds, W. Gasarch, M. Kummer, S. Kurtz, C. Smith, and F. Stephan, *On the relative size of learnable sets,* Theoretical Computer Science, 197 (1998), 139-156.
44. P. Gács, *Every sequence is reducible to a random one,* Information and Control 70 (1986) 186-192.
45. H. Gaifmann and M. Snir, *Probabilities over rich languages,* J. Symb. Logic, Vol. 47, (1982), 495-548.
46. F. Hausdorff, *Dimension und äußeres Maß,* Mathematische Annalen 79 (1919) 157–179.

47. J. Hitchcock, J. Lutz and S. Terwijn, *The arithmetical complexity of dimension and randomness,* manuscript, January 2003.
48. Ho, Chun-Kuen, *Relatively recursive reals and real functions,* Theoretical Computer Science, Vol. 219 (1999), 99-120.
49. S. Ishmukhametov, *Weak recursive degrees and a problem of Spector,* in *Recursion Theory and Complexity,* (ed. M. Arslanov and S. Lempp), de Gruyter, (Berlin, 1999), 81-88.
50. T. Jech, *Set theory,* Academic Press, 1978.
51. C. Jockusch,
52. T. Kamae, *Subsequences of normal sequences,* Israel J. Math. 16 (1973) 121–149.
53. H. Katseff, *Complexity dips in random infinite binary sequences,* Information and Control 38 (1978) 258–263.
54. S. Kautz, *Degrees of Random Sets,* PhD thesis, Cornell University, 1991.
55. B. Khoussainov, *Randomness, computability, and algebraic specifications,* Annals of Pure and Applied Logic 91 (1998) 1–15.
56. B. Kjos-Hanssen, F. Stephan, and A. Nies. *On a question of Ambos-Spies and Kučera,* to appear.
57. K.-I Ko, *On the notion of infinite pseudorandom sequences,* Theoretical Computer Science 48 (1986) 9–33.
58. A. Kolmogorov, *Three Approaches to the Quantitative Definition of Information,* in *Problems of Information Transmission (Problemy Peredachi Informatsii),* 1965, Vol. 1, 1-7.
59. L. Kraft, *A device for quantizing, grouping, and coding amplitude modulated pulses,* M.Sc. thesis, MIT, 1949.
60. A. Kučera, *Measure, Π^0_1-classes and complete extensions of PA,* in: H.-D. Ebbinghaus, G. H. Müller, and G. E. Sacks (eds), Recursion theory week, Lect. Notes in Math. 1141, 245–259, Springer-Verlag, 1985.
61. A. Kučera, *On the use of diagonally nonrecursive functions,* in *Logic Colloquium '87,* Elsevier, 1987, 219-239.
62. A. Kučera, *On relative randomness,* Annals of Pure and Applied Logic 63 (1993) 61–67.
63. A. Kučera and T. A. Slaman, *Randomness and recursive enumerability,* SIAM J. Comput. 31 (2001) 199–211.
64. A. Kučera and S. A. Terwijn, *Lowness for the class of random sets,* Journal of Symbolic Logic 64(4) (1999) 1396–1402.
65. M. Kummer, *Kolmogorov complexity and instance complexity of recursively enumerable sets,* SIAM Journal of Computing, Vol. 25 (1996), 1123-1143.
66. S. Kurtz, *Randomness and Genericity in the Degrees of Unsolvability,* PhD thesis, University of Illinois, 1981.
67. M. van Lambalgen, *Random sequences,* PhD thesis, University of Amsterdam, 1987.
68. M. van Lambalgen, *The axiomatization of randomness,* Journal of Symbolic Logic 55 (3), 1990, 1143–1167.
69. J. Lathrop and J. Lutz, *Recursive computational depth,* Information and Computation 153 (1999) 139–172.
70. L. Levin. *On the notion of a random sequence,* Soviet Math. Dokl. 14 (1973) 1413–1416.
71. M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications,* Springer-Verlag, 1993.
72. D. Loveland, *A variant of the Kolmogorov concept of complexity,* Information and Control, vol. 15 (1969), 510-526.

73. J. Lutz, *Category and measure in complexity classes,* SIAM J. Comput. 19 (1990) 1100–1131.
74. J. Lutz, *Almost everywhere high nonuniform complexity,* J. Comput. System Sci. 44 (1992) 220–258.
75. J. Lutz, *Dimension in complexity classes,* Proc. 15th Conference on Computational Complexity, 158–169, IEEE Computer Society, 2000.
76. J. Lutz, *The dimensions of individual strings and sequences,* Information and Computation, to appear. Preliminary version: *Gales and the constructive dimension of individual sequences,* Proc. 27th International Colloquium on Automata, Languages, and Programming, 902–913, Springer, 2000.
77. P. Martin-Löf, *The definition of random sequences,* Information and Control 9 (1966) 602–619.
78. E. Mayordomo, *A Kolmogorov complexity characterization of constructive Hausdroff dimension.,* Information Processing Letters 84 (2002) 1–3.
79. W. Merkle, *The complesity of stochastic sequences,* in *Computational Complexity, 2003,* IEEE, 230-235.
80. W. Merkle, J. Miller, A. Nies, J. Reimann, and F. Stephan, *Nonmonotonic randomness,* in preparation.
81. W. Merkle and N. Mihailovic, *On the construction of effective random sets,* in Proceedings MFCS 2002, (ed. Diks and Rytter) Springer-Verlag LNCS 2420.
82. W. Merkle, N. Mihailovic, and T. Slaman, *Some results on effective randomness,* to appear.
83. An. A. Muchnik, A. Semenov, and V. Uspensky, *Mathematical metaphysics of randomness,* Theor. Comput. Sci., Vol 207(2), (1998), 263-317.
84. J. Miller, *Kolmogorov random reals are 2-random,* to appear.
85. J. Miller, *More on randomness,* in preparation.
86. J. Miller, and L. Yu, *On initial segment complexity and degrees of randomness,* in preparation. (This may turn into two papers)
87. W. Miller and D. A. Martin. The degree of hyperimmune sets. *Z. Math. Logik Grundlag. Math.,* 14:159–166, 1968.
88. R. von Mises, *Grundlagen der Wahrscheinlichkeitsrechnung,* Math. Z., vol. 5 (1919), 52-99.
89. Y. N. Moschovakis, *Descriptive set theory,* North-Holland, 1980.
90. Ng Keng Meng, MSc Thesis, National University of Singapore, in preparation.
91. A. Nies, *Reals which compute little,* to appear.
92. A. Nies, *Lowness properties and randomness,* to appear.
93. A. Nies, *Each Low(CR) set is computable,* typeset manuscript, January 2003.
94. A. Nies, *Low for random sets: the story,* preprint, available on www/cs.auckland.ac.nz/˜nies.
95. A. Nies, *Ideals in the recursively enumerable degrees,* in preparation.
96. A. Nies, F. Stephan and S. A. Terwijn, *Randomness, relativization, and Turing degrees,* in preparation.
97. P. Odifreddi, *Classical recursion theory,* Vol. 1, Studies in logic and the foundations of mathematics Vol. 125, North-Holland, 1989.
98. P. Odifreddi, *Classical recursion theory,* Vol. 2, Studies in logic and the foundations of mathematics Vol. 143, North-Holland, 1999.
99. J. Oxtoby, *Measure and Category,* 2nd edition, Springer 1980.
100. A. Raichev, Ph.D. thesis, University of Wisconsin-Madison, in progress.
101. J. Raisonnier, *A mathematical proof of S. Shelah's theorem on the measure problem and related results,* Israel J. Math. 48 (1984) 48–56.

102. H. Rogers Jr., *Theory of recursive functions and effective computability,* McGraw-Hill, 1967.
103. G. E. Sacks, *Degrees of unsolvability,* Annals of Mathematics Studies 55, Princeton University Press, 1963.
104. C. Schnorr, *Zufälligkeit und Wahrscheinlichkeit,* Lect. Notes in Math. 218, Springer-Verlag, 1971.
105. C. Shannon, *The mathematical theory of communication,* Bell System Tech. J. 27 (1948) 379–423, 623–656.
106. A. Kh. Shen'. *The frequency approach to the definition of the notion of a random sequence (in Russian), Semiotika i Informatika,* 18:14–41, 1982.
107. A. Kh. Shen'. *On relations between different algorithmic definitions of randomness, Soviet Mathematics Doklady,* 38, (1988), 316–319.
108. R. Soare, *Recursively enumerable sets and degrees,* Springer, 1987.
109. R. Soare, *Computability Theory and Differential Geometry,* bulletin Symbolic Logic, to appear.
110. R. J. Solomonoff, *A preliminary report on a general theory of inductive inference,* Tech. Report ZTB-138, Zator Company, Cambridge, Mass., November 1960.
111. R. Solovay, *A model of set theory in which every set of reals is Lebesgue measurable,* Annals of Mathematics 92 (1970) 1–56.
112. R. Solovay, *Draft of a paper (or series of papers) on Chattin's work,* unpublished manuscript, IBM Thomas J. Watson Research Center, New York, May 1975, 215 pp.
113. L. Staiger, *Constructive dimension equals Kolmogorov complexity,* Research Report CDMTCS-210, University of Auckland, January 2003.
114. F. Stephan, personal communication.
115. F. Stephan, *Martin-Löf random and PA-complete sets,* Forschungsbericht Mathematische Logik 58 / 2002, Universit" at Heidelberg, 2002.
116. S. Terwijn, *Computability and Measure,* PhD thesis, University of Amsterdam/ILLC, 1998.
117. S. Terwijn, *On the quantitative structure of* Δ^0_2, in: U. Berger, H. Osswald, and P. Schuster (eds.), Reuniting the Antipodes — Constructive and Nonstandard Views of the Continuum, Synthèse Library, Kluwer, 2000, 271–284.
118. S. Terwijn, *Complexity and Randomness,* Notes for a course given at the University of Auckland, March 2003. report CDMTCS-212, University of Auckland.
119. S. Terwijn and D. Zambella, *Computational randomness and lowness,* Journal of Symbolic Logic 66(3) (2001) 1199–1205.
120. K. Tadaki, *A generalization of Chaitin's halting probability* Ω *and halting self-similar sets,* Hokkaido Math. J., Vol. 32 (2002), 219-253.
121. J. Ville, *Étude critique de la notion de collectif,* Gauthier-Villars, 1939.
122. Y. Wang, *Randomness and complexity,* PhD thesis, University of Heidelberg, 1996.
123. Y.Wang, *A separation of two randomness concepts,* Information Processing Letters 69(3) (1999) 115–118.
124. Yu Liang and Ding Decheng, *There is no sw-complete c.e. real,* Journal of Symbolic Logic, to appear.
125. Yu Liang and Ding Decheng, *The initial segment complexity of random reals,* to appear, *Proc. Amer. math. Soc.*
126. Yu Liang, Ding Decheng, and R. Downey, *The Kolmogorov complexity of random reals,* to appear Annals of Pure and Applied Logic.
127. D. Zambella, *On sequences with simple initial segments,* ILLC technical report ML-1990-05, University of Amsterdam, 1990.

128. X. Zheng, *On the Turing degrees of weakly computable real numbers,* Journal of Logic and Computation **13** (2003), 159-172.
129. A. Zvonkin and L. Levin, *The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms,* Russian Math. Surveys 25(6) (1970) 83–124.

Ubiquitous Parameterization — Invitation to Fixed-Parameter Algorithms

Rolf Niedermeier*

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany
niedermr@informatik.uni-tuebingen.de

Abstract. Problem parameters are ubiquitous. In every area of computer science, we find all kinds of "special aspects" to the problems encountered. Hence, the study of parameterized complexity for computationally hard problems is proving highly fruitful. The purpose of this article is to stir the reader's interest in this field by providing a gentle introduction to the rewarding field of fixed-parameter algorithms.

Keywords: NP-hardness, parameterized complexity, fixed-parameter algorithms, parameterization.

1 Introduction

Computationally hard problems are ubiquitous. The systematic study of computational (in)tractability lies at the very heart of computer science. Garey and Johnson's [44] monograph on computational intractability surely is a landmark achievement in this direction, providing an in-depth treatment of the theory of NP-completeness. With the theory of NP-completeness and the like at hand, we can prove meaningful statements about the computational complexity of problems. But what happens after we have succeeded in proving that a problem is NP-hard (that is, "intractable") and, nevertheless, the problem has to be solved in practice? In other words, how does one cope with computational intractability? Several methods to deal with this problem have been developed [60]: approximation algorithms [12,57,86], average-case analysis [59], randomized algorithms [72], and heuristic methods [71,78]. All of them have their drawbacks, such as, respectively, the difficulty of approximation, lack of mathematical tools and results, limited power of the method itself, or the lack of any provable performance guarantees at all.

NP-hardness, in accordance with the state of the art of computational complexity theory, means that we have to take into account algorithms with exponential running times to solve the corresponding problems exactly.[1] Clearly,

* Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

[1] Clearly, whenever we obtain a fast enough optimal solution there is no need to search for approximate solutions. The point is to better understand to what extent we can

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 84–103, 2004.
© Springer-Verlag Berlin Heidelberg 2004

however, exponential growth quickly becomes prohibitive when running algorithms in practice. Fixed-parameter algorithmics provides guidance on the feasibility of the "exact algorithm approach" to hard problems by means of a refined, two-dimensional complexity analysis. The fundamental idea is to strive for better insight into a problem's complexity by exploring (various) problem-specific parameters and to find out how they influence the problem's computational complexity. Ideally, we aim for statements such as "if some parameter k is small in problem X, then X can be solved efficiently." For instance, in case of the NP-complete graph problem VERTEX COVER we know that if the solution set we are searching for is "small", then this set can be found efficiently whatever the graph looks like and however big it is—the exponential factor in the running time amounts to approximately 1.29^k [22,76], where the parameter k is the size of the solution set sought. Thus, if we search for little (size of solution), then we have to pay little (running time). Unfortunately, an analogous behavior is highly unlikely for the NP-complete graph problem DOMINATING SET, as the machinery of parameterized complexity theory makes clear [28].

Parameterized complexity and exact algorithms are the subject of several surveys [10,27,34,35,36,52,87], indicating the increasing importance of this field.[2] The purpose of this incomplete and personally biased expository paper is to stir the reader's interest in the field with an emphasis on algorithmic questions and methods. A distinguishing feature of this survey compared to previous ones will be its focus on the "art of parameterizing problems." Altogether, it will be shown that fixed-parameter complexity is not at all a "small area." In fact, we believe that it is destined to become a driving force in modern, algorithmically-oriented research on computationally hard problems.

2 Satisfiability or Why to Parameterize Problems?

The SATISFIABILITY problem for boolean formulas in conjunctive normal form may be considered the "drosophila of computational complexity theory." This fundamental NP-complete problem has been the subject of research on exact algorithms for decades [23] and it continues to play a central role in algorithmic research—the annual "SAT" conference is devoted to theory and applications of satisfiability testing. We define the problem as follows.

Input: A boolean formula F in conjunctive normal form.
Task: Determine whether or not there exists a truth assignment for the variables in F such that F evaluates to true.

afford to do so, that is, how costly the search for optimal solutions in various cases (parameterizations) is.

[2] Also see the December 2003 special issue of *Journal of Computer and System Sciences,* volume 67(4), on parameterized complexity and computation, edited by Jianer Chen and Michael R. Fellows. In addition, there is a newly launched conference *IWPEC 2004 (International Workshop an Parameterized and Exact Computation),* held in Bergen, Norway, September 2004 (proceedings appear in Springer's *LNCS* series).

Numerous applications such as VLSI design and model checking make SATISFIABILITY of real practical interest. Often, however, in concrete applications the corresponding problem instances turn out to be much easier than one might expect given the fact of SATISFIABILITY's NP-completeness. For instance, large SATISFIABILITY instances arising in the validation of automotive product configuration data turn out to be efficiently solvable [65,81]. Hence, the question arises of whether we can learn more about the complexity of SATISFIABILITY by means of studying various problem parameters. In particular, we are less interested in empirical results but more so in provable performance bounds related to various parameterizations of SATISFIABILITY. In this way, we hope to obtain a better understanding of problem properties that might be of algorithmic use. In the rest of this section, we discuss several parameterizations of SATISFIABILITY.

Formally, an input of SATISFIABILITY is a conjunction of m clauses where each clause consists of a disjunction of negated or non-negated boolean variables (so-called literals).

Parameter "clause size." The maximum number k of literals in any of the given clauses is a very natural formula parameter. For $k = 3$, however, the problem remains NP-complete whereas it is polynomial-time solvable for $k = 2$ [44]. Thus, for coping with the problem's intractability, this parameterization seems of little help in cases where $k \geq 3$.

Parameter "number of variables." The number n of different variables occurring in a formula significantly influences the complexity. Since there are 2^n different truth assignments, in essentially[3] this number of steps SATISFIABILITY can be solved. When restricting the maximum clause size by some k, for instance, $k = 3$, better bounds are known. The current best upper bound for 3-SATISFIABILITY is 1.324^n [62].

Parameter "number of clauses." If the number of clauses in a formula can be bounded by k, then SATISFIABILITY can be solved in 1.239^k steps [56].

Parameter "formula length." If the total length of the formula is bounded by k, then SATISFIABILITY can be solved in 1.074^k steps [56].

The above parameterizations will not suffice to explain all the cases of good behavior of SATISFIABILITY in many practical situations. There are application scenarios where none of them would lead to an efficient algorithm. Hence, the following way of parameterizing SATISFIABILITY seems prospective.

Parameters exploiting "formula structure." Szeider [84] surveys several recent exact algorithms with exponential bounds depending on certain *structural* formula parameters. He discusses parameters based on structural graph decompositions (where "variable interaction graphs" etc. are considered). Similar, more empirically oriented investigations have been started in [80].

Finally, from the parameterized complexity theory [28] point of view the following parameterization is of particular relevance.

[3] We neglect polynomial-time factors in the running time throughout the paper whenever they play a minor role in our considerations.

Parameter "weight of assignment." Here it is asked whether a formula has a satisfying assignment with exactly (!) k variables being set true. Although the parameterization with this "weight parameter" seems rather artificial, it plays a major role in characterizing parameterized intractability [28]. It serves as a building block in the parameterized hardness program similar to the role SATISFIABILITY plays in classical NP-completeness theory.

To summarize, different ways of parameterizing SATISFIABILITY lead to a better understanding of the problem's inherent complexity facets. In particular, it might be hoped that structural parameterizations as discussed and surveyed by Szeider [84] will yield new insights on tractable cases of SATISFIABILITY. But there surely is *no "best parameterization."* As a rule, the nature of the complexity of hard computational problems will probably almost always require such a multi-perspective view in order to win better insight into practically relevant, efficiently solvable special cases. To cope with intractability in a mathematically sound way, it seems that we have to pay the price by shifting our view from considering the input just through one pair of glasses (mostly these glasses are called "complexity measurement relative to input size") to a multitude of pairs of glasses, each of them with a different focus (that is, parameter). Note, however, that the above indicated, and in the rest of this paper further exhibited, methodology of parameterizing problems still relies on worst-case analysis. Parameterized complexity theory, as chiefly developed by Downey and Fellows [28], puts the related issues into a mathematically sound, formal framework.

3 Parameterized Complexity Theory in a Nutshell

Parameterized complexity theory [28] offers a two-dimensional framework for studying the computational complexity of problems. A *parameterized language* (problem) L is a subset $L \subseteq \Sigma^* \times \Sigma^*$ for some finite alphabet Σ. For $(x, k) \in L$, by convention, the second component denotes the *parameter.* The two dimensions of parameterized complexity analysis are constituted by the input size n, i.e., $n := |(x, k)|$ and the parameter value k (usually a nonnegative integer). A parameterized language is *fixed-parameter tractable* if it can be determined in $f(k) \cdot n^{O(1)}$ time whether $(x, k) \in L$, where f is a computable function only depending on k. The corresponding complexity class is called FPT.

A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction rules,* often yielding a *reduction to a problem kernel.* Here, the goal is, given any problem instance I with parameter k, to transform it into a new instance I' with parameter k' such that the size of I' is bounded by k', $k' \leq k$, and (I, k) has a solution iff (I', k') has a solution—see [1] for a recent thorough investigation of reduction to a problem kernel (also called *kernelization*) for the VERTEX COVER problem (the problem parameter there being the size of the vertex cover set).

Downey and Fellows developed a completeness program as a formal framework to show fixed-parameter intractability [28]. Let L, $L' \subseteq \Sigma^* \times \mathbb{N}$ be two

parameterized languages.[4] We say that L *reduces to* L' *by a standard parameterized m-reduction* if there are functions $k \mapsto k'$ and $k \mapsto k''$ from $\mathbf{N}$ to $\mathbf{N}$ and a function $(x, k) \mapsto x'$ from $\Sigma^* \times \mathbf{N}$ to Σ^* such that

1. $(x, k) \mapsto x'$ is computable in time $k''|x|^c$ for some constant c and
2. $(x, k) \in L$ iff $(x', k') \in L'$.

Notably, most reductions from classical complexity turn out *not* to be parameterized ones [28]. The basic complexity class for fixed-parameter intractability, W[1], can be defined as the class of parameterized languages that are equivalent to the SHORT TURING MACHINE ACCEPTANCE problem (also known as the k-STEP HALTING problem). Here, we want to determine, for an input consisting of a nondeterministic Turing machine M (with unbounded nondeterminism and alphabet size), and a string x, whether or not M has a computation path accepting x in at most k steps. This can trivially be solved in $O(n^{k+1})$ time (where n denotes a bound on the total input size) and we would be surprised if this can be significantly improved. Therefore, this is the parameterized analogue of the TURING MACHINE ACCEPTANCE problem that is the basic generic NP-complete problem in classical complexity theory, and the conjecture that $\mathrm{FPT} \neq \mathrm{W}[1]$ is very much analogous to the conjecture that $\mathrm{P} \neq \mathrm{NP}$. Other problems that are *W*[1]-*hard* (and also W[1]-*complete*) include the graph problems CLIQUE and INDEPENDENT SET, where the parameter is the size of the relevant vertex set [28]. Also, for constant maximum clause size, the aforementioned parameterization of SATISFIABILITY by the weight of an assignment (see Sect. 2) gives a W[1]-complete problem. W[1]-hardness gives a concrete indication that a parameterized problem with parameter k is unlikely to allow for a solving algorithm with $f(k) \cdot n^{O(1)}$ running time, that is, restricting the combinatorial explosion to the parameter seems illusory.

Since the parameter k represents some aspect(s) of the input or the solution, there usually are many meaningful ways to *parameterize* a problem. An important issue herein is whether a hard problem is fixed-parameter tractable with respect to a chosen parameter or not, and, in case of fixed-parameter tractability, how small the usually exponential growth of the function f can be kept. Hence, investigating different parameterizations gives insight into what causes the computational (in)tractability of a problem and in which qualitative and quantitative senses this happens and how it can be (potentially) coped with.

4 Fixed-Parameter Tractability—Three Case Studies

How can one design fixed-parameter algorithms? Three core concepts in this context are

- preprocessing by data reduction rules and problem kernels,
- bounded search trees, and
- dynamic programming.

[4] Generally, the second component (representing the parameter) is drawn from Σ^*; for most cases (particularly, within this paper), assuming the parameter to be a nonnegative integer (or a tuple of nonnegative integers) is sufficient.

Clearly, there would be more to say about further techniques for developing fixed-parameter tractability results but the above three issues definitely are among the most important ones. Other algorithmic strategies in the derivation of exact algorithms are discussed in the surveys [35,36,87]. To illustrate the above three techniques and, in this way, some central topics in fixed-parameter complexity analysis, we discuss three important, easy to grasp, graph problems.

4.1 Case Study Multicut in Trees

We start with an optimization problem on graphs that already is NP-complete when restricted to trees [45]—(the unweighted version of) MULTICUT in Trees.

> **Input:** An undirected tree $T = (V, E)$, $n := |V|$, and a collection H of m pairs of nodes in V, $H = \{(u_i, v_i) \mid u_i, v_i \in V, 1 \leq i \leq m\}$.
> **Task:** Find a minimum size subset E' of E such that the removal of the edges in E' separates each pair of nodes in H.

By trying all possibilities (and using $|E| = n - 1$) we can solve the problem in 2^{n-1} steps. But can we do better if there exists a small solution set E'? Let us define the parameter $k := |E'|$ and study how k influences the problem complexity. Note that clearly $k < n$ but in some problem cases $k \ll n$ seems to be a reasonable assumption. Here, the following simple algorithm works. Assume that the given tree is (arbitrarily) rooted. Consider a pair of nodes $(u, v) \in H$ such that the uniquely determined path p between u and v has maximum distance from the root r. Call w the node on p that is closest to r. Then, one may easily see (using the tree structure) that there is an optimal solution that contains at least one of the two path edges connected to w. But this implies that we can build a search tree of depth bounded by k: Simply search for w as specified above and then branch into the two cases (at least one of them has to yield an optimal solution) of taking one of the two neighboring path edges of w. Iterating this process, we obtain a search tree of size bounded by 2^k and,[5] thus, MULTICUT IN TREES is fixed-parameter tractable with respect to parameter k. There are also simple data reduction rules for MULTICUT IN TREES, but to prove a problem kernel as discussed in the following for VERTEX COVER is quite hard for MULTICUT IN TREES [55]. Hence, we prefer to switch to the simpler problem VERTEX COVER in graphs.[6]

4.2 Case Study Vertex Cover

VERTEX COVER has been one of the most important problems in fixed-parameter tractability studies:

[5] In a straightforward way, by taking both edges, one also obtains a polynomial-time factor-2 approximation algorithm. Nothing better is known [45].

[6] VERTEX COVER can be directly reduced to MULTICUT IN TREES restricted to star graphs (that is, trees of height one) [45]. Thus, approximation and exact algorithms for MULTICUT IN TREES easily transfer to VERTEX COVER.

Input: An undirected graph $G = (V, E)$.
Task: Find a minimum size subset V' of V such that each edge in E has at least one of its endpoints in V'.

By way of contrast to MULTICUT, VERTEX COVER is trivial when restricted to trees. Assuming an arbitrarily rooted tree, in a bottom-up fashion from the leaves to the root one can determine an optimal solution in linear time by a straightforward dynamic programming. For general graphs, defining the parameter $k := |V'|$, one easily obtains a size 2^k search tree—one of the two endpoints of an edge *has to* be part of a vertex cover. Opposite to MULTICUT IN TREES, however, the search tree size can easily be shrunk below 2^k. We describe a very simple possibility based on *"degree-branching"* here:

1. If there is a vertex of degree one, then put its neighbor into V'.
2. If there is a vertex v of degree two, then either put v into V' together with all neighbors of its neighbors or put both neighbors of v into V'.[7]
3. If there is a vertex v of degree three, then either put v or all its neighbors into V'.

It is not hard to see that the above search strategy always leads to an optimal solution. Moreover, in steps 2 and 3 the search branches into two cases each time. In step 2, each branch puts at least two vertices into V' and in step 3 the first branch puts one vertex into V' and the second branch puts at least three vertices into V'. This branching process is recursively repeated until an optimal solution is found. If the solution has size k, the corresponding search tree has size bounded by 1.47^k (which can be determined by solving the corresponding recurrences using standard mathematical tools; see, for example, [66]). Much more refined branching strategies (significantly increasing the number of case distinctions) led to search tree sizes smaller than 1.29^k [22,76].

Other than search tree strategies which are inherently exponential, *data reduction by preprocessing* is a polynomial-time methodology to shrink the input size as much as possible. Ideally, in some rare cases this might even solve the complete problem. But, if no solution is obtained solely by preprocessing (as the name already indicates), how do we measure the quality of the data reduction process besides purely empirical statements? To this end, parameterized complexity offers the concept of *problem kernels.* A simple preprocessing strategy for VERTEX COVER is based on the following observation. Since we are looking for a vertex set V' of size at most k, for a vertex v of degree greater than k we have no choice rather than putting v into V'. Otherwise, no V' with $|V'| \leq k$ can exist. In this way, we can get rid of all high-degree vertices. The remaining instance consists of vertices of maximum degree k. But then one vertex can cover at most k edges and, thus, there only can exist a vertex cover set of size at most k if the remaining instance has at most k^2 edges. In summary, either we can decide in this way that there is no solution or the solution has to be found

[7] Chen et al. [22] present a "folding trick" that makes it possible to avoid branching for degree-two vertices.

in a graph of size $O(k^2)$—the problem kernel. The decisive point is that the size of the problem kernel exclusively depends on the size of the parameter k.

There is one drawback in the just described preprocessing—it assumes that we know the value of k in advance. By way of contrast, the much more sophisticated data reduction due to Nemhauser and Trotter [73,64] does not require the value of k in advance and, furthermore, it provides a problem kernel graph of at most $2k$ vertices (see [22] for its application to a fixed-parameter algorithm for VERTEX COVER). Hence, after this polynomial-time preprocessing the exponential-time (search tree) algorithm can concentrate on a size-$2k$ graph. Further data reduction techniques such as "crown reduction rules" [63] (which can be interpreted as a generalization of the just described "degree-one rule") and their benefits in practice are discussed in a new study [1]. Observe that it is worth keeping an eye on the fact whether or not the problem kernelization rules make explicit use of the parameter value k. Clearly,"*parameter-independence"* of data reduction rules is preferable. So far, this distinction seemingly has been neglected in formal fixed-parameter complexity studies. Finally, we briefly remark that it is (provably) beneficial to employ kernelization techniques over and over again, for example, by interleaving search tree branching with data reduction [75].

4.3 Case Study Dominating Set

DOMINATING SET is a "parameterized antagonist" of VERTEX COVER. Both are minimization problems but DOMINATING SET appears to be intractable with respect to the parameterization "solution size k." This also reflects in approximation results. Whereas VERTEX COVER has a trivial factor-2 polynomial-time approximation algorithm, for DOMINATING SET we only have a factor-$\Theta(\log n)$ approximation [33]. DOMINATING SET is defined as follows.

> **Input:** An undirected graph $G = (V,E)$.
> **Task:** Find a minimum size subset V' of V such that for every vertex $v \in V$ either $v \in V'$ or there is at least one vertex in V' that is a neighbor of v, or both.

With respect to parameter $k := |V'|$, DOMINATING SET is W[2]-complete [28] (thus, W[1]-hard). For instance, the bounded search tree approach based on degree-branching (cf. Sect. 4.2) appears to fail: In analogy to VERTEX COVER, one might consider the neighborhood of a vertex v. To dominate v, either v or *one*[8] of its neighboring vertices has to be in the desired solution set. The point is that this yields a branching into a potentially *unbounded* number of cases. For bounded degree graphs with maximum vertex degree d this yields a search tree of size upper-bounded by d^k. However, there seems to be no way to overcome the dependence on d and to restrict the combinatorial explosion exclusively to k.

[8] Compare this with VERTEX COVER where we use that either v or *all* of its neighbors have to be in an optimal solution.

Hence, concerning the parameterization "solution size," it is reasonable to study special graph classes.

Besides bounded-degree graphs as indicated above, fixed-parameter tractability results are also known for planar graphs[9] and slight generalizations thereof. For instance, here also the bounded search tree works. But things are more complicated than might be expected. Although Euler's formula tells us that there always exists a degree at most five vertex in a planar graph, this does not yield a size 6^k search tree by the degree-branching method. The difficulty comes from the fact that we have to distinguish between three vertex states in the course of the search tree processing: a vertex can either be "dominating" (that is, it is in V') or "dominated" (that is, at least one of its neighbors is in V') or "non-dominated." Then, the difficulty is that we cannot delete dominated vertices because an optimal solution may require that such a vertex is still part of it. An already dominated vertex, however, is not a suitable candidate for the degree-branching technique because it is *not* true that either itself or one of its neighbors *has* to be in the dominating set. Thus, using the Euler formula one has to make sure that the selected vertex is a non-dominated one. With significant technical expenditure one can show that there always exists a non-dominated degree-seven vertex, directly giving a size 8^k search tree [6]. Notably, comparing the search tree algorithms for VERTEX COVER and DOMINATING SET (restricted to planar graphs), in the VERTEX COVER case one has a trivial branching (leading to a size 2^k search tree) and the algorithm becomes more and more complicated in order to shrink the size below 1.29^k. By way of contrast, in the DOMINATING SET case, the search tree itself is simple (and it is not known how to improve on that) whereas to prove its correctness (that is, the existence of a non-dominated degree-seven vertex during the course of the whole search tree algorithm) is difficult.

A completely different approach to solve DOMINATING SET on planar and related graphs is due to the concept of *tree decompositions*. Intuitively, a graph has a tree decomposition of small width if it is "tree-like"[10] and the smaller this so-called *treewidth* is, the better many otherwise hard graph problems can be solved (see [13,14] for surveys). Note that, similar to VERTEX COVER, DOMINATING SET is easily linear-time solvable on trees. By a much more complicated dynamic programming, this can be generalized to graphs of bounded treewidth [85]. Using the parameter treewidth t together with the corresponding tree decomposition of the underlying graph, one can show that DOMINATING SET can be solved in $O(4^t \cdot n)$ time on n-vertex graphs of treewidth t [4]. Thus, we have a completely different parameterization that yields fixed-parameter tractability for DOMINATING SET—the *structural parameterization* by treewidth. Note that, in principle, in this way the size of the solution set does not matter anymore.

Can the two parameterizations "solution size k" and "treewidth t" be related? In case of planar graphs, the answer is yes. That is, it was shown that for DOMINATING SET—analogous results hold for VERTEX COVER and many other

[9] A graph is planar if it can be drawn in the plane without edge crossings. A planar graph with n vertices can have at most $3n - 6$ edges (Euler formula).

[10] A tree has treewidth 1.

domination-like problems [4]—on planar graphs, assuming that k denotes the size of an optimal dominating set, it holds that

$$t = O(\sqrt{k}).$$

This implies that DOMINATING SET on planar graphs can be solved in $2^{O(\sqrt{k})} \cdot n$ steps [4]. (See [41,42] for improvements in the constants of the O-notation.) We mention in passing that bounding the exponential growth by a function as $2^{O(\sqrt{k})}$ is asymptotically the best what can be hoped for [18]. The involved constants of the worst-case analysis are fairly large. Recently, further fixed-parameter tractability results for DOMINATING SET and related problems were achieved for generalizations of planar graphs such as, for example, graphs of bounded genus [31,25]. Moreover, a general study of the relationship between treewidth and solution size has been undertaken, yielding interesting characterization results concerning the corresponding graph classes [24].

There is also a second route to $2^{O(\sqrt{k})} \cdot n$ time algorithms for DOMINATING SET and related problems on planar graphs. This is based on Lipton and Tarjan's planar separator theorem [68]. As noted and further discussed in [8,9], the existence of a linear-size problem kernel (as we already know for VERTEX COVER by the Nemhauser-Trotter theorem, see Sect. 4.2) directly implies $2^{O(\sqrt{k})} \cdot n$ time algorithms for VERTEX COVER and DOMINATING SET on planar graphs. And, indeed, a size-$O(k)$ problem kernel by polynomial-time preprocessing could also be proven for DOMINATING SET [7]. The corresponding data reduction rules, which are based on "local neighborhood considerations," are relatively simple and can be efficiently implemented. (The technically most demanding part is the proof of the linear-size bound on the problem kernel.) Besides the theoretical result itself, these parameter-independent (in the sense as discussed in Sect. 4.2) preprocessing rules turn out to be of high practical value [3]. Finally, we remark that the "planar separator theorem approach" seems to be less practical (due to the high constant factors involved) than the "bounded treewidth approach" discussed before.

In summary, given the W[2]-completeness of DOMINATING SET on general graphs, a reasonable way out of this quandary is to look at special graph classes and different parameterizations. A promising route of future investigations is to study graph classes where DOMINATING SET is polynomial-time solvable or fixed-parameter tractable and then to introduce useful parameterizations "away from" these classes [16,53]—one example for that is the above mentioned parameter genus of a graph that "parameterizes away from planarity." As also noted by Leizhen Cai [16], more possibilities are around. On a broader perspective, this leads to the "art of parameterization" that will be discussed next.

5 The Art of Parameterization

The direct way to parameterize an optimization problem is to take the size of the desired solution set as the parameter. This adheres to a "pay for what you

get approach"—the smaller the desired solution is, the smaller the combinatorial explosion is one has to take into account. It also turns out that this is *the* parameterization to study when considering preprocessing rules to yield a reduction to problem kernel. By way of contrast, in Sect. 4.3 we additionally considered the structural parameter treewidth (relating to the structure of the input graph, that is, more precisely, its "tree-likeness") that also allowed for efficient solutions in case of small width values.

As a concrete example for the fruitfulness of different parameterizations, recall MULTICUT IN TREES. In the previous section we considered the parameter "number of edges to delete," which again measures the solution size. An alternative parameterization is to consider the maximum number of paths passing through a node or an edge of the given tree. This measures an input property more or less independent of the solution size. Such a parameterization is of particular interest in case of the (edge-)weighted version of MULTICUT IN TREES: By means of dynamic programming, it can be shown that WEIGHTED MULTICUT IN TREES is fixed-parameter tractable with respect to the parameter "maximum path number" [54] whereas this is open concerning the parameterization "solution size."

5.1 Standard Ways of Parameterization

The above discussion shall indicate that parameterizing problems itself is an interesting and important game to play. For instance, Fellows [35] discusses how to parameterize the MAX LEAF SPANNING TREE problem in at least five different ways. Since the leitmotif of parameterized complexity theory is to gain a better understanding of problem hardness through a refined complexity analysis that uses a two-dimensional view on problems, the choice of various reasonable problem parameters deserves special attention. In what follows, we try to illustrate various facets of problem parameterization and argue that it may become a task of its own.

First, let us consider the NP-complete CLOSEST STRING problem.

Input: Strings $s_1, s_2, \ldots, s_k$ over alphabet Σ of length L each.
Task: Find a string s of length L that minimizes the maximum Hamming distance to the k input strings.

Two immediately arising and practically motivated parameterizations are by the "error measure" Hamming distance d (which should be minimized) and by the number of input strings k. Two completely different methods show fixed-parameter tractability with respect to both parameterizations—deep results of integer linear programming for parameter k and a search tree of size d^d for parameter d [51]. Here, parameterization by solution size as might be measured by the length of the solution string probably is of less interest. Still, however, for constant alphabet size simply enumerating all possible solution candidates leads to fixed-parameter tractability also with respect to the parameter L.

The situation changes when moving on to generalized (and for applications in computational biology more relevant) versions of CLOSEST STRING, namely

CLOSEST SUBSTRING and, further on, DISTINGUISHING SUBSTRING SELECTION. In CLOSEST SUBSTRING, the goal string now only needs to be a *substring* in every of the given strings, thus increasing the combinatorial complexity of the problem. And, indeed, with respect to the parameter k, CLOSEST SUBSTRING is W[1]-hard [37] and its parameterized complexity is open with respect to the distance parameter d. For the still more general DISTINGUISHING SUBSTRING SELECTION, W[1]-hardness can be shown for both parameterizations [48]. Notably, CLOSEST STRING, CLOSEST SUBSTRING, and DISTINGUISHING SUBSTRING SELECTION all possess polynomial-time approximation schemes [67,26]. The parameterized intractability of DISTINGUISHING SUBSTRING SELECTION with respect to "optimization parameter" d, however, indicates that the corresponding PTAS [26] cannot be turned into an efficient one (see [48] for more on this, [19] for the concept of efficient PTAS, and [27,34,36] for general discussions concerning the efficiency of PTAS's, connections with parameterized complexity, and some case studies). For both CLOSEST STRING and DISTINGUISHING SUBSTRING SELECTION, the only known parameterization that works (for constant-size alphabet) is with respect to solution string length (simple enumeration of all possibilities); but this has limited applicability in many cases of practical interest. New parameterizations and special cases of practical relevance are sought—the search for them can be guided by known parameterized complexity results as the above ones.

We mention in passing that similar parameterized complexity studies have been undertaken for the LONGEST COMMON SUBSEQUENCE problem—the parameterized complexity investigations mostly show hardness results [15,28,79].

Summarizing, the above examples should illustrate the great wealth of possibilities and the strong need for parameterizing problems. So far, we have seen (also refer to Sect. 4) parameters "solution size" (such as size of the dominating set), "solution quality" (such as distance parameter in CLOSEST STRING), "input partitioning" (such as number of input strings in CLOSEST STRING), or "input structure" (such as treewidth of graphs or maximum vertex degree in DOMINATING SET or alphabet size in CLOSEST STRING). In Sect 5.2, we will propose a further, rich set of possibilities to parameterize problems—parameterization by "distance from triviality."

5.2 New Ways of Parameterization

Leizhen Cai [16] recently initiated a study of GRAPH COLORING (where the task is to color the graph vertices such that all adjacent vertices have different colors) as follows. Note that GRAPH COLORING is already NP-complete for three colors, so the parameterization by number of colors is of no help.[11] For instance, considering the class of split graphs (where GRAPH COLORING is known to be solvable in polynomial time) he showed that coloring is fixed-parameter tractable with respect to parameter k on graphs that are formed from split graphs by

[11] Juedes et al. [63] show that coloring an n-vertex-graph with $n - k$ colors is fixed-parameter tractable with respect to k.

adding or deleting at most k edges. By way of contrast, he shows that it is W[1]-hard when deletion of at most k vertices leads to a split graph. Interestingly, the problem is much harder in case of (the 2-colorable) bipartite graphs instead of split graphs: Coloring becomes NP-complete for graphs that origin from bipartite graphs by adding three edges or if two vertex deletions are needed to make a graph bipartite [16]. In summary, Cai states that "this new way of parameterizing problems adds a new dimension to the applicability of parameterized complexity theory" [16]. The prospective idea brought forward here is that the distance parameter k measures a distance from a "triviality": $k = 0$ represents a special case that is solvable in polynomial time.

A second example for the parameterization "distance from triviality" is described by Hoffmann and Okamoto [58] when studying exact solutions for the TRAVELING SALESMAN problem in the two-dimensional Euclidean plane: Given a set of n points with pairwise Euclidean distances, determine a shortest round-trip through all points. Consider a set of n points in the Euclidean plane. Determine their convex hull. If all points lie on the hull, then this gives the shortest tour. Otherwise, Hoffmann and Okamoto show that the problem is solvable in $O(k! \cdot k \cdot n)$ time where k denotes the number of points inside the convex hull. Thus, the distance from triviality here is the number k of inner points.

Finally, we give an example related to SATISFIABILITY. Assume that a formula in conjunctive normal form has a matching between variables and clauses that matches all clauses. Then, it is easy to observe that such a formula is satisfiable. For a formula F, considered as a set of m clauses over n variables, define the *deficiency as* $\delta(F) := m - n$. The maximum deficiency is $\delta^*(F) := \max_{F' \subseteq F} \delta(F')$. Szeider showed that the satisfiability of a formula F can be decided in $O(2^{\delta^*(F)} \cdot n^3)$ time [83]. Note that a formula F with $\delta^*(F) = 0$ has a matching as described above. Again, $\delta^*(F)$ is a structural parameter measuring the distance from triviality in the described sense.

Further case studies for CLIQUE, POWER DOMINATING SET, SET COVER, and LONGEST COMMON SUBSEQUENCE exhibiting new distance from triviality parameterizations can be found in [53].

In conclusion, to parameterize a problem usually can be done in several useful ways—the discovery and treatment of new parameterizations may require new ideas and techniques.

A further parameterized view we neglected so far is the "parameterizing above guaranteed values" approach introduced by Mahajan and Raman [69]. Whereas Mahajan and Raman study parameterizations of the MAXIMUM SATISFIABILITY and MAXIMUM CUT problems, here let us only give a brief example concerning the INDEPENDENT SET problem—which is the dual problem of VERTEX COVER—on planar graphs. Due to the famous four-color theorem, one can conclude that every planar graph with n vertices has an independent set of size at least $\lceil n/4 \rceil$. Thus, the natural task arises to determine whether a planar graph has an independent set of size $\lceil n/4 \rceil + d$—the new parameter being d and the guaranteed value being $\lceil n/4 \rceil$. The parameterized complexity with respect to parameter d is open. Seen from a different view, this again

can be interpreted as distance from triviality; now, however, triviality means a guaranteed parameter value.

In some cases, moreover, it may be of interest to try to find relationships among different parameters of one problem—an example is given by the size of the dominating set in a planar graph and the treewidth of the underlying graph [4] (see Sect. 4). Clearly, in this way a whole *tableau of parameterized complexity questions* comes up for nearly every problem. Note, however, that often the different parameterizations of a problem are incomparable to each other and there is no point in discussing what the "best" parameterization is in general. The nature of complexity (and vice versa) simply seems to bring along the need for different views (that is, parameterizations) on the considered problem. Parameterized complexity provides the formal framework and guidance to do so.

6 Concluding Remarks

Parameterized complexity with fixed-parameter algorithmics is a fast growing field. This expository paper, oriented more towards algorithms than towards structural complexity theory, did not even touch several important topics and results of current research. Our approach to fixed-parameter algorithms is further pursued in a forthcoming monograph [74]. Besides the so far sole, more complexity-theoretic monograph by Downey and Fellows [28], we refer the reader to the numerous recent surveys [10,27,34,35,36,52,87] to further broaden the perspective on the field. Our main focus here laid on questions of parameterization itself.

Three brief messages we additionally want to communicate are as follows:

- Fixed-parameter algorithms are *old*—for instance, see Dreyfus and Wagner's [30] algorithm for the STEINER TREE problem in graphs.
- Fixed-parameter algorithms are *beautiful*—for instance, see Alon et al.'s [11] color coding technique to solve the LONGEST PATH problem.
- Fixed-parameter algorithms' future is *promising*—for instance, see [2,46] for two recent examples[12] of successful young researchers' work.

We conclude with five spotlights on other important topics in current parameterized complexity research.

Implementation and experiments. Conceptual simplicity is not simply a virtue of its own but it is of high importance when it comes to implementing and testing fixed-parameter algorithms. A search tree with numerous case distinctions probably will not pay off in practice when compared to a search tree algorithm with few case distinctions and slightly worse upper bounds on the size of the search tree. Moreover, preprocessing by data reduction rules seems almost obligatory in any practical implementation. Without experiments, it often will be hard to evaluate the concrete virtue of preprocessing,

[12] Selection was restricted to the Tübingen area.

though. In the context of applications, it is also fruitful to think in "dual terms." That is, for example, to solve the W[1]-complete CLIQUE problem one may use strongly tuned fixed-parameter algorithms for the dual problem VERTEX COVER—an n-vertex graph has a CLIQUE of size k iff its complement graph has a VERTEX COVER of size $n-k$. In some cases, preprocessing, kernelization and sophisticated branching combined with branch-and-bound heuristics yield unexpectedly successful implementations even in cases where parameter values are not "small." Concerning VERTEX COVER, recent implementation work is described in [1,20]. Several other results for various problems are described in [3,5,49,50,51]. To better understand and exploit the links between the design of heuristics and fixed-parameter algorithms, much remains to be done concerning implementation and experiments.

Automated algorithm development. Many fixed-parameter algorithms with small bounds on the parameter function $f(k)$ are based on intricate, extensive case distinctions. Recently, a framework for an automated generation of search tree algorithms was presented [47] that applies to many hard, not only graph (modification) problems. In some cases, even better upper bounds than achieved by algorithms designed by humans were achieved. Automated upper bound proofs for satisfiability problems can also be found in [32,77]. In this way, tedious and error-prone tasks (that is, design of lengthy case distinctions) can be mechanized, using some few basic concepts and observations on the underlying problem.

Connections to approximation algorithms. Fully polynomial-time approximation schemes (FPTAS's) as well as "efficient PTAS's" directly imply fixed-parameter tractability with respect to parameterizations by the "optimization value" [17,19]. By way of contrast, a proof of W[1]-hardness thus gives concrete indication for the non-existence of efficient PTAS's and FPTAS's, also see [27,36] for a more thorough discussion. In a broader sense, it is highly desirable (and it seems likely) to obtain further connections between approximation algorithm techniques together with inapproximability results and parameterized complexity.

Lower bounds. If a problem is fixed-parameter tractable measured by a function $f(k)$, the question arises how small the growth of f can be kept. For instance, Liming Cai and David Juedes [18] showed that under reasonable complexity-theoretic assumptions no $2^{o(\sqrt{k})} \cdot n^{O(1)}$ time algorithm for DOMINATING SET on planar graphs exists, asymptotically matching the upper bounds from [4]. Note that there are close connections to the so-called power indices of Stearns and Hunt III [82] and the existence of "subexponential" time algorithms as studied by Impagliazzo et al. [61]. Also see the very recent results of Chen et al. [21] for further studies in this direction. Frick and Grohe [43] showed that, unless P = NP, there is no algorithm for evaluating monadic second-order queries on trees in $f(k) \cdot n^{O(1)}$ time for any elementary function f, where k denotes the query size and n is the tree size. This strongly limits the practical usefulness of the corresponding fixed-parameter tractability results (also see [52] for a more thorough discussion). Finally,

Fernau [38] presents first lower bound results (concerning constant factors) for linear size problem kernels.

New parameterized complexity classes. We only point to three recent developments in this context. In an effort to get a deeper understanding of the border between fixed-parameter tractability and intractability, Downey et al. [29] introduced the class M[1] seemingly intermediate between FPT and W[1] (also see [35] for further discussion). Motivated by the above mentioned lower bounds results by Frick and Grohe [43], Flum et al. [40] very recently initiated the study of a notion called bounded fixed-parameter tractability together with corresponding complexity classes. Finally, the parameterized complexity of counting problems also became subject of recent research [39,70].

Acknowledgments. I am very grateful to Rod Downey, Michael Fellows, Jens Gramm, Jiong Guo, Falk Hüffner, Klaus-Jörn Lange, Catherine McCartin, Naomi Nishimura, and Prabhakar Ragde for their constructive feedback on a preliminary version of this paper.

References

1. F. N. Abu-Khzam, R. L. Collins, M. R. Fellows, M. A. Langston, W. H. Suters, and C. T. Symons. Kernelization algorithms for the Vertex Cover problem: theory and experiments. In *Proc. ALENEX04*. ACM/SIAM, 2004.
2. J. Alber. *Exact Algorithms for NP-hard Problems on Networks: Design, Analysis, and Implementation.* PhD thesis, WSI für Informatik, Universität Tübingen, Germany, 2003.
3. J. Alber, N. Betzler, and R. Niedermeier. Experiments on data reduction for optimal domination in networks. In *Proc. International Network Optimization Conference (INOC 2003),* pages 1–6, Evry/Paris, France, Oct. 2003.
4. J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for Dominating Set and related problems on planar graphs. *Algorithmica,* 33(4):461–493, 2002.
5. J. Alber, F. Dorn, and R. Niedermeier. Experimental evaluation of a tree decomposition based algorithm for Vertex Cover on planar graphs. *Discrete Applied Mathematics,* 2004. To appear.
6. J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, and U. Stege. Refined search tree technique for Dominating Set on planar graphs. In *Proc. 26th MFCS,* volume 2136 of *LNCS,* pages 111–122. Springer, 2001. Long version to appear in *Journal of Computer and System Sciences.*
7. J. Alber, M. R. Fellows, and R. Niedermeier. Efficient data reduction for Dominating Set: A linear problem kernel for the planar case. In *Proc. 8th SWAT,* volume 2368 of *LNCS,* pages 150–159. Springer, 2002. Long version to appear in *Journal of the ACM.*
8. J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. In *Proc. 28th ICALP,* volume 2076 of *LNCS,* pages 261–272. Springer, 2001. Long version to appear in *Journal of Algorithms.*

9. J. Alber, H. Fernau, and R. Niedermeier. Graph separators: A parameterized view. *Journal of Computer and System Sciences,* 67(4):808–832, 2003.
10. J. Alber, J. Gramm, and R. Niedermeier. Faster exact solutions for hard problems: a parameterized point of view. *Discrete Mathematics,* 229:3–27, 2001.
11. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM,* 42(4):844–856, 1995.
12. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation — Combinatorial Optimization Problems and Their Approximability Properties.* Springer, 1999.
13. H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proc. 22nd MFCS,* volume 1295 of *LNCS,* pages 19–36. Springer, 1997.
14. H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science,* 209:1–45, 1998.
15. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and H. T. Wareham. The parameterized complexity of sequence alignment and consensus. *Theoretical Computer Science,* 147(1-2):31–54, 1995.
16. L. Cai. Parameterized complexity of Vertex Colouring. *Discrete Applied Mathematics,* 127(1):415–429, 2003.
17. L. Cai and J. Chen. On fixed-parameter tractability and approximability of NP optimization problems. *Journal of Computer and System Sciences,* 54(3):465–474, 1997.
18. L. Cai and D. Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences,* 67(4):789–807, 2003.
19. M. Cesati and L. Trevisan. On the efficiency of polynomial time approximation schemes. *Information Processing Letters,* 64(4): 165–171, 1997.
20. J. Cheetham, F. Dehne, A. Rau-Chaplin, U. Stege, and P. J. Taillon. Solving large FPT problems on coarse-grained parallel machines. *Journal of Computer and System Sciences,* 67(4):691–706, 2003.
21. J. Chen, X. Huang, I. A. Kanj, and G. Xia. Strong computational lower bounds via parameterized complexity. Manuscript, a preliminary version appears in *36th ACM STOC 2004,* 2004.
22. J. Chen, I. A. Kanj, and W. Jia. Vertex Cover: Further observations and further improvements. *Journal of Algorithms,* 41:280–301, 2001.
23. E. Dantsin, E. A. Hirsch, S. Ivanov, and M. Vsemirnov. Algorithms for SAT and upper bounds on their complexity. Technical Report TR01-012, Electronic Colloquium on Computational Complexity, 2001.
24. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Bidimensional parameters and local treewidth. In *Proc. 21st LATIN,* volume 2976 of *LNCS,* pages 109–118. Springer, 2004.
25. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded-genus and H-minor-free graphs. In *Proc. 15th SODA,* pages 830–839. ACM/SIAM, 2004.
26. X. Deng, G. Li, Z. Li, B. Ma, and L. Wang. Genetic design of drugs without side-effects. *SIAM Journal on Computing,* 32(4):1073–1090, 2003.
27. R. G. Downey. Parameterized complexity for the skeptic. In *Proc. 18th IEEE Annual Conference on Computational Complexity,* pages 147–169, 2003.
28. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.
29. R. G. Downey, M. R. Fellows, E. Prieto-Rodriguez, and F. A. Rosamond. Fixed-parameter tractability and completeness V: parametric miniatures. Manuscript, 2003.

30. S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks,* 1:195–207, 1972.
31. J. Ellis, H. Fan, and M. R. Fellows. The Dominating Set problem is fixed parameter tractable for graphs of bounded genus. In *Proc. 8th SWAT,* volume 2368 of *LNCS,* pages 180–189. Springer, 2002.
32. S. S. Fedin and A. S. Kulikov. Automated proofs of upper bounds on the running time of splitting algorithms. Manuscript, September 2003.
33. U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM,* 45(4):634–652, 1998.
34. M. R. Fellows. Parameterized complexity: The main ideas and connections to practical computing. In *Experimental Algorithmics,* volume 2547 of *LNCS,* pages 51–77. Springer, 2002.
35. M. R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proc. 29th WG,* volume 2880 of *LNCS,* pages 1–12. Springer, 2003.
36. M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Proc. 8th WADS,* volume 2748 of *LNCS,* pages 505–520. Springer, 2003.
37. M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of Closest Substring and related problems. In *Proc. 19th STACS,* volume 2285 of *LNCS,* pages 262–273. Springer, 2002.
38. H. Fernau. Parametric duality: kernel sizes & algorithmics. Technical Report TR04-027, Electronic Colloquium on Computational Complexity, 2004.
39. J. Flum and M. Grohe. The parameterized complexity of counting problems. In *Proc. 43rd FOCS,* pages 538–550. IEEE Computer Society, 2002.
40. J. Flum, M. Grohe, and M. Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. In *Proc. 31st ICALP,* LNCS. Springer, 2004. To appear.
41. F. V. Fomin and D. M. Thilikos. Dominating sets in planar graphs: branch-width and exponential speed-up. In *Proc. 14th SODA,* pages 168–177. ACM/SIAM, 2003.
42. F. V. Fomin and D. M. Thilikos. A simple and fast approach for solving problems on planar graphs. In *Proc. 21st STACS,* volume 2996 of *LNCS,* pages 56–67. Springer, 2004.
43. M. Frick and M. Grohe. The complexity of first-order and monadic second-order logic revisited. In *Proc. Logic in Computer Science,* pages 215–224. IEEE Computer Society, 2002.
44. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.
45. N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica,* 18(1):3–20, 1997.
46. J. Gramm. *Fixed-Parameter Algorithms for the Consensus Analysis of Genomic Sequences.* PhD thesis, WSI für Informatik, Universität Tübingen, Germany, 2003.
47. J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica,* 39(4):321–347, 2004.
48. J. Gramm, J. Guo, and R. Niedermeier. On exact and approximation algorithms for Distinguishing Substring Selection. In *Proc. 14th FCT,* volume 2751 of *LNCS,* pages 261–272. Springer, 2003. Long version to appear in *Theory of Computing Systems.*
49. J. Gramm and R. Niedermeier. Breakpoint medians and breakpoint phylogenies: a fixed-parameter approach. *Bioinformatics,* 18(Supplement 2):S128–S139, 2002.
50. J. Gramm and R. Niedermeier. A fixed-parameter algorithm for Minimum Quartet Inconsistency. *Journal of Computer and System Sciences,* 67(4):723–741, 2003.

51. J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for Closest String and related problems. *Algorithmica,* 37(1):25–42, 2003.
52. M. Grohe. Parameterized complexity for the database theorist. *SIGMOD Record,* 31(4):86–96, 2002.
53. J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems: distance from triviality. Manuscript, April 2004.
54. J. Guo and R. Niedermeier. Exact algorithms for Tree-like Weighted Set Cover. Manuscript, April 2004.
55. J. Guo and R. Niedermeier. Fixed-parameter tractability results for Mulitcut in Trees. Manuscript, May 2004.
56. E. A. Hirsch. New worst-case upper bounds for SAT. *Journal of Automated Reasoning,* 24(4):397–420, 2000.
57. D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, 1997.
58. M. Hoffmann and Y. Okamoto. The traveling salesman problem with few inner points. In *Proc. 10th COCOON,* volume 3106 of *LNCS.* Springer, 2004. To appear.
59. M. Hofri. *Analysis of Algorithms: Computational Methods and Mathematical Tools.* Oxford University Press, 1995.
60. J. Hromkovič. *Algorithmics for Hard Problems.* Springer, 2002.
61. R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences,* 63(4):512–530, 2001.
62. K. Iwama and S. Tamaki. Improved upper bounds for 3-SAT. Technical Report TR03-053, Electronic Colloquium on Computational Complexity, 2003. Also appears in *Proc. ACM/SIAM SODA 2004.*
63. D. Juedes, B. Chor, and M. R. Fellows. Linear kernels in linear time, or how to save k colors in $o(n^2)$ steps. In *Proc. 30th WG,* LNCS. Springer, 2004. To appear.
64. S. Khuller. The Vertex Cover problem. *SIGACT News,* 33(2):31–33, 2002.
65. W. Küchlin and C. Sinz. Proving consistency assertions for automotive product data management. *Journal of Automated Reasoning,* 24(1–2):145–163, 2000.
66. O. Kullmann. New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science,* 223(1–2):1–72, July 1999.
67. M. Li, B. Ma, and L. Wang. On the Closest String and Substring problems. *Journal of the ACM,* 49(2):157–171, 2002.
68. R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing,* 9(3):615–627, 1980.
69. M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms,* 31(2):335–354, 1999.
70. C. McCartin. Parameterized counting problems. In *Proc. 27th MFCS,* volume 2420 of *LNCS,* pages 556–567. Springer, 2002.
71. Z. Michalewicz and B. F. Fogel. *How to Solve it: Modern Heuristics.* Springer, 2000.
72. R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
73. G. L. Nemhauser and J. L. E. Trotter. Vertex packing: structural properties and algorithms. *Mathematical Programming,* 8:232–248, 1975.
74. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2005. Forthcoming.
75. R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters,* 73:125–129, 2000.
76. R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for Weighted Vertex Cover. *Journal of Algorithms,* 47(2):63–77, 2003.

77. S. I. Nikolenko and A. V. Sirotkin. Worst-case upper bounds for SAT: automated proof. In *15th European Summer School in Logic Language and Information (ESSLLI 2003),* 2003.
78. J. Pearl. *Heuristics.* Addison–Wesley, Reading, Massachusetts, 1984.
79. K. Pietrzak. On the parameterized complexity of the fixed alphabet Shortest Common Supersequence and Longest Common Subsequence problems. *Journal of Computer and System Sciences,* 67(4):757–771, 2003.
80. C. Sinz. Visualizing the internal structure of SAT instances (preliminary report). In *Proc. 7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2004),* Vancouver, Canada, May 2004.
81. C. Sinz, A. Kaiser, and W. Küchlin. Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 17(1):75–97, 2003.
82. R. E. Stearns and H. B. Hunt III. Power indices and easier hard problems. *Mathematical Systems Theory,* 23:209–225, 1990.
83. S. Szeider. Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. In *Proc. 9th COCOON,* volume 2697 of *LNCS,* pages 548–558. Springer, 2003.
84. S. Szeider. On fixed-parameter tractable parameterizations of SAT. In *Proc. 6th SAT,* volume 2919 of *LNCS,* pages 188–202. Springer, 2004.
85. J. A. Telle and A. Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In *Proc. 3rd WADS,* volume 709 of *LNCS,* pages 610–621. Springer, 1993.
86. V. V. Vazirani. *Approximation Algorithms.* Springer, 2001.
87. G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Proc. 5th International Workshop on Combinatorial Optimization – Eureka, You Shrink!,* volume 2570 of *LNCS,* pages 185–208. Springer, 2003.

PRAM-On-Chip: A Quest for Not-So-Obvious Non-obviousness

Uzi Vishkin

University of Maryland Institute for Advanced Computer Studies and Electrical and Computer Engineering Department, College Park MD 20742-3251, USA,
vishkin@umd.edu,
http://www.umiacs.umd.edu/users/vishkin/XMT/

Abstract. Consider situations where once you were told about a new technical idea you reacted by saying: "but this is so obvious, I wonder how I missed it". I found out recently that the US patent law has a nice formal way of characterizing such a situation. The US patent law protects inventions that meet three requirements: utility, novelty and non-obviousness. Non-obviousness[1] is considered the most challenging of the three to establish.

The talk will try to argue that a possible virtue for a technical contribution is when, in restrospect, its non-obviousness is not too obvious; and since hindsight is always 20/20, one may often need to resort to various types of circumstantial evidence in order to establish non-obviousness. There are two reasons for bringing this issue up in my talk:

(i) seeking such a virtue has been an objective of my work over the years, and

(ii) issues of taste in research are more legitimate for invited talks; there might be merit in reminding younger researchers that not every "result" is necessarily also a "contribution"; perhaps the criterion of not-so-obvious non-obviousness could be helpful in some cases to help recognize a contribution.

The focus of the second focal point for my talk, the PRAM-On-Chip approach, meets at least one of the standard legal ways to support non-obviousness: "Expressions of disbelief by experts constitute strong evidence of non-obviousness". It is well documented that the whole PRAM algorithmic theory was considered "unrealistic" by numerous experts in the field, prior to the PRAM-On-Chip project. In fact, I needed recently to use this documentation in a reply to the U.S. patent office.

An introduction of the PRAM-On-Chip approach follows. Many parallel computer systems architectures have been proposed and built over the last several decades. The outreach of the few that survived has been severely limited due to their programmability problems. The question

[1] The legal definition of the non-obviousness requirement is as a follow: A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 104–105, 2004.
© Springer-Verlag Berlin Heidelberg 2004

of how to think algorithmically in parallel has been the fundamental problem for which these architectures did not have an adequate answer. A computational model, the Parallel Random Access Model (PRAM), has been developed by numerous (theoretical computer science) algorithm researchers to address this question during the 1980s and 1990s and is considered by many as the easiest known approach to parallel programming. Despite the broad interest the PRAM generated, it had not been possible to build parallel machines that adequately support it using multi-chip multiprocessors, the only multiprocessors that were buildable in the 1990s since low-overhead coordination was not possible. Our main insight is that this is becoming possible with the increasing amounts of hardware that can be placed on a single chip.

From the PRAM, as a starting point, a highly parallel explicit multi-threaded (XMT) on-chip processor architecture that relies on new low-overhead coordination mechanisms and whose performance objective is reducing single task completion time has been conceived and developed. Simulated program executions have shown dramatic performance gains over conventional processor architectures. Namely, in addition to the unique parallel programmability features, which set XMT apart from any other current approach, XMT also provides very competitive performance.

If XMT will meet expectations, its introduction would greatly enhance the normal rate of improvement of conventional processor architectures leading to new applications.

Theory and Applied Computing: Observations and Anecdotes

Matthew Brand, Sarah Frisken, Neal Lesh, Joe Marks, Daniel Nikovski, Ron Perry, and Jonathan Yedidia

Mitsubishi Electric Research Labs (MERL), Cambridge, MA 02139, USA
{brand,frisken,lesh,marks,nikovski,perry,yedidia}@merl.com

Abstract. While the kind of theoretical computer science being studied in academe is still highly relevant to systems-oriented research, it is less relevant to applications-oriented research. In applied computing, theoretical elements are used only when strictly relevant to the practical problem at hand. Theory is often combined judiciously with empiricism. And increasingly, theory is most useful when cross-pollinated with ideas and methods from other fields. We will illustrate these points by describing several recent projects at Mitsubishi Electric Research Labs that have heavy mathematical and algorithmic underpinnings. These projects include new algorithms for: traffic analysis; geometric layout; belief propagation in graphical models; dimensionality reduction; and shape representation. Practical applications of this work include elevator dispatch, stock cutting, error-correcting codes, data mining, and digital typography. In all cases theoretical concepts and results are used effectively to solve practical problems of commercial import.

1 Introduction

Many of the classical topics of theoretical computer science (e.g., algebra, finite automata, geometry, graph algorithms, logic, numerical methods, queuing theory, string processing) are still studied to good effect in the industrial labs of systems-oriented companies like IBM and Microsoft. However, as ubiquitous computing becomes a reality, many companies (e.g., Mitsubishi Electric, Philips, Siemens, and Sony) are focusing on *applied* computing. Computer-science theory has still to establish itself in applied computing. We argue that theory has a useful role in this context, but only when the following precepts are borne in mind:

- *Theory is a tool to understand and solve practical problems.* In other words, the nail should take precedence over the hammer: in applied computing the problem is paramount and the means of solution is secondary. This mindset leads to better, more eclectic problem selection and ultimately to more-relevant research.
- *Theory and empiricism complement each other.* Many real-world problems involve incomplete or uncertain data; many are NP-hard. Completeness, optimality, and asymptotic complexity are theoretical concepts that are rarely

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 106–118, 2004.
© Springer-Verlag Berlin Heidelberg 2004

useful on their own for such problems. However, in combination with experimentation and statistical analysis—the tools of empirical analysis—these theoretical notions can be very useful.

– *Theoretical computer science can be informed by insights from other fields.* Cognitive science, economics, electrical engineering, statistics, theoretical physics: concepts from these fields and others have proven useful for practical problems when used in tandem with computer-science theory.

We illustrate these points by describing several recent projects at Mitsubishi Electric Research Labs (www.merl.com) that have strong mathematical and algorithmic underpinnings. These projects include new algorithms for: traffic analysis; geometric layout; belief propagation in graphical models; dimensionality reduction; and shape representation. Practical applications of this work include vehicular-traffic prediction, elevator dispatch, stock cutting, error-correcting codes, image processing, data mining, and digital typography. In all cases theoretical concepts and results are being used in accord with the precepts above to solve practical problems of commercial import.

2 Traffic Analysis

Traffic of goods, vehicles, and passengers is a very complex phenomenon characterized by significant stochasticity, non-stationarity, incomplete observability, and huge problem sizes. Two transportation problems of large economic significance are optimal routing of vehicular traffic from origin to destination, and optimal elevator service for passengers in large buildings. Recent progress on both of these problems has resulted from a synergistic combination of theoretical and empirical concepts and methods.

Early successes in the field of vehicle routing were the result of important theoretical insights, most notably the formulation of the principle of optimality by the mathematician Richard Bellman, and the subsequent widespread use of dynamic programming [1]. In particular, efficient algorithms for finding shortest paths in static graphs have been available for a long time and now run on the relatively weak computers found in car-navigation systems. Current models offer spectacular performance, planning routes between two points in a whole country in less than one second, and navigation-system vendors are looking for novel and more advanced applications [2].

One such application is dynamic route guidance, or car navigation in dynamic stochastic networks. Nowadays, heavy congestion plagues the roads of most cities in the developed industrial world, and travel times can vary significantly depending on the time of day, week, year, etc. Finding optimal routes and optimal departure times under such conditions opens a number of novel problems such as sensing the state of the transportation network, predicting travel time on short-term and long-term horizons, and finding the shortest-routing policies in time-varying stochastic networks.

Predicting travel times from past observations is decidedly on the practical side of scientific research, and is currently an active area of investigation in industry. Recent results include a fast and efficient linear method for travel-time prediction with surprisingly good accuracy [3]. However, the very foundations of the traffic-prediction enterprise depend on answering some very theoretical questions that concern the limits of predictability in congested transportation networks. Nagel and Rasmussen have put forward the hypothesis that a transportation network would exhibit chaotic behavior when its load is pushed to its capacity, and hence its long-term prediction would be impossible [4]. An alternative and simpler hypothesis explains the high variance of travel times in heavily congested regimes from the point of view of queuing theory, without adverse implications to expected predictability. While finding the correct explanation is ultimately a highly theoretical question, its answer is likely to affect significantly all fielded systems.

Planning routes with dynamic stochastic travel times and scheduling elevators under dynamic stochastic passenger flows are two related problems that are also accompanied by partial observability. Modern frameworks such as decision-theoretic planning, considered theoretical and abstract only until quite recently, are slowly starting to bear fruit and find their way into practical applications. For example, Figure 1 shows a seven-floor building with four hall calls and one car call assigned to an elevator car. The uncertainty in passenger destinations can lead to an exponential number of possible car trajectories, illustrated here by a partial tree. However, dynamic programming can be employed to marginalize out this uncertainty in linear time [5]. Although explicit decision-theoretic methods have provable performance, they have yet to be embraced by industry: all current commercial elevator-scheduling systems use heuristic AI methods. To be accepted by engineers, new elevator-scheduling algorithms must prove themselves empirically, through thorough simulation and field tests.

3 Human-Guided Search

The Human-Guided Search (HuGS) project is an ongoing investigation into designing interactive human-in-the-loop optimization systems. This work illustrates the value of combining theory with empiricism, and of combining theory with techniques from other fields, in this case human-computer interaction (HCI). Interactive optimization produces more usable solutions than automatic optimization because users can steer an interactive algorithm based on their knowledge of real-world constraints. People are better able to trust, justify, and modify solutions when they help construct those solutions. Additionally, interactive optimization leverages people's skills in areas in which people currently outperform computers, such as visual perception, strategic thinking, and the ability to learn.

A major component in HuGS research is designing algorithms that are amenable to human guidance. We have developed a human-guidable version of tabu search for jobshop scheduling, edge-crossing minimization, the selective

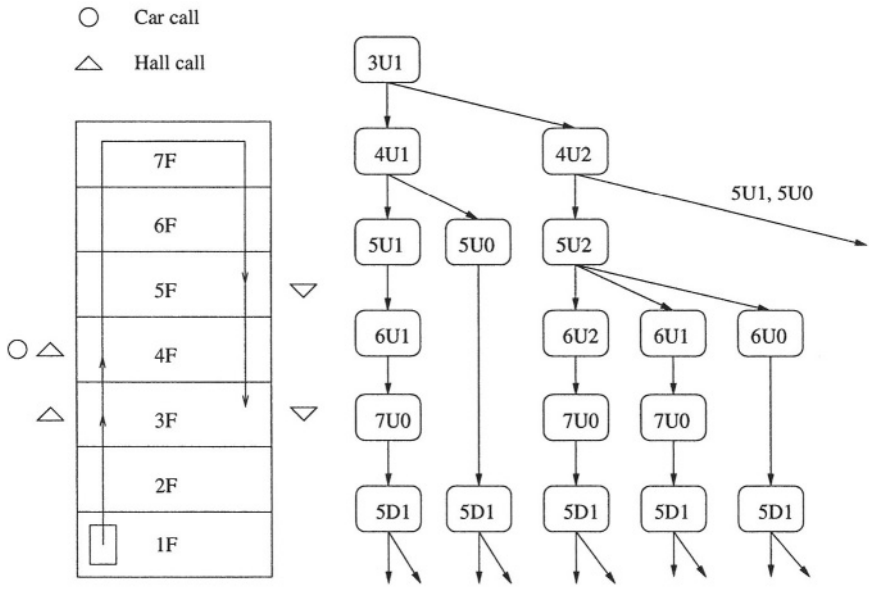


Fig. 1. An elevator car (one of several) in a seven-floor building can have an exponentially large number of possible trajectories when serving all calls assigned to it, due to the uncertainty in passengers' destinations. (Each node in the tree denotes the stopping floor, direction, and number of passengers inside the car.) However, the resulting probabilistic tree has sufficient structure that allows these trajectories to be folded into an efficient Markov model that can be evaluated in linear time.

traveling salesman problem, and simplified protein folding [6,7]. More recently, we have built a HuGS application for a 2D rectangular strip packing, which has many industrial applications, such as glass and steel cutting [8].

Our interactive packing application consists of several components. One component is an interface, shown in Figure 2, that is common to many of our applications. The interface allows the user to manually modify solutions, backtrack to previous solutions, and invoke, monitor, and halt a variety of optimization algorithms on the whole problem or a subset of the problem.

We provide two packing algorithms to the user. The first is a branch-and-bound algorithm for producing perfect packings, i.e., packings in which there is no unused space [9]. One can think of this special case as a jigsaw puzzle with rectangular pieces. We developed several powerful bounding methods that enable our algorithm to produce exceptionally good results on artificially constructed benchmarks in the literature that were designed to have solutions that are perfect packings. The real value of our algorithm, however, is as a subroutine that the user can invoke on a portion of the target packing area. Our algorithm fills as much of the user-defined region as it can with the user-selected rectangles without leaving any unused space between rectangles. Even though this algorithm cannot solve realistic problems by itself, it is a very useful tool or subroutine for realistic problems.

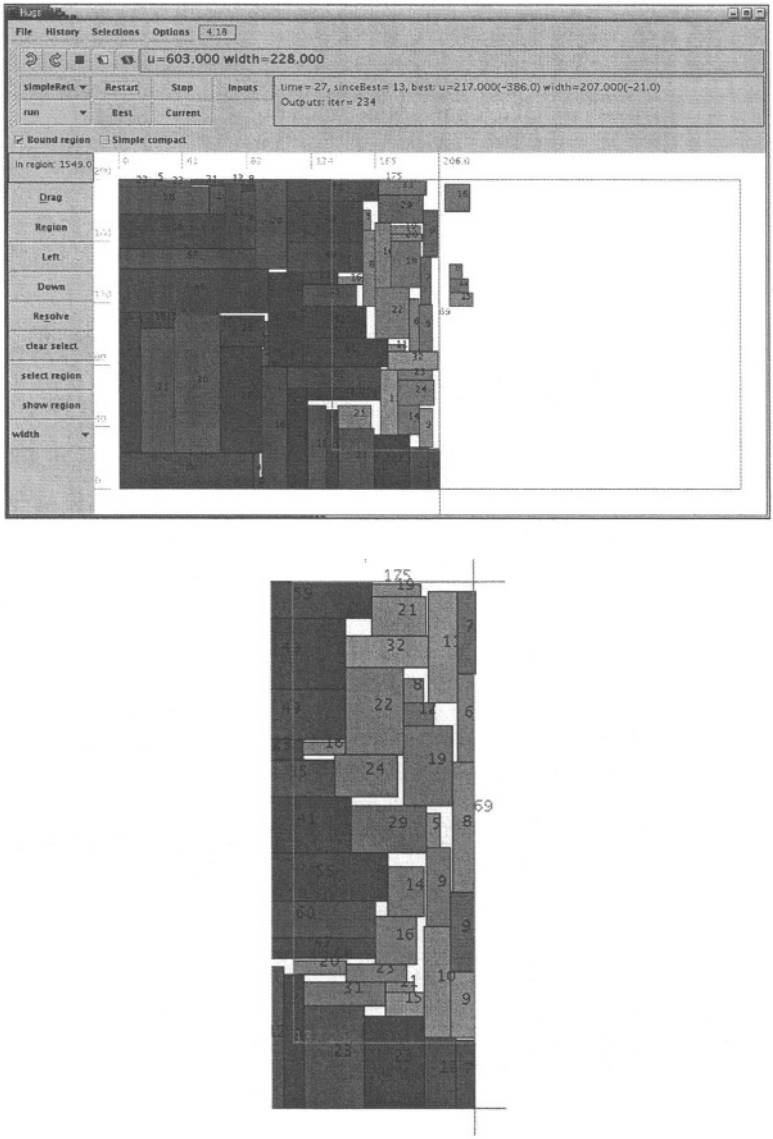


Fig. 2. Interactive system for an industrial cutting-stock problem, 2D Rectangular Strip Packing. In the first image, the user has selected a region to which an optimization algorithm can be applied and has "frozen" most of the already-packed rectangles (those shown in red) in their place. The second image shows a blowup of the selected portion of the packing after the optimization algorithm has run for a few seconds. By allowing the human user to focus the search on a small region and subset of the rectangles, a better packing for the problematic region is found quickly, thereby improving the overall solution. The combination of user interaction and automatic placement does better than either approach on its own.

The second algorithm we provide is an extension of a priority-based greedy heuristic that was shown to be a 3-approximation [10]. We produced an anytime algorithm that stochastically searches for solutions near the single solution produced by these heuristic. We often find substantially better solutions after only a small number of iterations. The results produced by this algorithm are better than previously published results on these benchmarks.

However, the most distinctive aspect of our system is the way in which it leverages our algorithmic novelties by incorporating the innate geometric-reasoning abilities of humans into the process. We have found that people can identify particularly well-packed subregions of solutions, and focus our algorithms on improving the other parts. Furthermore, people can readily envision multi-step repairs to a packing problem to reduce unused space. Our experiments on large benchmarks show that interactive use of our system can produce solutions 1% closer to optimal in about 15 minutes than our algorithms can produce automatically in 2 hours [8]. For typical industrial applications, this represents a commercially significant advantage.

4 Belief Propagation

The "belief propagation" (BP) algorithm is used to solve "inference" problems, at least approximately. Inference problems are important in many different scientific and industrial fields. Essentially any time you receive a noisy signal and need to *infer* what is really out there, you are dealing with an inference problem. Some fields that are dominated by the issue of inference are computer vision, speech recognition, and digital communications. In recent work at MERL, results from statistical mechanics have been combined with results from theoretical computer science to shed new light on the BP algorithm.

It is therefore perhaps not so surprising that a good algorithm to solve such problems has been repeatedly re-discovered in different scientific communities. In fact, one can show that such apparently different methods as the "forward-backward algorithm," the "Viterbi algorithm," Gallager's "probabilistic decoding" algorithm for low-density parity check codes, the "turbo-decoding" algorithm, the Kalman filter, Pearl's belief propagation algorithm for Bayesian networks, and the "transfer-matrix" approach in statistical physics are all special cases of the BP algorithm.

BP algorithms come in many flavors. The goal of the version that we consider here is to compute marginal probabilities for variables defined in a graphical model. These graphical models are referred to in various communities as Bayesian networks, Markov random fields, factor graphs, or statistical mechanical spin systems. Computing marginal probabilities thus corresponds to computing magnetizations for a spin system, or making a diagnosis in a Bayesian network, or computing a bit value for an error-correcting code.

In a BP algorithm, variable nodes in a graphical model iteratively send each other "messages" (see Figure 3). These messages are estimates that each variable node has of its own state, given what it is told by all of its neighboring nodes

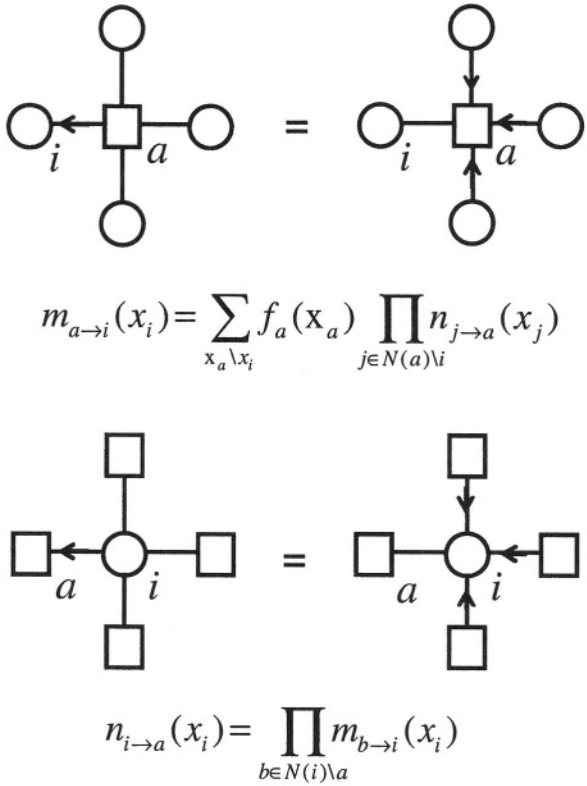


Fig. 3. This figure illustrates BP message-update rules, operating on a "factor graph." A factor graph has two kinds of nodes: variable nodes, indicated by a circle, and function nodes, indicated by a square. Variable nodes are connected to function nodes if they are arguments of that function. Messages are sent from variable nodes to function nodes and vice-versa according to a set of rules that can be derived by minimizing the Bethe free energy, a measure from the field of statistical mechanics.

except for the node to which it is sending a message. If and when the iterative algorithm converges, the desired marginal probabilities can be read off from the converged messages.

It is known that BP is exact when the graphical model has a tree-like topology; that is, when it has no cycles. However, the graphical models used in computer vision, or those for error-correcting codes, are infested with cycles, and BP still seems to give excellent approximate answers in these cases. The goal of the MERL project described here was to understand why BP worked so well even for cases when it seemed to have no justification [11, 12].

At MERL, we showed that the fixed points of the BP algorithm are the same as the stationary points of the "Bethe free energy," which is an approximate free energy for the graphical model. This deep connection between a classical approximation in statistical mechanics and a classical algorithm in computer science has important implications. For example, it means that by minimizing the Bethe free energy, one can invent algorithms that have the same fixed points as BP, but for which convergence is *guaranteed* [13,14]. Moreover, it means that one can *improve* upon the approximation made by BP, by improving upon the Bethe-free-energy approximation. We have developed a theory of such generalized belief propagation (GBP) algorithms [11,12].

Algorithms that minimize the Bethe free energy directly have been shown to improve upon standard BP decoding algorithms for state-of-the-art error-correcting codes, by eliminating the failure mode of lack of convergence [15]. GBP algorithms have given improved results over BP algorithms for such disparate problems as decoding error-correcting codes [16,17], and the computer vision problem of recovering shading and reflectance information from a single image [18]. GBP has inspired closely-related algorithms like "structured summary propagation," which has been applied with good results to the problem of synchronization using linear feedback shift registers [19]. Finally, GBP has been combined with fast Fourier transforms to give an exciting new algorithm for reconstructing missing data [20].

5 Dimensionality Reduction

Cheap sensing and storage devices have produced massive data streams, and new challenges to researchers in data mining, machine learning, and machine perception. Because data processing has not kept pace with data production, it is often necessary to reduce data sizes by "squeezing" out redundancies before any expensive processing begins. This challenge of dimensionality reduction is being met successfully by hybrid approaches that combine theory from many fields with empirical methods such as simulation and visualization.

Where data can be interpreted as points in a vector space with a Euclidean metric, squeezing out redundancy is usually synonymous with reducing the dimensionality of that space via subspace projection, e.g., principal components analysis. The orthogonal basis of this subspace is computed via singular-value or eigen-value decomposition. These decompositions are usually computed in quadratic time. We have developed a linear-time online approximation for the principal singular vectors and values of streaming data that is exact for data having true low rank and provably convergent to the optimal vectors when high-rank data arrives in a random order [21]. It is an enabling technology for compression and correlational analysis of massive data sets and streams. For example, correlated tastes between movie-goers in a 14-dimensional subspace of movie ratings are remarkably accurate predictors of how well 1000s of people will like 1000s of different movies. Since consumer tastes are non-stationary and not sampled at

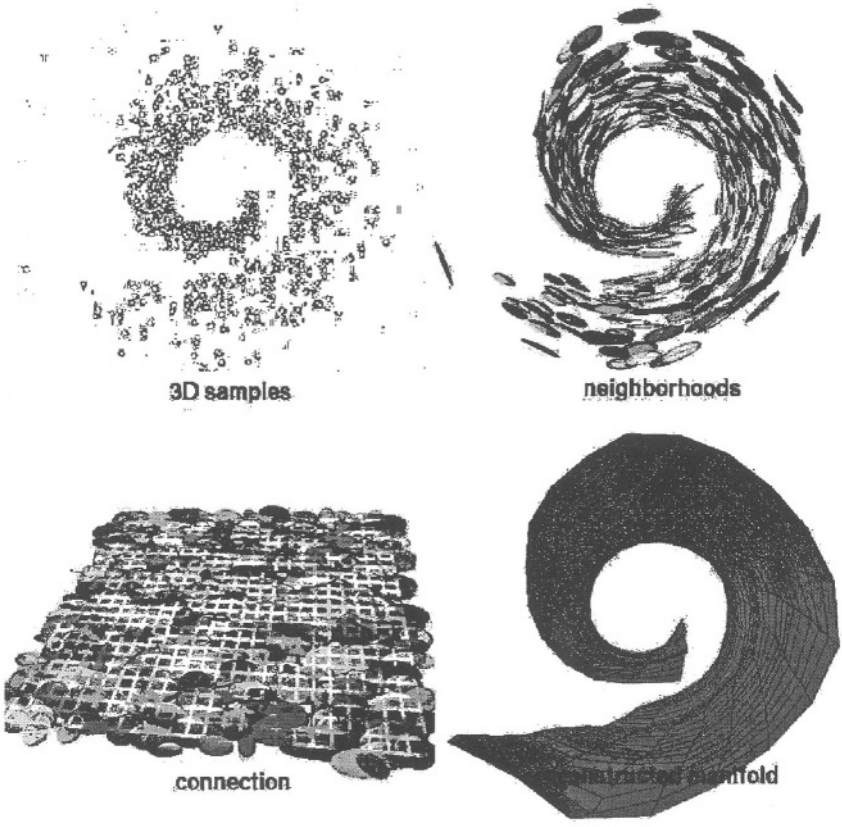


Fig. 4. A kernel-based reconstruction of a non-linear manifold.

random, there are interesting questions as to convergence rate and stability over time, and thence sample and computational complexity.

Many data sets do not comfortably fit in low-dimensional linear subspaces. Instead, the data lies on some low-dimensional manifold embedded with curvature in the high-dimensional measurement space. Recently there has been great ferment in nonlinear dimensionality reduction, which aims to unfurl the manifold in a low-dimensional space so that the distribution of the data can be studied in a linear space. This problem area is a rich interface between graph theory, differential geometry, and statistics. Nearly all current methods have (unacknowledged) ancestry in Tutte's theorems on graph embeddings from the 1960s, which view data points as vertices in a locally connected graph that is to be embedded in a Euclidean space with minimal distortion. We have shown how to estimate smooth maps that relate the original data space to a coordinate system intrinsic to the manifold, so that, for example, high-resolution 3D scans of human faces can be assigned low-dimensional coordinates, and novel faces can be synthesized

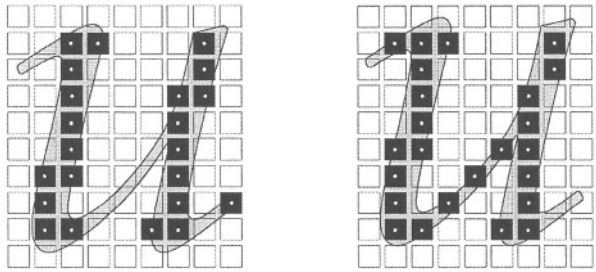


Fig. 5. Hints for grid fitting and other geometric adjustments radically improve the quality of rendered type for conventional outline fonts. An unhinted rendering is shown on the left; a hinted rendering of the same character is shown on the right.

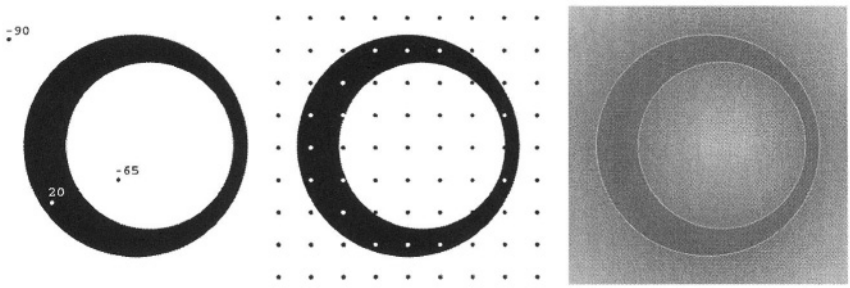


Fig. 6. A distance field gives signed distance values at each point in the plane: these values correspond to the shortest distance to an outline edge. The continuous field depicted on the right can be represented inefficiently by the discrete regular samples depicted in the center.

by varying those coordinates as if they were a linear system [22]. Although this depends on a locally linear approximation of the manifold, we have shown that a slight elaboration of the scheme is capable of exact isometric embeddings of a significant class of curved manifolds that includes developable surfaces. A sample reconstruction is shown in Figure 4. Although these methods have already seen extensive practical use, the relationship between topology, geometry, and sample complexity remains largely unexplored.

6 Shape Representation

The representation of geometry is a fundamental topic in computational geometry. Different representations are usually compared in terms of the computational efficiency of different computations on those representations. However, efficiency of computation is but one criterion: for digital typography several other measures are relevant. These additional criteria include perceptual measures, aesthetic

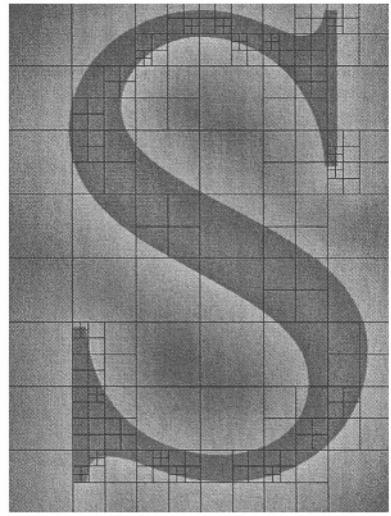


Fig. 7. A depiction of an adaptively sampled distance field for the character 'S'.

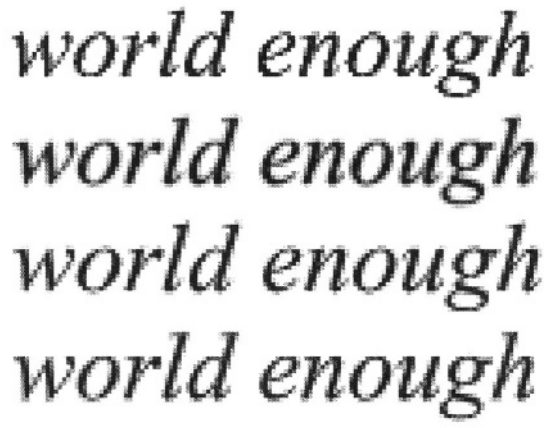


Fig. 8. A comparison of different representations and rendering algorithms, from top to bottom: unhinted outlines, box filter, four samples per pixel; unhinted outlines, Gaussian filter, sixteen samples per pixel; hinted outlines with sophisticated filtering and multiple samples per pixel (a proprietary algorithm); unhinted adaptive distance fields (Saffron), one sample per pixel.

design considerations, and ease of hardware implementation: these criteria are inherently empirical.

The dominant paradigm for representing high-quality, antialiased, scalable type is hinted outline fonts [23]. Outline fonts render horribly as they scale unless hints—arbitrary procedures—are provided that perform grid fitting and

geometric adjustment for a given scale (see Figure 5). Hints have to be crafted by hand for each typeface by skilled typographers. The arbitrary nature of hints makes hardware implementation problematic.

Saffron is a new digital font technology that represents 2D shape with adaptively sampled distance fields [24,25]. A distance field is an implicit representation of shape. A continuous distance field can be represented by regular samples (see Figure 6). However, regularly sampled distance fields are too big and inefficient and do not provide enough detail in some critical regions, such as corners.

Adaptive sampling preserves geometric detail where needed. An adaptively sampled distance field can be stored in an efficient spatial data structure (see Figure 7). The original shape can be reconstructed using various methods: biquadratic interpolation is one good technique. Antialiased images can be computed directly from the distance field, in contrast to the approximate area-coverage computations used with other representations (see Figure 8). Thus Saffron type can be rendered without hints. The simple rendering algorithm is amenable to hardware implementation.

7 Conclusions

It is likely that the projects described here are very different from those described in the other papers in this proceedings. Theoretical computer science is useful for applied computing, but usually only when combined with other techniques and when evaluated and refined empirically. A broader view of what constitutes computer-science theory can make it more useful for the many varied problems that arise in applied computing.

References

1. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Scientific, Belmont, Massachusetts (2000) Volumes 1 and 2.
2. Yokouchi, K., Ideno, H., Ota, M.: Car-navigation systems. Mitsubishi Electric Advance **91** (2000) 10–14
3. Nishiuma, N., Kumazawa, H., Goto, Y., Nikovski, D., Brand, M.: Traffic prediction using singular value decomposition. In: Proceedings of the 11th ITS World Congress (to appear), Nagoya, Japan (2004)
4. Nagel, K., Rasmussen, S.: Traffic at the edge of chaos. In: Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, Cambridge, MA, MIT Press (1994)
5. Nikovski, D., Brand, M.: Decision-theoretic group elevator scheduling. In: 13th International Conference on Automated Planning and Scheduling, Trento, Italy, AAAI (2003) 133–142
6. Klau, G., Lesh, N., Marks, J., Mitzenmacher, M.: Human-guided tabu search. In: Proceedings of AAAI 2002. (2002) 41–47
7. Lesh, N., Mitzenmacher, M., Whitesides, S.: A complete and effective move set for simplified protein folding. In: Proceedings of the 7th Intl. Conf. on Research in Computational Molecular Biology. (2003) 188–195

8. Lesh, N., Marks, J., McMahon, A., Mitzenmacher, M.: New exhaustive, heuristic, and interactive approaches to 2D rectangular strip packing. Technical Report TR2003-05, Mitsubishi Electric Research Laboratories (MERL) (2003)
9. Lesh, N., Marks, J., McMahon, A., Mitzenmacher, M.: Exhaustive approaches to 2D rectangular perfect packings. Information Processing Letters **90** (2004) 7–14
10. Baker, B.S., E. G. Coffman, J., , Rivest, R.L.: Orthogonal packings in two dimensions. SIAM Journal on Computing **9** (1980) 846–855
11. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. In Lakemeyer, G., Nebel, B., eds.: Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann (2003) 239–269
12. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free energy approximations and generalized belief propagation algorithms. Technical Report TR2004–040, Mitsubishi Electric Research Laboratories (MERL) (2004)
13. Welling, M., Teh, Y.W.: Belief optimization for binary networks: A stable alternative to belief propagation. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence. (2001) 554–561
14. Yuille, A.L.: CCCP algorithms to minimize the Bethe and Kikuchi free energies: Convergent alternatives to belief propagation. Neural Computation **14** (2002) 1691–1722
15. Shibuya, T., Harada, K., Tohyama, R., Sakaniwa, K.: Iterative decoding based on concave-convex procedure. In review (2004)
16. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Characterizing belief propagation and its generalizations. Technical Report TR2001-15, Mitsubishi Electric Research Laboratories (MERL) (2001)
17. Yedidia, J.S., Chen, J., Fossorier, M.: Generating code representations suitable for belief propagation decoding. In: Proceedings of the 40th Allerton Conference on Commmunication, Control, and Computing. (2002)
18. Tappen, M.F., Freeman, W.T., Adelson, E.H.: Recovering intrinsic images from a single image. In: Advances in Neural Information Processing Systems. Volume 15., MIT Press (2003)
19. Dauwels, J., Loeliger, H.A., Merkli, P., Ostojic, M.: Structured-summary propagation, LFSR synchronization, and low-complexity trellis decoding. In: Proceedings of the 41st Allerton Conference on Communication, Control, and Computing. (2003) 459–467
20. Storkey, A.: Generalized propagation for fast fourier transforms with partial or missing data. In: Advances in Neural Information Processing Systems 16. MIT Press (2004)
21. Brand, M.: Incremental singular value decomposition of uncertain data with missing values. In: Proceedings of the European Conference on Computer Vision. Volume 2350., Berlin, Springer (2003) 707–720
22. Brand, M.: Continuous nonlinear dimensionality reduction by kernel eigenmaps. In Gottlob, G., Walsh, T., eds.: Proceedings of the International Joint Conference on Artificial Intelligence, San Mateo, CA, Morgan Kaufmann (2003) 547–552
23. Adode Systems, Inc.: Adobe Type 1 Font Format. Addison Wesley (1990)
24. Frisken, S., Perry, R., Rockwood, A., Jones, T.: Adaptively sampled distance fields: A general representation of shape for computer graphics. In: Proceedings of SIGGRAPH 2000. (2000) 249–254
25. Perry, R., Frisken, S.: A new framework for representing, rendering, editing, and animating type. (In preparation.)

Boxed Ambients with Communication Interfaces*

Eduardo Bonelli[1], Adriana Compagnoni[1],
Mariangiola Dezani-Ciancaglini[2], and Pablo Garralda[1]

[1] Stevens Institute of Technology, U.S.A.
[2] Università di Torino, Italy

Abstract. We define **BACI** *(Boxed Ambients with Communication Interfaces),* an ambient calculus allowing a liberal communication policy. Each ambient carries its local view of the topic of conversation (the type of the information being exchanged) with parents and children that will condition where it is allowed to stay or migrate to and which ambients may be allowed to enter it. The topic of conversation view of ambients can dynamically change during migration. **BACI** is flexible enough to allow different topics of conversation between an ambient and different parents, without compromising type-safety: it uses port names for communication and ambient names for mobility. Capabilities and co-capabilities exchange port names and run-time typing information to control mobility. We show the type-soundness of **BACI** proving that it satisfies the subject reduction property. Moreover we study its behavioural semantics by means of a labelled transition system.

1 Introduction

In an ambient calculus one can distinguish two forms of dynamic behavior: *communication* and *migration* [10]. By communication we mean the exchange of information between processes possibly located in different ambients. By migration, we mean the ability of an ambient to relocate itself by entering or exiting other ambients. Communication and migration are deeply related, since migration may enable or disable communication and vice-versa.

In calculi such as BA and NBA, and those in [6,23,7,11], an ambient can communicate with its parent ambient (the host ambient) or with a child ambient (an ambient it contains), and there may also be local communication among the processes within an ambient. In typed ambient calculi, communication is controlled by types, and the type of information being exchanged is often called *topic of conversation (TOC).* For example, if an ambient sends the number 3 to its parent, we can say that the TOC is Int.

* This research was partially supported by the EU within the FET – Global Computing initiative, project DART ST-2001-33477, by MURST Cofin'02 project McTati, and by the USA under the National Science Foundation project No. CCR-0220286 ITR: Secure Electronic Transactions. The funding bodies are not responsible for any use that might be made of the results presented here.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 119–148, 2004.
© Springer-Verlag Berlin Heidelberg 2004

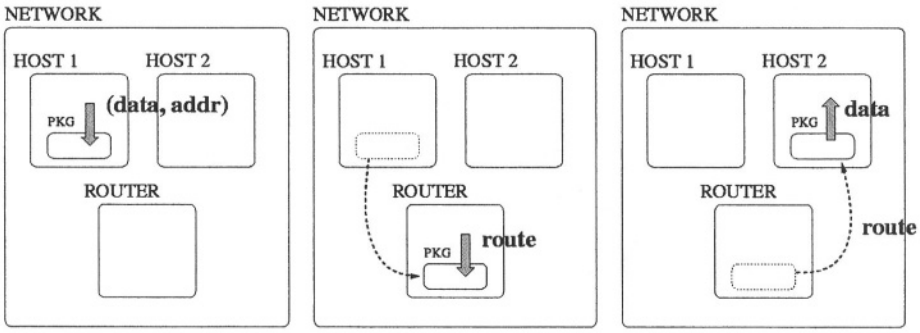


Fig. 1. Example of an ambient using different TOCs with different parents

Furthermore, notice that migration (entering or exiting an ambient) changes the parent of an ambient. The existing typed mobile ambient calculi fix a TOC for communication with the parent for each ambient, and even if migrating changes the parent the TOC remains fixed.

In this paper we introduce **BACI,** a new mobile ambient calculus where each ambient carries a communication interface specifying how an ambient may interact with the environment. The design of **BACI** was driven by the desire to lift the fixed-TOC-with-parent restriction allowing an ambient to change TOC when changing parents and enabling straightforward design of ambients that need to exchange information of different types with different ambients.

Consider the example in Figure 1, where HOST 1 needs to send **data** to HOST 2, but HOST 1 does not know where HOST 2 is located. However, HOST 1 knows the location **(addr)** of a ROUTER that can forward to HOST 2 the packet (PKG) containing the **data.** Assuming this, HOST 1 spawns the packet and forwards the data to be transported along with the location of the router. Next, the packet moves inside the ROUTER where it obtains the **route** to HOST 2. Finally, using that **route** the packet reaches HOST 2 and delivers the **data.**

Notice that the PKG ambient uses three different communication types with its three different parents (i.e. HOST 1, ROUTER and HOST 2). In order to implement this example in calculi where each ambient has a fixed type for parent communication, additional messenger ambients are needed to encode the communication with the different parents, using an auxiliary messenger ambient for each communication type. **BACI's** new features offer more flexibility to the designer to deliver more natural specifications.

Ports and Names. The communication with a child ambient is often labeled with the ambient's name (named communication):

$$n[\langle 3\rangle^{\downarrow m} | m[\cdots] \mid \cdots] \quad \text{(ambient } n \text{ wants to send 3 to its child } m\text{)}$$

However, in communication with a parent the name is often left implicit, since the parent can be uniquely determined by the location of an ambient.

$$n[m[\langle 3\rangle^{\uparrow} \mid \cdots] \mid \cdots] \quad \text{(ambient } m \text{ wants to send 3 to its parent } n\text{)}$$

In order to allow different TOCs with different parents, **BACI** introduces named communication with parents.

As we mentioned earlier, we can distinguish between two forms of dynamic behavior: communication and mobility. Previous calculi used ambient names to specify both communication and mobility. However, adding the possibility to communicate with different parents in different types stressed the difference between them. Our calculus, **BACI**, induces the separation of concerns by using names only for mobility and introducing the concept of *ports* for communication. Therefore to migrate to an ambient we use its name, but to exchange information with it we use its port. In this framework, an ambient n with communication port c is written $n[c\| \cdots]$, so for example:

$n[c_n\| \cdots \mid m[c_m\|\langle 3\rangle^{\uparrow c_n}] \mid \cdots]$ (ambient m wants to send 3 to parent port c_n)

The introduction of communication ports naturally leads to associate TOCs to ports, instead of associating TOCs to ambient names as usual. There is no global knowledge on this association: each ambient has its *local view* which can dynamically increase with relocation. An ambient n with port c and local view Γ is written $n[\Gamma\|c\| \cdots]$. Then our last example becomes:

$$n[\cdots \mathsf{Int}^{\downarrow c_m}\|c_n\| \cdots \mid m[\cdots \mathsf{Int}^{\uparrow c_n}\|c_m\|\langle 3\rangle^{\uparrow c_n} \mid \cdots]]$$

To sum up, in **BACI** each ambient comes equipped with *its own local communication interface.* A communication interface consists of

- a *communication port* used by the other ambients to communicate with the current one and
- a *local view* associating topics of conversation to parent and child ports.

Communication interfaces are required for communication across ambient boundaries. In order for communication across ambient boundaries to take place a pair of communication ports must be coupled: a *sending communication port* (provided by the sending ambient) and a *receiving communication port* (provided by the receiving ambient). The following (INPUT $\downarrow$-$\uparrow$) reduction rule models one form of communication across ambient boundaries (the other one consists in reversing the roles of sending ambient and receiving ambient)

$$m[\Gamma_m\|c_m\|(\tilde{x}:\tilde{\varphi})^{\downarrow c_n}.P \mid n[\Gamma_n\|c_n\|\langle\tilde{M}\rangle^{\uparrow c_m}.Q \mid R] \mid S] \longrightarrow m[\Gamma_m\|c_m\|P\{\tilde{x} := \tilde{M}\} \mid n[\Gamma_n\|c_n\|Q \mid R] \mid S]$$

The sending ambient is n and the receiving ambient is m. The communication interface of m consists of the communication port c_m and the local view Γ_m. Likewise, the communication interface of n consists of the communication port c_n and the local view Γ_n. In order for communication to succeed, the type of the information sent through the sending communication port (c_n) must coincide with the one expected by the receiving communication port (c_m). More precisely, the message $\tilde{M}$ in n should have some type $\tilde{\varphi}$ that coincides with the one that the ambient m is expecting.

Non-determinism. The same port name may be used by different ambients. For example, in

$$n[\Gamma_n \| c \| \langle q \rangle^{\uparrow c_p} \mid P] \mid m[\Gamma_m \| c \| \langle \overline{\mathsf{in}} \rangle^{\uparrow c_p} \mid Q]$$

both ambients use the same port name c as part of their communication interfaces. Moreover, both ambients intend to communicate with a parent port c_p, but in different ways: ambient n wants to send the name q while ambient m wants to send the capability $\overline{\mathsf{in}}$ (which is used to allow an ambient to enter). In other words, for n, c_p is a parent port for exchanging ambient names, and Γ_n contains $\mathsf{amb}^{\uparrow c_p}$, and for m, c_p is a parent port for exchanging capabilities and Γ_m contains $\mathsf{cap}^{\uparrow c_p}$. As a consequence, only one of these ambients would be able to enter a host ambient p with communication port c_p willing to communicate with them. For example, in

$$p[\Gamma_p \| c_p \| (x : \varphi)^{\downarrow c}.R \mid \overline{\mathsf{in}}.\mathbf{0}] \mid n[\Gamma_n \| c \| \langle q \rangle^{\uparrow c_p} \mid \mathsf{in}\ p.\mathbf{0}] \mid m[\Gamma_m \| c \| \langle \overline{\mathsf{in}} \rangle^{\uparrow c_p} \mid \mathsf{in}\ p.\mathbf{0}]$$

if $\varphi = \downarrow c(\Gamma_p) = \mathsf{amb}$, then n is allowed to enter and if $\varphi = \downarrow c(\Gamma_p) = \mathsf{cap}$ then m is allowed to enter. Both of the above examples are typable in our system assuming that the port of the host ambient is different from c_p.

Related Work. Modeling the word wide web requires the space and the mobility in space as new dimensions of computing. In the first proposals, i.e. in the *Dπ-calculus* [18] and in the language *Klaim* [14], the structure of locations is fiat. Instead the *Mobile Ambient (MA)* calculus [10] deals with a hierarchical structure of locations (called ambients). An interesting *core model* generalising many of the available calculi and languages has been developed inside the Mikado project [4].

Many variants of MA have been designed: for a tutorial see [15]. A crucial choice in all these calculi is the form of *interaction* between processes in different ambients. In the original calculus [10] interaction is only local to an ambient, and in order for processes in different ambients to communicate, at least one of the ambients' boundaries have to be dissolved. In [10,20,5,22,1], the open capability dissolves the ambient boundary. The calculus **M3** [13,12], allows general process mobility. In *Boxed Ambients (BA)* [6,23,7], parents and children can communicate as in the *Seal* calculus [11]. Our calculus, **BACI,** follows this last protocol. The *co-actions* (first introduced in [20], and then used with modifications in [5, 22,23,7]) require also the agreement of the "passive" ambients involved in mobility. The co-actions of **BACI** in which port names are communicated are inspired by those of [7]: there the communication involves only the name of the entering ambient.

Ambient calculi are often typed: the types assure behavioural properties concerning communication, mobility, resource access, security, etc. [8,20,9,5,1,6,22, 23,7,21]. To our knowledge before **BACI** only the calculi of [5] and [12] consider *type information local to ambients,* while in the other proposals there is a global environment containing all typing assumptions. When dealing with computing in wide area "open" systems it is sensible to assume the existence of different local environments. The price to pay is that static checks are no longer enough

to assure correctness: we now need to carry typing information at run time. Following ideas from [16] we define an *operational semantics with types,* which is simpler that a fully-fledged *typed operational semantics* in the sense that we only need to check agreement between the local views upon mobility. In some sense it can be seen as special case of *proof carrying code* [24]. The local type information in **BACI** can dynamically increase with ambient movements: something similar happens in the version of Dπ-calculus considered in [19].

Behavioural types [1,2] look mainly as computational traces: they allow polymorphic communications. **BACI**'s communication interfaces also are a permissive tool for typing non local communications. In [2] the type of communication with the parent changes when communication takes place. However, they do not have named communication with the parent, and cannot express the fact that communication with different parents has different types as in our last example.

Paper Plan. The rest of this paper is structured as follows. Section 2 introduces the syntax of the calculus and its operational semantics. Section 3 presents the type system. Section 4 studies a reduction barbed congruence and a labeled transition system (LTS). The bisimilarity induced by the LTS is shown to be sound with respect to the congruence. Some congruence laws are also identified. Section 5 discusses two extended examples highlighting the features of **BACI.** Finally, we conclude and suggest further research.

2 The Calculus

2.1 Terms and Types

The syntax for types and terms in **BACI** is given in Table 1. Notice that **BACI** is a typed calculus and as such by process we always mean a well-formed process according to the rules of Table 6.We assume two disjoint denumerable sets of variables one for name, capability and message variables, and the other for port variables. We use $m, n, o, p, q \ldots$ for ambient name constants and $x, y, z, \ldots$ for ambient name variables, while α, β range over both ambient constants and ambient variables. Communication port constants are written $c, c_n, \ldots$ and $v, v', \ldots$ are used for communication port variables, while γ is either a communication port constant or variable. The expressions $\mathsf{inC}(v : \rho)\ \alpha$, $\mathsf{outC}(v : \rho)\ \alpha$, $\overline{\mathsf{inC}}(v : \rho), \overline{\mathsf{outC}}(v : \rho)$ are *binders* for v in prefixes and processes.

The process $\mathbf{0}$ is the null process; $P_1 \mid P_2$ denotes the parallel composition of processes P_1 and P_2; $(\nu n)P$ is the usual restriction operator that binds all free occurrences of n in P; ! is the replication operator. The expression $\pi.P$ denotes the process that performs an action (or a co-action) π and then continues with P. The π actions includes input/output (I/O) actions and mobility actions. The I/O exchanges are directed upwards to the parent ambient, downwards to a child ambient or locally to other processes at the same level. The direction of each communication is determined by η. The location η is the location of the communication exchange: $\uparrow \gamma$ denotes communication with a parent having the port γ, $\downarrow \gamma$ denotes communication with a child with port γ, and $\star$ denotes local communication.

Table 1. Syntax of **BACI**

Basic types	
$\varphi ::= \mathsf{amb}$	ambient
$\mid \mathsf{cap}$	capability
Located types	
$\tau ::= (\varphi_1, \ldots, \varphi_k)^\eta$	
Names:	
$\alpha, \beta ::= n$	name constant
$\mid x$	name variable
(Co)-Capabilities:	
$C, D ::= \mathsf{in}\ \alpha$	enter
$\mid \mathsf{out}\ \alpha$	exit
$\mid \overline{\mathsf{in}}$	allow enter
$\mid \overline{\mathsf{out}}$	allow exit
$\mid C.D$	path
$\mid x$	capability variable
Prefixes:	
$\pi ::= C$	capabilities
$\mid (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^\eta$	input
$\mid \langle M_1, \ldots, M_k \rangle^\eta$	output
$\mid \mathsf{inC}(v : \rho)\ \alpha$	port enter
$\mid \mathsf{outC}(v : \rho)\ \alpha$	port exit
$\mid \overline{\mathsf{inC}}(v : \rho)$	allow port enter
$\mid \overline{\mathsf{outC}}(v : \rho)$	allow port exit
Communication types	
$\rho ::= \mathsf{shh}$	no exchange
$\mid \varphi_1, \ldots, \varphi_k$	exchange tuple
Local view	
$\Gamma ::= \emptyset$	empty
$\mid \Gamma, \tau$	interface
Ports:	
$\gamma ::= c$	port constant
$\mid v$	port variable
Locations:	
$\eta ::= \uparrow \gamma$	parent port γ
$\mid \downarrow \gamma$	child port γ
$\mid \star$	local
Messages:	
$M, N ::= \alpha$	name
$\mid C$	capability
$\mid x$	message variable
Processes:	
$P ::= \mathbf{0}$	nil process
$\mid P_1 \mid P_2$	composition
$\mid (\nu n)P$	restriction
$\mid !P$	replication
$\mid \pi.P$	prefixing
$\mid \alpha[\Gamma \| c \| P]$	ambient

The information exchanged in input/output are tuples of messages. Each message can be either an ambient name, or a (co)-capability[1]. The ambient names received as messages, can substitute a variable ambient name in an ambient constructor or in a capability. Capabilities and co-capabilities constitute the mobility actions and co-actions: the capabilities allow an ambient to enter another ambient α, using $\mathsf{in}\ \alpha$; or to exit an ambient α, using $\mathsf{out}\ \alpha$. In order to be executed, each capability must be matched with its respective co-capability: in with $\overline{\mathsf{in}}$ and out with $\overline{\mathsf{out}}$. Both capabilities and co-capabilities can be sent as messages. A single (co)-capability or several (co)-capabilities forming a path may be sent.

Beside these standard mobility actions and co-actions, **BACI** introduces the inC and outC actions and their corresponding co-actions $\overline{\mathsf{inC}}$ and $\overline{\mathsf{outC}}$. These actions and co-actions are similar to the enter and exit (co)-capabilities. However, they also have a port variable that is bound at execution time with the port of the counterpart ambient involved in the mobility action. Because **BACI** uses

[1] The types of messages can be easily extended to handle basic types such as integer or boolean without any technical problems, but that is omitted here for the sake of simplicity.

Table 2. Structural Congruence

$!P \equiv P \mid !P$	(STRUCT REP PAR)
$(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(STRUCT RES RES)
$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$, if $n \notin \mathrm{fn}(P)$	(STRUCT RES PAR)
$(\nu n)m[\Gamma_m \| c_m \| P] \equiv m[\Gamma_m \| c_m \| (\nu n)P]$, if $n \neq m$	(STRUCT RES AMB)
$\mathsf{in}\ n.P \equiv \mathsf{inC}(v : \mathsf{shh})\ n\ .P$, if $v \notin \mathrm{fv}(P)$	(STRUCT IN INC)
$\mathsf{out}\ n.P \equiv \mathsf{outC}(v : \mathsf{shh})\ n\ .P$, if $v \notin \mathrm{fv}(P)$	(STRUCT OUT OUTC)
$\overline{\mathsf{in}}.P \equiv \overline{\mathsf{inC}}(v : \mathsf{shh}).P$, if $v \notin \mathrm{fv}(P)$	(STRUCT CO-IN CO-INC)
$\overline{\mathsf{out}}.P \equiv \overline{\mathsf{outC}}(v : \mathsf{shh}).P$, if $v \notin \mathrm{fv}(P)$	(STRUCT CO-OUT CO-OUTC)
$(C.D).P \equiv C.D.P$	(STRUCT .)

port names exclusively for communication and ambient names only for mobility, knowing the name of an ambient is not enough to establish a communication, the port associated with that ambient must be known as well.

Port names cannot be sent as messages; therefore, the only way of learning a port name is by using the inC and outC actions with their co-actions. In their execution, the ambients affected by this action exchange port names using the binders in these special (co)-actions. Additionally, in order to retain typability, port variables have an associated communication type ρ. If the communication type is shh, there is no exchange of information, otherwise an exchange tuple of basic types is used to indicate exchange of information of that type. The types used on these actions and co-actions must be compatible, their relation will be established by the operational semantics rules.

An ambient is written $\alpha[\Gamma \| c \| P]$ where α is an ambient name constant or an ambient name variable and P the enclosed process. The local view Γ is a finite set of located types, i.e. of exchange tuple types decorated with a location. The local view is used to specify the communication type of the enclosed process P. The local view together with the communication port c constitute the communication interface of the ambient.

Terms differing only in the names of their bound variables are considered equal. Furthermore, *Barendregt's convention* [3] is assumed: all variables are pairwise distinct and distinct from all free variables. This avoids cluttering the presentation with conditions on the names of variables in order to prevent variable clash and variable capture.

2.2 Operational Semantics

The operational semantics is defined in terms of structural congruence and reduction rules.

Structural congruence is the least congruence such that $(\mid, \mathbf{0})$ is a commutative monoid and the axioms of Table 2 are satisfied. The definition is standard except for the rules in the second group (where $\mathrm{fv}(P)$ is the set of free vari-

Table 3. Operations on Locations and Local Views

Join of communication types

$$\rho \sqcup \rho' = \begin{cases} \rho & \text{if } \rho = \rho' \text{ or } \rho' = \mathsf{shh}, \\ \rho' & \text{if } \rho = \mathsf{shh}, \\ \bot & \text{otherwise.} \end{cases}$$

Preorder on communication types

$$\rho \preceq \rho' \text{ iff } \rho \sqcup \rho' = \rho'$$

Addition of located types to local views

$$\Gamma \oplus \rho^\eta = \begin{cases} \Gamma & \text{if } \rho = \mathsf{shh}, \\ \Gamma, \rho^\eta & \text{otherwise.} \end{cases}$$

Application of locations

- to located types

$$\eta(\tau) = \begin{cases} \rho & \text{if } \tau = \rho^\eta, \\ \mathsf{shh} & \text{otherwise.} \end{cases}$$

- to local views

$$\begin{aligned} \eta(\emptyset) &= \mathsf{shh} \\ \eta(\Gamma) &= \bigsqcup_{\tau \in \Gamma} \eta(\tau) \end{aligned}$$

Location substitution

- for prefixes

$$\pi\{\eta := \eta'\} = \begin{cases} (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^{\eta'} & \text{if } \pi = (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^\eta, \\ \langle M_1, \ldots, M_k \rangle^{\eta'} & \text{if } \pi = \langle M_1, \ldots, M_k \rangle^\eta, \\ \pi & \text{otherwise.} \end{cases}$$

- for processes

  - $\mathbf{0}\{\eta := \eta'\} = \mathbf{0}$
  - $(P_1 \mid P_2)\{\eta := \eta'\} = P_1\{\eta := \eta'\} \mid P_2\{\eta := \eta'\}$
  - $((\boldsymbol{\nu} n)P)\{\eta := \eta'\} = (\boldsymbol{\nu} n)P\{\eta := \eta'\}$
  - $(!P)\{\eta := \eta'\} = !P\{\eta := \eta'\}$
  - $(\alpha[\Gamma_\alpha \| c_\alpha \| P])\{\eta := \eta'\} = \alpha[\Gamma_\alpha \| c_\alpha \| P]$
  - $(\pi.P)\{\eta := \eta'\} = \pi\{\eta := \eta'\}.P\{\eta := \eta'\}$

ables occurring in P). They state that the (co-)capabilities in n, out n, $\overline{\mathsf{in}}$, $\overline{\mathsf{out}}$ may be identified with particular instances of the prefixes "port enter/exit" and "allow port enter/exit". The rationale here is that these prefixes behave as the corresponding (co-) capabilities when they cancel the communicated port name (condition $v \notin \mathrm{fv}(P)$) and no topic of conversation is communicated (i.e. the

Table 4. Operational Semantics

Mobility

(ENTER)

$$n[\Gamma_n \| c_n \| \mathsf{inC}(v:\rho)\ m\ .P_1 \mid P_2] \mid m[\Gamma_m \| c_m \| \overline{\mathsf{inC}}(v':\rho').Q_1 \mid Q_2] \longrightarrow m[\Gamma_m \oplus \rho'^{\downarrow c_n} \| c_m \| n[\Gamma_n \oplus \rho^{\uparrow c_m} \| c_n \| P_1\{\uparrow v := \uparrow c_m\} \mid P_2] \mid Q_1\{\downarrow v' := \downarrow c_n\} \mid Q_2] \quad \text{if } \uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$$

(EXIT)

$$p[\Gamma_p \| c_p \| n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| \mathsf{outC}(v:\rho)\ n\ .P_1 \mid P_2] \mid Q] \mid \overline{\mathsf{outC}}(v':\rho').R_1 \mid R_2] \longrightarrow p[\Gamma_p \oplus \rho'^{\downarrow c_m} \| c_p \| m[\Gamma_m \oplus \rho^{\uparrow c_p} \| c_m \| P_1\{\uparrow v := \uparrow c_p\} \mid P_2] \mid n[\Gamma_n \| c_n \| Q] \mid R_1\{\downarrow v' := \downarrow c_m\} \mid R_2] \quad \text{if } \uparrow c_p(\Gamma_m) \sqcup \rho \preceq \downarrow c_m(\Gamma_p) \sqcup \rho'$$

Communication

(LOCAL)

$$(\tilde{x}:\tilde{\varphi})^\star.P \mid \langle \tilde{M} \rangle^\star.Q \longrightarrow P\{\tilde{x} := \tilde{M}\} \mid Q$$

(INPUT ↓-↑)

$$m[\Gamma_m \| c_m \| (\tilde{x}:\tilde{\varphi})^{\downarrow c_n}.P \mid n[\Gamma_n \| c_n \| \langle \tilde{M} \rangle^{\uparrow c_m}.Q \mid R] \mid S] \longrightarrow m[\Gamma_m \| c_m \| P\{\tilde{x} := \tilde{M}\} \mid n[\Gamma_n \| c_n \| Q \mid R] \mid S]$$

(OUTPUT ↓-↑)

$$m[\Gamma_m \| c_m \| \langle \tilde{M} \rangle^{\downarrow c_n}.P \mid n[\Gamma_n \| c_n \| (\tilde{x}:\tilde{\varphi})^{\uparrow c_m}.Q \mid R] \mid S] \longrightarrow m[\Gamma_m \| c_m \| P \mid n[\Gamma \| c_n \| Q\{\tilde{x} := \tilde{M}\} \mid R] \mid S]$$

Structural Rules

(STRUCT)

$$\frac{P \equiv P', \quad P' \longrightarrow Q', \quad Q' \equiv Q}{P \longrightarrow Q}$$

(CONTEXT)

$$\frac{P \longrightarrow Q}{\mathbf{E}\{P\} \longrightarrow \mathbf{E}\{Q\}}$$

Evaluation Contexts

$$\mathbf{E} ::= \{\cdot\} \mid \mathbf{E}|P \mid P|\mathbf{E} \mid (\nu n)\mathbf{E} \mid \alpha[\Gamma \| c \| \mathbf{E}]$$

communication type is shh). We cannot, however, do away completely with the (co-)capabilities since, in contrast to the aforementioned port prefixes, they may be sent as messages.

The reduction relation is given by three groups of rules: *mobility, communication* and *structural.* The structural rules are standard. Before describing mobility and communication, we need some definitions given in Table 3.

The join of communication types $(\rho \sqcup \rho')$ and the preorder on them $(\rho \preceq \rho')$ simply state that shh is smaller than any other communication type.

The addition of an expression ρ^η to a local view Γ $(\Gamma \oplus \rho^\eta)$ changes Γ only if ρ is a tuple of basic types, i.e. if $\rho \neq \mathsf{shh}$. Notice that only in this last case ρ^η is a located type.

The application of a location η to a located type τ $(\eta(\tau))$ returns either the communication type of τ or shh according to whether the location of τ is η or not. The application of a location η to a local view Γ $(\eta(\Gamma))$ is the join of the applications of η to the located types in Γ. So $\eta(\Gamma)$ is different from $\perp$ only if Γ contains at most one located type whose location is η; $\eta(\Gamma)$ is shh when Γ contains no located type whose location is η.

The location substitution $(\{\eta := \eta'\})$ replaces locations as superscripts of input and output prefixes. They propagate on processes in the standard way but they never cross ambient boundaries.

The *mobility rules* consist of (ENTER) and (EXIT). Since they may both be explained along similar lines, we discuss only the former. The (ENTER) rule allows an ambient n to enter a sibling ambient m. Once n has entered m, communication may take place. However, this requires that n directs its messages through m's communication port (namely c_m) and, likewise, that m directs its messages through n's communication port (namely c_n). Since n may not know the name of m's communication port, the capability $\mathsf{inC}(v : \rho)\ m$ and the co-capability $\overline{\mathsf{inC}}(v' : \rho')$ provide port variables v and v' for such communication ports to be made available to the interested parties.

Note that the type of the information that may be exchanged on the port is also provided at run-time so that n and m may use compatible topics of conversation. Indeed, as a consequence of n entering m, the local views of both ambients are updated. Since this takes place at run-time, appropriate checks are required in order to guarantee that such extensions are sound. This is the role of the condition $\uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'$. This condition may be explained as follows:

1. First notice that $\uparrow c_m(\Gamma_n) \sqcup \rho$ must be defined, since $\preceq$ is a partial relation between communication types. This implies that the new communication type ρ with location $\uparrow c_m$ to be added to the local view Γ_n is compatible with any existing located type in Γ_n. A similar comment applies to $\downarrow c_n(\Gamma_m) \sqcup \rho'$.
2. Assuming $\rho_1 = \uparrow c_m(\Gamma_n) \sqcup \rho$ and $\rho_2 = \downarrow c_n(\Gamma_m) \sqcup \rho'$, the condition $\rho_1 \preceq \rho_2$ checks to see whether n and m agree on a topic of conversation. Note that n may safely avoid listening to m but not vice-versa.

The following example shows why children can safely avoid to listen to parents but not vice-versa, i.e. why the condition $\rho_2 \preceq \rho_1$ is unsafe.

Take the process

$$p[\emptyset \| c \| \overline{\mathsf{inC}}(v_1 : \mathsf{shh}) \mid \overline{\mathsf{inC}}(v_2 : \mathsf{amb}).\langle q \rangle^{\downarrow v_2}]$$
$$\mid m[\mathsf{cap}^{\uparrow c} \| c' \| \mathsf{inC}(v : \mathsf{cap})\ p \mid (x : \mathsf{cap})^{\uparrow c}.P] \mid n[\emptyset \| c' \| \mathsf{inC}(v' : \mathsf{amb})\ p\ ]$$

Using rule (ENTER) with the pre-order condition reversed this process might reduce first to

$$p[\emptyset \| c \| \overline{\mathsf{inC}}(v_2 : \mathsf{amb}).\langle q \rangle^{\downarrow v_2} \mid m[\mathsf{cap}^{\uparrow c} \| c' \| (x : \mathsf{cap})^{\uparrow c}.P]] \mid n[\emptyset \| c' \| \mathsf{inC}(v' : \mathsf{amb})\ p\ ]$$

and then to

$$p[\mathsf{amb}^{\downarrow c'} \| c \| \langle q \rangle^{\downarrow c'} \mid m[\mathsf{cap}^{\uparrow c} \| c' \| (x : \mathsf{cap})^{\uparrow c}.P] \mid n[\mathsf{amb}^{\uparrow c} \| c' \|]]$$

and in this last process a wrong communication of an ambient name when a capability is expected could occur.

Notice that with the sound rule (ENTER) the initial process only reduces to

$$p[\mathsf{amb}^{\downarrow c'} \| c \| \overline{\mathsf{inC}}(v_1 : \mathsf{shh}) \mid \langle q \rangle^{\downarrow c'} \mid n[\mathsf{amb}^{\uparrow c} \| c' \|]] \mid m[\mathsf{cap}^{\uparrow c} \| c' \| \mathsf{inC}(v : \mathsf{cap})\ p \mid (x : \mathsf{cap})^{\uparrow c}.P]$$

i.e. only ambient n is allowed to go inside ambient p.

Thanks to the structural congruence between prefixes and (co-)capabilities, the standard enter and exit rules are mimicked by (ENTER) and (EXIT) , respectively. For example, the following (STANDARD ENTER) rule:

$$n[\Gamma_n \| c_n \| \mathsf{in}\ m.P_1 \mid P_2] \mid m[\Gamma_m \| c_m \| \overline{\mathsf{in}}.Q_1 \mid Q_2]$$
$$\longrightarrow$$
$$m[\Gamma_m \| c_m \| n[\Gamma_n \| c_n \| P_1 \mid P_2] \mid Q_1 \mid Q_2]$$
$$\text{if } \uparrow c_m(\Gamma_n) \preceq \downarrow c_n(\Gamma_m)$$

may be simulated by (ENTER). Indeed, if v, v' are any port variables such that $v \notin \mathsf{fv}(P_1)$ and $v \notin \mathsf{fv}(Q_1)$, then since $\mathsf{in}\ m.P_1 \equiv \mathsf{inC}(v : \mathsf{shh})\ m\ .P_1$ and $\overline{\mathsf{in}}.Q_1 \equiv \overline{\mathsf{inC}}(v : \mathsf{shh}).Q_1$, by applying (ENTER) we obtain

$$m[\Gamma_m \oplus \mathsf{shh}^{\downarrow c_n} \| c_m \| n[\Gamma_n \oplus \mathsf{shh}^{\uparrow c_m} \| c_n \| P_1\{\uparrow v := \uparrow c_m\} \mid P_2] \mid Q_1\{\downarrow v' := \downarrow c_n\} \mid Q_2]$$

Note that $P_1\{\uparrow v := \uparrow c_m\} = P_1$ and $Q_1\{\downarrow v' := \downarrow c_n\} = Q_1$. Also, $\Gamma_m \oplus \mathsf{shh}^{\downarrow c_n} = \Gamma_m$ and $\Gamma_n \oplus \mathsf{shh}^{\uparrow c_m} = \Gamma_n$ (cf. Table 3).

As for the *communication rules,* the local communications are standard, while the parent-child communications require the knowledge of the partner communication port, as already discussed in the introduction.

3 Typing Rules

The typing environment is very simple: it says if a variable stands for an ambient name or a capability.

Environments

$\Sigma ::= \emptyset$ empty environment
$\mid \quad \Sigma, x : \varphi$ environment

In the sequel, we only consider typing environments that assign a unique type to each name in its domain. The typing rules define two judgements:

Table 5. Well-formed Messages

$$\frac{n \in \mathcal{N}}{\Sigma \vdash n : \mathsf{amb}}\ (\mathsf{amb}) \qquad \frac{}{\Sigma, x : \varphi \vdash x : \varphi}\ (\mathrm{AXIOM}) \qquad \frac{C \in \{\overline{\mathsf{in}}, \overline{\mathsf{out}}\}}{\Sigma \vdash C : \mathsf{cap}}\ (\mathsf{cap}\text{-}\overline{\mathsf{in}}\ \text{-}\overline{\mathsf{out}})$$

$$\frac{\Sigma \vdash \alpha : \mathsf{amb} \quad C \in \{\mathsf{in}\ \alpha, \mathsf{out}\ \alpha\}}{\Sigma \vdash C : \mathsf{cap}}\ (\mathsf{cap}\text{-}\mathsf{in}\ \text{-}\mathsf{out}\ ) \qquad \frac{\Sigma \vdash C : \mathsf{cap} \quad \Sigma \vdash D : \mathsf{cap}}{\Sigma \vdash C.D : \mathsf{cap}}\ (\mathsf{cap}\text{-}\mathrm{COMP})$$

- $\Sigma \vdash M : \varphi$, read "$M$ is a well-formed message of type φ".
- $\Sigma \vdash_c P : \Gamma$, read "$P$ is a well-formed process assuming the local communication interface of its host consists of the communication port c and the local view Γ".

The typing rules for the first judgements appear in Table 5.

In contrast to other systems, the (amb) rule assigns an ambient name the constant type `amb` rather than a more informative type as in the majority of systems [15]. Indeed, more informative types presuppose the availability of (global) information on the type of ambients. In our setting based on local views the only assumption we make is that we can identify an ambient name when we see one.

The (cap-in -out) and ($\mathsf{cap}\text{-}\overline{\mathsf{in}}\ \text{-}\overline{\mathsf{out}}$) rules are also simpler than in formulations based on global knowledge of the communication types of ambients, since the corresponding control is delegated to run-time.

The (AXIOM) and (cap-COMP) rules are standard.

The rules defining the judgement $\Sigma \vdash_c P : \Gamma$ are given in Table 6.

Regarding the typing rule (**0**), since **0** does not interact with its host it may be typed under a communication interface consisting of any port name c and interface view Γ proviso Γ is *ok*, i.e. $\eta(\Gamma) \neq \bot$ for all η, and each port variable occurs at most once in Γ.

The rule for replication (PROC-REP) is standard, however (PROC-RES) is not. Usually, the name n together with its type is assumed to belong to the global environment Σ. However, in our local setting all we know is that n is an ambient name.

The rule for parallel composition (PROC-COMP) is also standard.

The typing rule (PROC-CAP) reveals that all what is known about a capability is that it is just a capability. Since we rely only on local information we shall relegate the correct use of capabilities at run-time.

A process of the form $\mathsf{inC}(v : \rho)\ \alpha\ .P$ is well-formed under the assumption that the host ambient has local view Γ, if P is well-formed under the assumption that the host ambient has local view $\Gamma \oplus \rho^{\uparrow v}$. Thus, the prefix $\mathsf{inC}(v : \rho)\ \alpha$ allows its host ambient to extend its local knowledge and hence be ready to communicate with arbitrary ambients willing to enter. Note that this prefix binds

Table 6. Well-formed Processes

$$\frac{\Gamma\ ok}{\Sigma \vdash_c \mathbf{0} : \Gamma}\ (\mathbf{0}) \qquad \frac{\Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c !P : \Gamma}\ (\text{PROC-REP}) \qquad \frac{\Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c (\boldsymbol{\nu} n)P : \Gamma}\ (\text{PROC-RES})$$

$$\frac{\Sigma \vdash_c P_1 : \Gamma \quad \Sigma \vdash_c P_2 : \Gamma}{\Sigma \vdash_c P_1 \mid P_2 : \Gamma}\ (\text{PROC-COMP}) \qquad \frac{\Sigma \vdash C : \mathsf{cap} \quad \Sigma \vdash_c P : \Gamma}{\Sigma \vdash_c C.P : \Gamma}\ (\text{PROC-CAP})$$

$$\frac{\Sigma \vdash_c P : \Gamma \oplus \rho^{\uparrow v} \quad \pi \in \{\mathsf{inC}(v : \rho)\ \alpha\ , \mathsf{outC}(v : \rho)\ \alpha\ \}}{\Sigma \vdash_c \pi.P : \Gamma}\ (\text{PROC-INC-OUTC})$$

$$\frac{\Sigma \vdash_c P : \Gamma \oplus \rho^{\downarrow v} \quad \pi \in \{\overline{\mathsf{inC}}(v : \rho), \overline{\mathsf{outC}}(v : \rho)\}}{\Sigma \vdash_c \pi.P : \Gamma}\ (\text{PROC-COINC-COOUTC})$$

$$\frac{\Sigma, x_1 : \varphi_1, \ldots, x_k : \varphi_k \vdash_c P : \Gamma \quad (\varphi_1 \times \ldots \times \varphi_k)^\eta \in \Gamma}{\Sigma \vdash_c (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^\eta.P : \Gamma}\ (\text{PROC-INPUT})$$

$$\frac{\Sigma \vdash_c P : \Gamma \quad \Sigma \vdash M_i : \varphi_i\ (1 \leq i \leq k) \quad (\varphi_1 \times \ldots \times \varphi_k)^\eta \in \Gamma}{\Sigma \vdash_c \langle M_1, \ldots, M_k \rangle^\eta.P : \Gamma}\ (\text{PROC-OUTPUT})$$

$$\frac{\Sigma \vdash_{c'} P : \Gamma' \quad \Sigma \vdash \beta : \mathsf{amb} \quad \uparrow c(\Gamma') \preceq \downarrow c'(\Gamma) \quad \Gamma' \text{ is closed} \quad \Gamma \text{ is } ok}{\Sigma \vdash_c \beta[\Gamma' \| c' \| P] : \Gamma}\ (\text{PROC-AMB})$$

the free occurrences of the port variable v in P. The typing of the other prefixes mentioned in rules (PROC-INC-OUTC) and (PROC-COINC-COOUTC) is similar. The difference between these two rules is that in the first one the process shall communicate with a new host ambient, whereas in the second one the process shall communicate with a newly entering child ambient.

The (PROC-INPUT) and (PROC-OUTPUT) request that the type of the information that is exchanged together with its location must belong to the local view of the host ambient.

The (PROC-AMB) rule may be interpreted as follows. In order for $\beta[\Gamma' \| c' \| P]$ to be considered a well-formed process under a host ambient whose communication interface consists of a port c and a local view Γ, it must be satisfied that:

1. process P is well-formed under a host ambient whose communication interface consists of port c' and local view Γ', where β must be an ambient name or an ambient variable;
2. either $\beta[\Gamma' \| c' \| P]$ does not communicate with its host ambient or the type of the information exchange between it and its host ambient must be the same (condition $\uparrow c(\Gamma') \preceq \downarrow c'(\Gamma)$);
3. no free port variables should occur in Γ', i.e. Γ' should be *closed;*
4. the local view Γ of the host ambient must be *ok.*

The example considered on page 129 shows why we do not allow $\beta[\Gamma' \| c' \| P]$ to offer a communication to its host ambient when the host ambient does not communicate along the port c'. In fact, the process

$$p[\emptyset \| c \| \overline{\mathsf{inC}}(v : \mathsf{amb}).\langle q \rangle^{\downarrow v} \mid m[\mathsf{cap}^{\uparrow c} \| c' \| (x : \mathsf{cap})^{\uparrow c}.P]] \mid n[\emptyset \| c' \| \mathsf{inC}(v' : \mathsf{amb})\ p\ ]$$

is typable by replacing the condition $\uparrow c'(\Gamma) \preceq \downarrow c(\Gamma')$ to $\uparrow c(\Gamma') \preceq \downarrow c'(\Gamma)$ in rule (PROC-AMB).

The type system guarantees that communication inside ambients and across ambient boundaries never leads to type mismatches. This is formalized as:

Theorem 1 (Subject Reduction). *If* $\Sigma \vdash_c P : \Gamma$ *and* $P \longrightarrow Q$, *then* $\Sigma \vdash_c Q : \Gamma$.

4 Behavioral Semantics

In order to study the behavioral semantics of **BACI** we define an intuitive notion of barbed congruence [25,17] based on the unlabelled reduction semantics given in Table 4. We then introduce a labelled transition semantics inspired by [20, 22,7] and state that it coincides with unlabelled reduction. Finally, we define a notion of labelled bisimilarity and show that it is sound with respect to barbed congruence. The immediate benefit is that the co-inductive nature of bisimilarity can be exploited by putting the vast body of proof techniques to work in order to reason about barbed congruence. Note that in this short presentation we omit the global environment Σ, the host port c and the local view Γ over which the relations on well-formed processes are indexed by.

Since **BACI** has co-capabilities and allows parent-child communications there are several reasonable choices of barbs, among which we have:

$$P\downarrow_n^1 \triangleq P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \overline{\mathsf{in}}.Q \mid R] \mid S) \quad (1)$$

$$P\downarrow_n^2 \triangleq P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \overline{\mathsf{inC}}(v : \rho).Q \mid R] \mid S) \quad (2)$$

$$P\downarrow_{\langle c,c' \rangle}^3 \triangleq P \equiv (\nu \tilde{m})(n[\Gamma \| c \| (x_1 : \varphi_1, \ldots, x_k : \varphi_k)^{\uparrow c'}.Q \mid R] \mid S) \quad (3)$$

$$P\downarrow_{\langle c,c' \rangle}^4 \triangleq P \equiv (\nu \tilde{m})(n[\Gamma \| c \| \langle M_1, \ldots, M_k \rangle^{\uparrow c'}.Q \mid R] \mid S) \quad (4)$$

provided that $n \notin \tilde{m}$ in (1) and (2). In order to *observe* whether a process P may interact with the environment via some ambient name n or via a pair of port names $\langle c, c' \rangle$, it must be placed in a context that presents an ambient that attempts to enter it or to communicate with it.

We write $P \Downarrow_n$ ($P \Downarrow_{\langle c,c' \rangle}$) if $P \Longrightarrow P'$ and $P' \downarrow_n$ ($P \downarrow_{\langle c,c' \rangle}$), where $\Longrightarrow$ is the reflexive and transitive closure of $\longrightarrow$.

The notions of observational congruence induced by the above definitions of barb are standard in ambient calculi.

Definition 1. *A relation $\mathcal{R}$ is reduction closed if $P \mathcal{R} Q$ and $P \longrightarrow P'$ imply the existence of some Q' such that $Q \Longrightarrow Q'$ and $P' \mathcal{R} Q'$. $\mathcal{R}$ is barb preserving if $P \mathcal{R} Q$ and $P \downarrow_n$ ($P \downarrow_{\langle c,c' \rangle}$) imply $Q \Downarrow_n$ ($Q \Downarrow_{\langle c,c' \rangle}$).*

Definition 2 (Reduction Barbed Congruence). *Reduction barbed congruence is the largest equivalence relation that is preserved by contexts and, when restricted to closed processes, is reduction closed and barb preserving. Let then $\cong_i$ be the reduction barbed congruence relation from choosing the notion of observation as in (i) above (with $i \in [1..4]$).*

Notice that since we only consider processes which are well-formed, a relation $\mathcal{R}$ is preserved by contexts if $P \mathcal{R} Q$ and $C[P]$ well-formed imply $C[Q]$ well-formed and $C[P] \mathcal{R} C[Q]$, for all processes P, Q and contexts $C[\cdot]$.

As expected the above congruencies coincide, so we can denote barbed congruence for **BACI** simply by $\cong$.

Lemma 1 (Independence from barbs). $\cong_i = \cong_j$ *for all* $i, j \in [1..4]$.

Proof. We need to show that all barbs imply each other. This can be accomplished, as usual, by exhibiting a corresponding context. For instance, to see that $\cong_2$ implies $\cong_3$ use the context $C[\cdot] = m[\Gamma \| c' \| [\cdot] \mid \langle \tilde{M} \rangle^{\downarrow c}.\overline{\mathsf{inC}}(v : \rho)]$, and note that for all P such that m is fresh in P one has $P \Downarrow^3_{\langle c,c' \rangle}$ if and only if $C[P] \Downarrow^2_m$. A suitable context to show that $\cong_4$ implies $\cong_1$ is $C[\cdot] = p[\Gamma \| c \| (\nu q)(q[\emptyset \| c_q \| [\cdot] \mid m[\emptyset \| c_m \| \mathsf{in}\ n.\mathsf{out}\ n.\mathsf{out}\ q] \mid \overline{\mathsf{out}}]) \mid \overline{\mathsf{out}}.\langle \tilde{M} \rangle^{\uparrow c'}]$, and similarly for the other cases.

Notice that processes with different types can be distinguished irrespective of their purely behavioural properties. This means that if two processes P and Q cannot be typed with the same Γ (w.r.t. a given c), they cannot be congruent. In fact, if $\Sigma \vdash_c P : \Gamma$ but $\Sigma \vdash_c Q : \Gamma$ does not hold we can find a context $C[\cdot]$ such that $C[P]$ is a (well-formed) process while $C[Q]$ is not. A suitable context is simply $n[\Gamma' \| c \| \pi.[\cdot]]$, where Γ' is the maximum subset of Γ which is closed, and π contains (in any order) exactly the set of actions $\{\mathsf{inC}(v : \rho)\ m \mid \rho^{\uparrow v}\}$ and the set of co-actions $\{\overline{\mathsf{inC}}(v : \rho) \mid \rho^{\downarrow v}\}$. For that reason, type equality is not required *per se* for the definition of barbed congruence.

4.1 Algebraic Laws

This section presents some algebraic laws that better portray the semantics of processes in **BACI.** These and other laws can be proved by means of the labelled bisimilarity developed in the next section.

The laws holding in **BACI** which deal with mobility are very similar to those true for the NBA calculus [7], so we will not discuss them.

Instead **BACI**'s refined treatment of communication using port names allows to get quite interesting laws concerning input-output. For example, an ambient only willing to communicate with its father but using a "wrong" port name is dead, i.e. we have the following *garbage collection laws:*

$$n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| (\tilde{x} : \tilde{\varphi})^{\uparrow c}.P] \mid Q] \cong n[\Gamma_n \| c_n \| Q]$$

$$n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| \langle \tilde{M} \rangle^{\uparrow c}.P] \mid Q] \cong n[\Gamma_n \| c_n \| Q]$$

In NBA a communication parent-child can be forced only if it is the only active process inside both ambients. In **BACI** instead there can be other active processes provided that they do not know the port name of the communication partner and some ambient names do not occur in some processes and/or they are restricted. The conditions on port names avoid interfering communications and the conditions on ambient names avoid interfering movements. In particular in the third group of equivalencies R cannot contain m since otherwise an ambient inside R could exit m and communicate the port c_m to process S. More precisely we have:

if c_m does not occur in R

$$(\nu n)(n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| \langle \tilde{M} \rangle^{\uparrow c_n}.P] \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q \mid R])$$
$$\cong$$
$$(\nu n)(n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| P] \mid Q\{\tilde{x} := \tilde{M}\} \mid R])$$

$$(\nu n)(n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P] \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q \mid R])$$
$$\cong$$
$$(\nu n)(n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| P\{\tilde{x} := \tilde{M}\}] \mid Q \mid R])$$

if c_n and n do not occur in R

$$n[\Gamma_n \| c_n \| (\nu m)(m[\Gamma_m \| c_m \| \langle \tilde{M} \rangle^{\uparrow c_n}.P \mid R]) \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q]$$
$$\cong$$
$$(\nu m)(n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| P \mid R]) \mid Q\{\tilde{x} := \tilde{M}\}]$$

$$n[\Gamma_n \| c_n \| (\nu m)(m[\Gamma_m \| c_m \| (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P \mid R]) \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q]$$
$$\cong$$
$$n[\Gamma_n \| c_n \| (\nu m)(m[\Gamma_m \| c_m \| P\{\tilde{x} := \tilde{M}\} \mid R]) \mid Q]$$

Table 7. Labels and Outcomes

$$
\begin{array}{lll}
\text{Pre-prefixes} & \kappa & ::= \mathsf{in}\ n \mid \mathsf{out}\ n \mid \overline{\mathsf{in}} \mid \mathsf{inC}(c:\rho)\ n \mid \mathsf{outC}(c:\rho)\ n \mid \overline{\mathsf{inC}}(c:\rho) \\
\text{Prefixes} & \mu & ::= \kappa \mid \langle \tilde{M} \rangle^{\eta} \mid \overline{\mathsf{outC}}(c:\rho) \mid \overline{\mathsf{out}} \\
\text{Labels} & \xi & ::= \tau \mid \mu \mid \langle \tilde{-} \rangle^{\eta} \mid n[\Gamma_n \| c_n \| \kappa] \mid \tilde{M}\ \mathsf{get}\ (c_m, c_n) \mid \mathsf{put}\ (c_m, c_n) \\
\text{Concretions} & K & ::= (\boldsymbol{\nu}\tilde{p})\langle\!\langle P \rangle\!\rangle Q \mid (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P \\
\text{Outcomes} & O & ::= P \mid K
\end{array}
$$

if c_m, m do not occur in S and c_n, n, m do not occur in R

$$
\begin{gathered}
(\boldsymbol{\nu}n)(n[\Gamma_n \| c_n \| (\boldsymbol{\nu}m)(m[\Gamma_m \| c_m \| \langle \tilde{M} \rangle^{\uparrow c_n}.P \mid R]) \mid (\tilde{x} : \tilde{\varphi})^{\downarrow c_m}.Q \mid S]) \\
\cong \\
(\boldsymbol{\nu}n)(\boldsymbol{\nu}m)(n[\Gamma_n \| c_n \| m[\Gamma_m \| c_m \| P \mid R]) \mid Q\{\tilde{x} := \tilde{M}\} \mid S]) \\
(\boldsymbol{\nu}n)(n[\Gamma_n \| c_n \| (\boldsymbol{\nu}m)(m[\Gamma_m \| c_m \| (\tilde{x} : \tilde{\varphi})^{\uparrow c_n}.P \mid R]) \mid \langle \tilde{M} \rangle^{\downarrow c_m}.Q \mid S]) \\
\cong \\
(\boldsymbol{\nu}n)(n[\Gamma_n \| c_n \| (\boldsymbol{\nu}m)(m[\Gamma_m \| c_m \| P\{\tilde{x} := \tilde{M}\} \mid R]) \mid Q \mid S])
\end{gathered}
$$

4.2 Labelled Transition Semantics

This section presents a labelled transition semantics (LTS) and proves that it coincides with reduction. It is the first step towards a characterization of reduction barbed congruence in terms of labelled bisimulation. The LTS is given in Tables 8, 9 and 10. These tables define the labelled transition relation

$$P \xrightarrow{\xi} O$$

where P is a process, ξ is a label and O is an "outcome". Labels and outcomes are defined in Table 7.

An outcome may be a process or a *concretion.* Concretions are required for dealing with transitions of components of the system that interact with the environment in order to be completed. Indeed, they prove convenient for formulating the silent transitions. In the concretion $(\boldsymbol{\nu}\tilde{p})\langle\!\langle P \rangle\!\rangle Q$, the process P is the part of the system that interacts with the environment. For example, to complete an in n transition, the sibling ambient which hosts the entering one must be requested from the context. Likewise, in the concretion $(\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle Q$, the message $\tilde{M}$ is the part of the system that interacts with the environment. This outcome is required only for the case of the transition for message output. In both cases, Q represents the remaining part of the process that is not affected by the transition.

Table 8. Commitments: Visible transitions

$$\frac{M \in \{\mathsf{in}\ n, \mathsf{out}\ n, \overline{\mathsf{in}}, \overline{\mathsf{out}}\}}{M.P \xrightarrow{M} P}\ (\textsc{cap}) \qquad \frac{M_1.(M_2.P) \xrightarrow{\xi} P'}{(M_1.M_2).P \xrightarrow{\xi} P'}\ (\textsc{path})$$

$$\frac{\zeta \in \{\mathsf{inC}, \mathsf{outC}\}}{\zeta(v:\rho)\ n\ .P \xrightarrow{\zeta(c:\rho)^n} P\{\uparrow v := \uparrow c\}}\ (\textsc{capC}) \qquad \frac{\zeta \in \{\overline{\mathsf{inC}}, \overline{\mathsf{outC}}\}}{\zeta(v:\rho).P \xrightarrow{\zeta(c:\rho)} P\{\downarrow v := \downarrow c\}}\ (\textsc{co-capC})$$

$$\frac{P \xrightarrow{\xi} P' \quad \xi \in \{\mathsf{in}\ n, \mathsf{out}\ n\}}{m[\Gamma_m \| c_m \| P] \xrightarrow{m[\Gamma_m \| c_m \| \xi]} \langle\!\langle m[\Gamma_m \| c_m \| P'] \rangle\!\rangle \mathbf{0}}\ (\textsc{in-out}) \qquad \frac{P \xrightarrow{\xi} P' \quad \xi \in \{\overline{\mathsf{in}}, \overline{\mathsf{inC}}(c:\rho)\}}{m[\Gamma_m \| c_m \| P] \xrightarrow{m[\Gamma_m \| c_m \| \xi]} \langle\!\langle P' \rangle\!\rangle \mathbf{0}}\ (\textsc{co-in})$$

$$\frac{P \xrightarrow{\xi} P' \quad \xi \in \{\mathsf{inC}(c:\rho)\ n, \mathsf{outC}(c:\rho)\ n\} \quad \Gamma_m \oplus \rho^{\uparrow c}\ ok}{m[\Gamma_m \| c_m \| P] \xrightarrow{m[\Gamma_m \| c_m \| \xi]} \langle\!\langle m[\Gamma_m \oplus \rho^{\uparrow c} \| c_m \| P'] \rangle\!\rangle \mathbf{0}}\ (\textsc{inC-outC})$$

$$\frac{}{(\tilde{x}:\tilde{\varphi})^\eta.P \xrightarrow{(\tilde{M})^\eta} P\{\tilde{x} := \tilde{M}\}}\ (\textsc{input}) \qquad \frac{}{\langle \tilde{M} \rangle^\eta.P \xrightarrow{\langle - \rangle^\eta} \langle\!\langle \tilde{M} \rangle\!\rangle P}\ (\textsc{output})$$

$$\frac{P \xrightarrow{(\tilde{M})^{\uparrow c}} P_1}{m[\Gamma_m \| c_m \| P] \xrightarrow{\tilde{M}\ \mathsf{get}\ (c_m, c)} m[\Gamma_m \| c_m \| P_1]}\ (\textsc{get})$$

$$\frac{P \xrightarrow{\langle - \rangle^{\uparrow c}} (\nu \tilde{p}) \langle\!\langle \tilde{M} \rangle\!\rangle P_1}{m[\Gamma_m \| c_m \| P] \xrightarrow{\mathsf{put}\ (c_m, c)} (\nu \tilde{p}) \langle\!\langle \tilde{M} \rangle\!\rangle m[\Gamma_m \| c_m \| P_1]}\ (\textsc{put})$$

The structural congruence relation for concretions is obtained by extending the homonymous relation for processes with the following axioms and rules:

$$\begin{array}{lcl} (\nu r)((\nu \tilde{p}) \langle\!\langle P \rangle\!\rangle Q) & \equiv & (\nu r, \tilde{p}) \langle\!\langle P \rangle\!\rangle Q \\ (\nu r)((\nu \tilde{p}) \langle\!\langle \tilde{M} \rangle\!\rangle P) & \equiv & (\nu r, \tilde{p}) \langle\!\langle \tilde{M} \rangle\!\rangle P \end{array}$$

$$\begin{array}{lcl} P \equiv P' \text{ and } Q \equiv Q' & \Longrightarrow & (\nu \tilde{p}) \langle\!\langle P \rangle\!\rangle Q \equiv (\nu \tilde{p}) \langle\!\langle P' \rangle\!\rangle Q' \\ P \equiv P' & \Longrightarrow & (\nu \tilde{p}) \langle\!\langle \tilde{M} \rangle\!\rangle P \equiv (\nu \tilde{p}) \langle\!\langle \tilde{M} \rangle\!\rangle P' \end{array}$$

Also, we use the following notational conventions:

- $((\nu\tilde{p})\langle\!\langle P \rangle\!\rangle Q) \mid R = (\nu\tilde{p})\langle\!\langle P \rangle\!\rangle (Q \mid R)$
- $((\nu\tilde{p})\langle\!\langle M \rangle\!\rangle P) \mid R = (\nu\tilde{p})\langle\!\langle M \rangle\!\rangle (P \mid R)$

The transitions are inspired by those of NBA [7]. The τ transitions for message exchanges are (τ-EXCHANGE) for local exchange and (τ-PUT) and (τ-GET) for non-local exchange. For example, in (τ-PUT) the directed output action towards the child ambient must be met by a corresponding input action from the child. Rule (GET) makes sure that this input action is executed inside some ambient whose local communication port coincides with the one specified in the output action.

The τ transitions for mobility are (τ-ENTER), (τ-ENTERC), (τ-EXIT), (τ-EXITC). Since these are similar in spirit we shall discuss only (τ-ENTERC). Rule (τ-ENTERC) is in charge of synchronizing two actions, namely the request of an ambient to enter a host ambient with the action witnessing the approval (by means of an appropriate co-capability) on the part of the host ambient. Therefore, the label of the first action is $n[\Gamma_n \| c_n \| \mathsf{inC}(c_m : \rho)\ m\ ]$ while that of the second is $m[\Gamma_m \| c_m \| \overline{\mathsf{inC}}(c_n : \rho')]$. The former tells of the name and local interface information of the moving ambient and the latter does the same for the host ambient. The process that actually moves is represented by P_1 in the concretion resulting from the first action while Q_1 represents the process that shall run alongside the visiting ambient. The processes P_2 and Q_2 are the subcomponents of P and Q, respectively, that do not participate in the movement. Note that the third premise of the rule (τ-ENTERC) corresponds to the dynamic type checking that we discussed for reduction.

As expected, unlabelled reduction and labelled reduction coincide. Both items of Theorem 2 are proved by induction on the derivation of the antecedent. Moreover, item 2 requires the following lemma that relates labelled reduction and structural congruence, for the case when the derivation is obtained using the rule (STRUCT).

Lemma 2. *If* $P \xrightarrow{\xi} O$ *and* $P \equiv Q$, *then there exists* O' *such that* $Q \xrightarrow{\xi} O'$ *and* $O \equiv O'$.

Theorem 2.

1. *If* $P \xrightarrow{\tau} P'$, *then* $P \longrightarrow P'$.
2. *If* $P \longrightarrow P'$, *then* $P \xrightarrow{\tau} \equiv P'$, *where* $\xrightarrow{\tau} \equiv$ *denotes the composition of the relations* $\xrightarrow{\tau}$ *and* $\equiv$.

By comparing the notion of observability (cf. the definition of barbs) with the rules of Table 8 and in particular with rule (CO-IN) one can easily see that a name is observable iff at least one of the two actions $n[\Gamma \| c \| \overline{\mathsf{in}}]$ or $n[\Gamma \| c \| \overline{\mathsf{inC}}(c' : \rho)]$ can be performed. In particular,

Lemma 3. $P \downarrow_n^1$ *iff* $P \xrightarrow{n[\Gamma \| c \| \overline{\mathsf{in}}]} (\nu\tilde{p})\langle\!\langle Q \rangle\!\rangle R$ *for some* Γ, c, $\tilde{p}$, Q, R.

A similar observation applies to rules (GET), (PUT) and the observability of pairs of port names (cf. barbs (3) and (4) above). Thanks to Lemma 1 we only need to consider one notion of barb.

Table 9. Commitments: τ transitions

$$(\tau\text{-ENTER})\quad \frac{P \xrightarrow{n[\Gamma_n \| c_n \| \mathsf{in}\ m]} (\nu\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad Q \xrightarrow{m[\Gamma_m \| c_m \| \overline{\mathsf{in}}]} (\nu\tilde{q})\langle\!\langle Q_1 \rangle\!\rangle Q_2 \quad \uparrow c_m(\Gamma_n) \preceq \downarrow c_n(\Gamma_m)}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p},\tilde{q})(m[\Gamma_m \| c_m \| Q_1 \mid P_1] \mid P_2 \mid Q_2)}$$

$$(\tau\text{-ENTERC})\quad \frac{P \xrightarrow{n[\Gamma_n \| c_n \| \mathsf{inC}(c_m:\rho)\ m\ ]} (\nu\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad Q \xrightarrow{m[\Gamma_m \| c_m \| \overline{\mathsf{inC}}(c_n:\rho')]} (\nu\tilde{q})\langle\!\langle Q_1 \rangle\!\rangle Q_2 \quad \uparrow c_m(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_m) \sqcup \rho'}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p},\tilde{q})(m[\Gamma_m \oplus \rho'^{\downarrow c_n} \| c_m \| Q_1 \mid P_1] \mid P_2 \mid Q_2)}$$

$$(\tau\text{-EXIT})\quad \frac{P \xrightarrow{n[\Gamma_n \| c_n \| \mathsf{out}\ m]} (\nu\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad Q \xrightarrow{\overline{\mathsf{out}}} Q_1 \quad \uparrow c_q(\Gamma_n) \preceq \downarrow c_n(\Gamma_q)}{q[\Gamma_q \| c_q \| m[\Gamma_m \| c_m \| P] \mid Q] \xrightarrow{\tau} (\nu\tilde{p})(q[\Gamma_q \| c_q \| m[\Gamma_m \| c_m \| P_2] \mid P_1 \mid Q_1])}$$

$$(\tau\text{-EXITC})\quad \frac{P \xrightarrow{n[\Gamma_n \| c_n \| \mathsf{outC}(c_q:\rho)\ m\ ]} (\nu\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad Q \xrightarrow{\overline{\mathsf{outC}}(c_n:\rho')} Q_1 \quad \uparrow c_q(\Gamma_n) \sqcup \rho \preceq \downarrow c_n(\Gamma_q) \sqcup \rho'}{q[\Gamma_q \| c_q \| m[\Gamma_m \| c_m \| P] \mid Q] \xrightarrow{\tau} (\nu\tilde{p})(q[\Gamma_q \oplus \rho'^{\downarrow c_n} \| c_q \| m[\Gamma_m \| c_m \| P_2] \mid P_1 \mid Q_1])}$$

$$(\tau\text{-EXCHANGE})\quad \frac{P \xrightarrow{(\tilde{M})^\star} P_1 \quad Q \xrightarrow{\langle - \rangle^\star} (\nu\tilde{q})\langle\!\langle \tilde{M} \rangle\!\rangle Q_1}{P \mid Q \xrightarrow{\tau} (\nu\tilde{q})(P_1 \mid Q_1)}$$

$$(\tau\text{-PUT})\quad \frac{P \xrightarrow{\langle - \rangle^{\downarrow c_m}} (\nu\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P_1 \quad Q \xrightarrow{\tilde{M}\ \mathsf{get}\ (c_n,c_m)} Q_1}{n[\Gamma_n \| c_n \| P \mid Q] \xrightarrow{\tau} (\nu\tilde{p})n[\Gamma_n \| c_n \| P_1 \mid Q_1]}$$

$$(\tau\text{-GET})\quad \frac{P \xrightarrow{(\tilde{M})^{\downarrow c_m}} P_1 \quad Q \xrightarrow{\mathsf{put}\ (c_m,c_n)} (\nu\tilde{q})\langle\!\langle \tilde{M} \rangle\!\rangle Q_1}{n[\Gamma_n \| c_n \| P \mid Q] \xrightarrow{\tau} (\nu\tilde{q})(n[\Gamma_n \| c_n \| P_1 \mid Q_1])}$$

4.3 Full Bismilarity and Its Soundness

This section defines a notion of labelled bisimilarity and shows that it is sound with respect to reduction barbed congruence. Labelled bisimilarity requires checking when two processes produce equal observable actions. The problem is that the current definition of labelled reduction may produce a concretion instead of a process. This situation is remedied by introducing *higher-order (HO)*

Table 10. Commitments: Structural transitions

$$\text{(PAR)}\quad \frac{P \xrightarrow{\xi} O}{P \mid Q \xrightarrow{\xi} O \mid Q} \qquad \text{(RES)}\quad \frac{P \xrightarrow{\xi} O \quad n \notin \mathrm{fn}(\xi)}{(\nu n)P \xrightarrow{\xi} (\nu n)O}$$

$$(\tau\text{-AMB})\quad \frac{P \xrightarrow{\tau} P'}{n[\Gamma\|c\|P] \xrightarrow{\tau} n[\Gamma\|c\|P']} \qquad \text{(REPL)}\quad \frac{\pi.P \xrightarrow{\xi} O}{!\pi.P \xrightarrow{\xi} !\pi.P \mid O}$$

transitions [22] for those labelled transitions of Table 8 that produce a concretion as an outcome.

The HO-transitions are given in Table 11. In these transitions we use richer labels obtained by adding to the previous labels ξ a new component which can be of one of the following five shapes:

- P;
- $[\Gamma\|c\|P]$;
- $n[\Gamma\|c\|P]$;
- $[\Gamma\|c\|P]n[\Gamma'\|c'\|Q]$;
- $n[\Gamma\|c\|P] \mid Q$.

This component describes the minimum contribution of the context necessary to fire the transition. For example in rule (HO OUT) the context must provide both the 3 components (local view, port and process) of the ambient n from which the process P_1 exits and in which the process P_2 remains and the whole ambient q in which the process P_1 enters.

For HO transitions we get the following version of Lemma 3:

Lemma 4. *$P \downarrow^1_n$ iff $P \xrightarrow{n[\Gamma\|c\|\overline{\mathrm{in}}]m[\emptyset\|c'\|\mathbf{0}]} (\nu\tilde{p})(n[\Gamma\|c\|Q \mid m[\emptyset\|c'\|\mathbf{0}]] \mid R)$ for all m, c' and for some Γ, c, $\tilde{p}$, Q, R.*

As last step towards defining labelled bisimilarity, let Λ denote the set of labels that includes both the first order labels defined in Tables 8 and 10 and the HO ones of Table 11. In the following notational convention we let λ range over Λ. Let $\Longrightarrow$ denote the reflexive and transitive closure of $\xrightarrow{\tau}$.

1. $\stackrel{\lambda}{\Longrightarrow}$ denotes $\Longrightarrow\xrightarrow{\lambda}\Longrightarrow$.
2. $\stackrel{\hat{\lambda}}{\Longrightarrow}$ denotes $\Longrightarrow$ if $\lambda = \tau$ and $\stackrel{\lambda}{\Longrightarrow}$ otherwise.

Definition 3 (Bisimulation). *A symmetric relation $\mathcal{R}$ over closed processes is a bisimulation if $P\mathcal{R}Q$ and $P \xrightarrow{\lambda} P'$ imply there exists Q' such that*

Table 11. Commitments: Higher-Order transitions

(HO IN)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{in}\ n]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{in}\ n][\Gamma_n \| c_n \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid Q] \mid P_2)}$$

(HO INC)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{inC}(c_n:\rho)\ n\ ]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \sqcup \rho \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{inC}(c_n:\rho)\ n\ ][\Gamma_n \| c_n \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid Q] \mid P_2)}$$

(HO CO-IN)

$$\frac{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{in}}]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{in}}] m[\Gamma_m \| c_m \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid m[\Gamma_m \| c_m \| Q]] \mid P_2)}$$

(HO CO-INC)

$$\frac{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{inC}}(c_m:\rho)]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \preceq \downarrow c_m(\Gamma_n) \sqcup \rho}{P \xrightarrow{n[\Gamma_n \| c_n \| \overline{\mathsf{inC}}(c_m:\rho)] m[\Gamma_m \| c_m \| Q]} (\boldsymbol{\nu}\tilde{p})(n[\Gamma_n \| c_n \| P_1 \mid m[\Gamma_m \| c_m \| Q]] \mid P_2)}$$

(HO OUT)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{out}\ n]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_q(\Gamma_m) \preceq \downarrow c_m(\Gamma_q)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{out}\ n][\Gamma_n \| c_n \| Q] q[\Gamma_q \| c_q \| R]} (\boldsymbol{\nu}\tilde{p})(q[\Gamma_q \| c_q \| P_1 \mid n[\Gamma_n \| c_n \| P_2 \mid Q] \mid R])}$$

(HO OUTC)

$$\frac{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{outC}(c_n:\rho)\ n\ ]} (\boldsymbol{\nu}\tilde{p})\langle\!\langle P_1 \rangle\!\rangle P_2 \quad \uparrow c_n(\Gamma_m) \sqcup \rho \preceq \downarrow c_m(\Gamma_n)}{P \xrightarrow{m[\Gamma_m \| c_m \| \mathsf{outC}(c_n:\rho)\ n\ ][\Gamma_n \| c_n \| Q] q[\Gamma_q \| c_q \| R]} (\boldsymbol{\nu}\tilde{p})(q[\Gamma_q \| c_q \| P_1 \mid n[\Gamma_n \| c_n \| P_2 \mid Q] \mid R])}$$

(HO OUTPUT)

$$\frac{P \xrightarrow{\langle - \rangle^{\eta}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P' \quad \eta \in \{\star, \downarrow c\}}{P \xrightarrow{\langle - \rangle^{\eta} Q} (\boldsymbol{\nu}\tilde{p})(P' \mid Q\{\tilde{x} := \tilde{M}\})}$$

(HO PUT)

$$\frac{P \xrightarrow{\mathsf{put}\ (c_m, c_n)} (\boldsymbol{\nu}\tilde{q})\langle\!\langle \tilde{M} \rangle\!\rangle P'}{P \xrightarrow{\mathsf{put}\ (c_m, c_n) Q} (\boldsymbol{\nu}\tilde{q})(P' \mid Q\{\tilde{x} := \tilde{M}\})}$$

(HO OUTPUT$^{\uparrow}$)

$$\frac{P \xrightarrow{\langle - \rangle^{\uparrow c}} (\boldsymbol{\nu}\tilde{p})\langle\!\langle \tilde{M} \rangle\!\rangle P'}{P \xrightarrow{\langle - \rangle^{\uparrow c} m[\Gamma_m \| c_m \| R] \mid Q} (\boldsymbol{\nu}\tilde{p})(m[\Gamma_m \| c_m \| P' \mid R] \mid Q\{\tilde{x} := \tilde{M}\})}$$

– $Q \stackrel{\lambda}{\Longrightarrow} Q'$ *and*
– $P' \mathcal{R} Q'$.

Two closed processes P and Q are bisimilar, written $P \approx_c Q$, *if* $P \mathcal{R} Q$ *for some bisimulation* $\mathcal{R}$.

The definition of bisimulation is extended to arbitrary processes as usual:

Definition 4 (Full Bisimilarity). *Two processes P and Q are fully bisimilar, written* $P \approx Q$, *if* $P\mathbf{s} \approx_c Q\mathbf{s}$ *for every closing substitution* $\mathbf{s}$ *that respects types.*

Following the proof scheme of [22,7] we can show that full bisimilarity is preserved by context.

Theorem 3. *Full bisimilarity is a congruence.*

Moreover from Lemma 4 it follows that:

Lemma 5. *Full bisimilarity is barb preserving over closed processes.*

Proof. Suppose P, Q are closed processes, $P \approx_c Q$ and $P \downarrow_n^1$.
By Lemma 4 $P \stackrel{n[\Gamma \| c \| \overline{\text{in}} m[\emptyset \| c' \| \mathbf{0}]}{\longrightarrow} P'$ for all m, c' and some Γ, c, P'. As a consequence $Q \stackrel{n[\Gamma \| c \| \overline{\text{in}} m[\emptyset \| c' \| \mathbf{0}]}{\Longrightarrow} Q'$ for some Q'. In particular, there is a Q'' such that $Q \Longrightarrow Q'' \stackrel{n[\Gamma \| c \| \overline{\text{in}} m[\emptyset \| c' \| \mathbf{0}]}{\longrightarrow} Q'$. From Lemma 4 we deduce $Q'' \downarrow_n^1$ and hence $Q \Downarrow_n^1$, as required.

Finally, we prove the desired result that $\approx$ is contained in $\cong$.

Theorem 4 (Soundness of Full Bisimilarity). *If* $P \approx Q$ *then* $P \cong Q$.

Proof. It suffices to show that $\approx$ is a barbed bisimulation up to $\equiv$ (since then it follows that $\equiv\approx\equiv$ - ie. the composition of the relations $\equiv$, $\approx$ and $\equiv$ - is also a barbed bisimulation and $P \approx Q$ and $\equiv\approx\equiv \subseteq \cong$ imply $P \cong Q$). This follows from the fact that $\approx$:

1. is a congruence: Theorem 3.
2. is reduction closed on closed processes: Suppose P, Q are closed processes, $P \approx Q$ and $P \longrightarrow P'$. By Theorem 2, $P \stackrel{\tau}{\longrightarrow} \equiv P'$. Since $P \approx Q$, there exists Q' such that $Q \Longrightarrow Q'$ and $P' \equiv\approx\equiv Q'$.
3. is barb preserving on closed processes: Lemma 5.

We conjecture incompleteness of $\approx$ for the same reason the authors of [7] conjecture incompleteness of the full bisimilarity arising from a similar LTS for NBA, namely the difficulty of finding a context which discriminates the label $\langle \tilde{M} \rangle^{\uparrow c}$. We conjecture also that a LTS for **BACI** inducing a complete full bisimilarity could be developed in the style of [7].

5 Examples

In this section we sketch some examples in order to show the expressiveness of **BACI.** Before doing so, we define the following auxiliary notation to make the examples easier to read.

$$\alpha^{\rightleftharpoons}[\Gamma \| c \| P] \triangleq \alpha[\Gamma \| c \| !\overline{\mathsf{in}} \mid !\overline{\mathsf{out}} \mid P]$$

This allows sibling and nested ambients of α to freely enter and exit. Note that $\alpha^{\rightleftharpoons}$ allows to enter either ambients which do not communicate with α or ambients whose communication port name is already known by $\alpha^{\rightleftharpoons}$.

We convene not to write the types of the input variables since they are always clear from the context.

5.1 Remote Printer

For this example we consider two networks (represented as ambients) called $n1$ and $n2$. Ambient $n1$ is the network where a client is located and $n2$ the one where a printer is located. Although the client ignores the path to the printer network, in $n1$ there is also a *router,* called $r1to2$, that knows the path to $n2$. For simplicity, we place $n1$ and $n2$ at the same nesting level inside a larger ambient, called *inter.* However, in general, $n1$ and $n2$ can be far from each other within the nesting hierarchies.

$$INTERNET \triangleq inter^{\rightleftharpoons}[\emptyset \| c \| N1 \mid N2]$$

$$N1 \triangleq n1^{\rightleftharpoons}[\emptyset \| c_1 \| CLIENT \mid ROUTER]$$

$$N2 \triangleq n2^{\rightleftharpoons}[\emptyset \| c_2 \| PRINTER]$$

The idea is that the client sends a print *job* to *PRINTER* via *ROUTER.* A *job* ambient should receive two parameters (data and printer name) from *CLIENT* after releasing the job. After receiving the parameters, the job exits the client and enter *ROUTER.* There, it shall receive the path to $n2$, where the printer is located. After reaching $n2$, the job enters the printer and communicate the data to be printed.

$$JOB_{cl} \triangleq job[\Gamma_{job} \| c_j \| (d,p)^{\uparrow c_{cl}}.\mathsf{out}\ cl.\mathsf{in}\ r1to2.(route)^{\uparrow c_r}.route.\mathsf{in}\ p.\langle d \rangle^{\uparrow c_{pr}}]$$
$$\text{where } \Gamma_{job} \triangleq \{(\mathsf{data} \times \mathsf{amb})^{\uparrow c_{cl}}, \mathsf{cap}^{\uparrow c_r}, \mathsf{data}^{\uparrow c_{pr}}\}$$

Notice that the *job* ambient is able to communicate with different parent ports in different TOCs. Here, c_j is the port of the job, c_{cl} is the port of the client, c_r is the port of the router and c_{pr} is the port of the printer.

CLIENT spawns the job and sends the data to be printed using the job ambient. Then, the job is received by *ROUTER* which gives the job the route to $n2$. Finally, the job enters *PRINTER* and delivers its data.

$$CLIENT \triangleq client1^{\rightleftharpoons}[\Gamma_{client1} \| c_{cl} \| \langle (d1, printer1) \rangle^{\downarrow c_j} \mid !\ JOB_{client1}]$$
$$\text{where } \Gamma_{client1} \triangleq \{(\mathsf{data} \times \mathsf{amb})^{\downarrow c_j}\}$$

$$ROUTER \triangleq r1to2 \rightleftharpoons [\mathsf{cap}^{\downarrow c_j} \| c_r \| !\langle(\mathsf{out}\ r1to2.\mathsf{out}\ n1.\mathsf{in}\ n2)\rangle^{\downarrow c_j}]$$

$$PRINTER \triangleq printer1 \rightleftharpoons [\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\,(d)^{\downarrow c_j}]$$

After delivering its data the *job* ambient becomes inactive and useless. Using the algebraic properties we can show that $job[\Gamma_{job} \| c_j \| \mathbf{0}] \cong \mathbf{0}$ and therefore

$$printer1 \rightleftharpoons [\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\,(d)^{\downarrow c_j} \mid job[\Gamma_{job} \| c_j \| \mathbf{0}]] \cong printer1 \rightleftharpoons [\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\,(d)^{\downarrow c_j}]$$

All these *garbage* ambients that accumulate inside the printer ambient can be safely discarded.

Since different ambients can have the same port name, more than one client can have port name c_{cl} and more than one server the port name c_{pr}, even if the ambients have different names. Moreover, we can add more clients and printers without changing *JOB* or *ROUTER*.

$$N1' \triangleq n1 \rightleftharpoons [\emptyset \| c_1 \| CLIENT \mid CLIENT' \mid ROUTER]$$

$$N2' \triangleq n2 \rightleftharpoons [\emptyset \| c_2 \| PRINTER \mid PRINTER']$$

$$CLIENT' \triangleq client2 \rightleftharpoons [\Gamma_{client2} \| c_{cl} \| \langle(d2, printer2)\rangle^{\downarrow c_j} \mid !\ JOB_{client2}]$$
where $\Gamma_{client2} \triangleq \{(\mathsf{data} \times \mathsf{amb})^{\downarrow c_j}\}$

$$PRINTER' \triangleq printer2 \rightleftharpoons [\mathsf{data}^{\downarrow c_j} \| c_{pr} \| !\,(d)^{\downarrow c_j}]$$

Having c_{cl} as a port name for all clients and c_{pr} for all servers allows any client to use any available printer, and not just a particular one as in the previous example. *CLIENT'* can also use *PRINTER'* by sending the message $\langle(d2, printer2)\rangle^{\downarrow c_j}$ to the spawned job.

The routing in the previous example was relatively simple with only one destination, only one route and only one router. How can we route a job to two different networks, for instance, $n2$ and $n3$? Here, we can get the destination network parameter from the client and use it to find the corresponding route. However, we need some mechanism to determine if we choose the route for $n2$ or the route for $n3$ depending on that parameter. There are no control flow primitives in the calculus similar to the test for equality found in π-calculus, for instance. Nevertheless, we can instruct the client to send the name of the router serving a given printer network (assuming we have a different router for each destination network), but the client would need to know the relation between the destination and the router that serves that destination. That is not very tidy. Besides, we would need to change the job interface, which seems to be very "natural" as it is. Another option is to take advantage of the locality of names and use the same name for both the destination network and the router serving the route to that destination. We now re-define the components of the system according to these new requirements:

$$INTERNET \triangleq inter \rightleftharpoons [\emptyset \| c_i \| N1" \mid N2" \mid N3]$$

$$N1" \triangleq n1 \rightleftharpoons [\emptyset \| c_1 \| CLIENT \mid CLIENT' \mid ROUTER \mid ROUTER']$$

$N2'' \triangleq n2[\emptyset \| c_2 \| PRINTER]$

$N3 \triangleq n3[\emptyset \| c_3 \| PRINTER']$

We moved one of the printers to network $n3$ to make the example more interesting. With this setting, any client located on $n1$ should be able to send a job to either the printer on $n2$ or the printer on $n3$. For this purpose, we change the name of the existing router (routing jobs to $n2$) and also add a new router that serves jobs heading to $n3$. Both routers have the same name as the routes they serve. Using the same names, we don't need to require the client to know the name of the routers. This gives us a clean and natural representation. We have to change the definition of *JOB* and the *CLIENTs* since in the previous example the router name was "hard-coded" and now is a parameter given by the client.

$JOB_{cl} \triangleq job[\Gamma_{job} \| c_j \| (d, p, n)^{\uparrow c_{cl}}.\mathsf{out}\ cl.\mathsf{in}\ n.(route)^{\uparrow c_r}.route.\mathsf{in}\ p.\langle data \rangle^{\uparrow c_{pr}}]$
where $\Gamma_{job} \triangleq \{(\mathsf{data} \times \mathsf{amb} \times \mathsf{amb})^{\uparrow c_{cl}}, \mathsf{cap}^{\uparrow c_r}, \mathsf{data}^{\uparrow c_{pr}}\}$

$CLIENT \triangleq client1[\Gamma_{client1} \| c_{cl} \| \langle (d1, printer1, n2) \rangle^{\downarrow c_j} \mid !\ JOB_{client1}]$
where $\Gamma_{client1} \triangleq \{(\mathsf{data} \times \mathsf{amb} \times \mathsf{amb})^{\downarrow c_j}\}$

$CLIENT' \triangleq client2[\Gamma_{client2} \| c_{cl} \| \langle (d2, printer2, n3) \rangle^{\downarrow c_j} \mid !\ JOB_{client2}]$
where $\Gamma_{client2} \triangleq \{(\mathsf{data} \times \mathsf{amb} \times \mathsf{amb})^{\downarrow c_j}\}$

Finally, we change the name of the routers and we add the new router which has the same structure as the old one but with different route, of course.

$ROUTER \triangleq n2[\mathsf{cap}^{\downarrow c_j} \| c_r \| !\langle (\mathsf{out}\ n2.\mathsf{out}\ n1.\mathsf{in}\ n2) \rangle^{\downarrow c_j}]$

$ROUTER' \triangleq n3[\mathsf{cap}^{\downarrow c_j} \| c_r \| !\langle (\mathsf{out}\ n3.\mathsf{out}\ n1.\mathsf{in}\ n3) \rangle^{\downarrow c_j}]$

The two orthogonal concepts of interfaces and names allow us to separate the input/output from the mobility concerns. We can use the interfaces to group several ambients with similar input/output abilities and, at the same time, we can keep each ambient identity by using different ambient names for each of them.

5.2 File Servers Cluster

This example represents some free download sites in which the user has a list of servers to choose for his download. However, for this example, we require that every time a customer requests a file download, the cluster designates one server from all the available servers in the cluster (i.e. all the servers that are not serving other clients) to serve that request. Additionally, we want a cluster administrator to be able to execute some administrative operations like shutdown or power up any particular server. For this reason, we assign a unique and distinctive name to each server. However, we use a common port name and interface for all of them to allow the cluster to communicate with all of them.

$CLUSTER \triangleq cluster[\Gamma_{clu} \| c_{clu} \| LOAD_BAL \mid SRV1 \mid SRV2]$
where $\Gamma_{clu} \triangleq \{\mathsf{amb} \times \mathsf{Filename}^{\downarrow c_{srv}}\}$

$$LOAD_BAL \triangleq !(\overline{\mathsf{inC}}(v_{cl} : (\mathsf{amb} \times \mathsf{Filename})).(clname, fn)^{\downarrow v_{cl}}.\langle clname, fn\rangle^{\downarrow c_{srv}})$$

The *cluster* includes all the servers (only two in this example) and the *load balancing* mechanism. This mechanism allows a client to enter the cluster: the cluster receives the client's request that it forwards to any available server. Notice that the client's communication port is not known in advance to the cluster and vice-versa. They are learned on the ENTER reduction, where the port names replace the variables bound by inC and $\overline{\mathsf{inC}}$. Each server has two main sub-processes: the service itself and the *power management* process. The *SERVE* process receives the forwarded request from the cluster ambient, and then it responds spawning a messenger ambient called *job.* This job reaches the client and deliver the requested file. The acute reader will notice that, before receiving a request, *SERVE* waits for an "on" message from the power management ambient called *pwr.* The *pwr* ambient is used to inform the serving process that the server is still on. We now show how to use this feature to "shutdown" a server.

$$SRVi \triangleq srvi\rightleftharpoons[\Gamma_{srvi} \| c_{srv} \| !(on)^{\downarrow c_{pwr}}.SERVE \mid PWR]$$
$$\text{where } \Gamma_{srvi} \triangleq \{\mathsf{onMsg}^{\downarrow c_{pwr}}, \mathsf{amb} \times \mathsf{Filename}^{\uparrow c_{clu}}\}$$

$$SERVE \triangleq (clname, fname)^{\uparrow c_{clu}}.JOB$$

$$JOB \triangleq job[\emptyset \| c_j \| \mathsf{out}\ srvi.\mathsf{inC}(v : \mathsf{data})\ clname\ .(\mathtt{file}(fname))^{\uparrow v}]$$

$$PWR \triangleq pwr[\mathsf{onMsg}^{\uparrow c_{srv}} \| c_{pwr} \| !\langle on\rangle^{\uparrow c_{srv}} \mid \mathsf{in}\ pwroff\,]$$

The purpose of *pwr* is simple. If it is present inside a server, it enables the service by continuously sending "on" messages. However, if it is not present, the server is not able to listen (and respond) to a request. Therefore, in order to shut a server down, the administrator should send a *POWER_OFF* message to that server.

$$POWER_OFF(s) \triangleq pwroff\rightleftharpoons[\emptyset \| c_{poff} \| \mathsf{in}\ cluster.\mathsf{in}\ s.\overline{\mathsf{in}}]$$

The *pwr* ambient would be locked inside *pwroff* after entering that ambient. Once inside *pwroff, pwr is* rendered inoperative. In fact, using algebraic properties we can show that

$$pwroff\rightleftharpoons[\emptyset \| c_{poff} \| pwr[\mathsf{onMsg}^{\uparrow c_{srv}} \| c_{pwr} \| !\langle on\rangle^{\uparrow c_{srv}}]] \cong \mathbf{0}$$

and get rid of these *garbage* ambients.
Likewise, the administrator can restore the *pwr* ambient inside the server to "power on" that server.

$$POWER_ON(s) \triangleq pwron\rightleftharpoons[\emptyset \| c_{pon} \| \mathsf{in}\ cluster.\mathsf{in}\ s.TURN_ON]$$

$$TURN_ON \triangleq pwr[\mathsf{onMsg}^{\uparrow c_{srv}} \| c_{pwr} \| \mathsf{out}\ pwron \mid !\langle on\rangle^{\uparrow c_{srv}} \mid \mathsf{in}\ poweroff\,]$$

Finally, we present a "generic" client. The clients are generic in the sense that they do not need to know any of the port names in advance, all of them are learned on execution. The only requirement is that the client is well behaved and

it sends its own name in the request. A malicious client could send a different name. However, this can only cause a response to be lost or sent to the wrong client, which is unlikely since the malicious client needs to guess a correct client name.

$$CLIENT \triangleq client^{\rightleftharpoons}[\Gamma_{cl} \| c_{client} \| \mathsf{inC}(v_{clu} : (\mathsf{amb} \times \mathsf{Filename}))\ cluster\ .\ \langle client, afilename \rangle.\overline{\mathsf{inC}}(c_j : \mathsf{data}).(\mathsf{file})^{\downarrow c_j}.\mathsf{P} \mid \mathsf{Q}]$$

This is the basic structure of a client ambient. The port name can be changed without restrictions. Γ_{cl} , P and Q can be anything that does not have conflicting types with the *cluster* and *job* ambients . The whole configuration looks like this:

$$SYSTEM \triangleq ADMIN \mid CLUSTER \mid CLIENTS$$

The *ADMIN* process could include processes like those in the power management and the *CLIENTS* are also placed (initially) outside the cluster. As we have seen, they need to enter the cluster to get served.

6 Conclusions

We have presented a typed calculus of mobile ambients that features both local and dynamic typing. Each ambient comes equipped with a local communication interface consisting of a communication port and a local view indicating the type of the information that may be exchanged over parent and children ports. Besides the usual communication within an ambient, messages may be exchanged across ambient boundaries. The type system guarantees that in this case the types of the local ports of the sending and receiving ambients agree. Since communication interfaces are local and ambients may migrate, ambients must be able to increase their local knowledge of their surroundings. Therefore, the mobility rules allow an ambient to learn the communication type of the local port which it enters. Appropriate run-time checks are required so that the entering and the host ambient agree on a topic of conversation. Among the novel aspects of **BACI** are:

- *Communicating ports.* In contrast with previous ambient calculi, **BACI** uses names for mobility and ports for communication.
- *Named communication with parents.* While in previous calculi communication with a parent was decided by the location of an ambient, in **BACI,** the communication with a parent is indexed by the parent's port, in a similar way in which communication with a child is usually indexed. This new named communication allows an ambient to communicate with different parents in different types (TOCs).
- *Finer control of non-determinism.* The division between names and ports introduces the ability to have non-determinism for mobility and determinism for communication and vice-versa, while in previous calculi, that was not possible.

– *Local typing.* Having different TOCs with different parents allows control over which parent can exchange information, while in previous calculi the type of a communication with the parent remained fixed.

Although communication control is local this is not so for mobility. Mobility is currently unrestricted and this poses the question if one might also include, in the local knowledge of an ambient, some indication of whether the ambient is allowed to move or not. Other items that warrant further work include: considering a restriction operator on port names, considering multiple ports (possibly taking dynamic port creation into account), matching and mismatching constructs and group types in order to impose access control.

References

1. Torben Amtoft, Assaf J. Kfoury, and Santiago M. Pericas-Geertsen. What are Polymorphically-Typed Ambients? In David Sands, editor, *ESOP'01,* volume 2028 of *LNCS,* pages 206–220, Berlin, 2001. Springer-Verlag.
2. Torben Amtoft, Henning Makholm, and Joe B. Wells. PolyA: True Type Polymorphism for Mobile Ambients. In *TCS'04,* 2004. to appear.
3. H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics.* Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, revised edition, 1984.
4. Gérard Boudol. A Parametric Model of Migration and Mobility, Release 1. Mikado Deliverable D1.2.1, available at http://mikado.di.fc.ul.pt/repository/D1.2.1.pdf, 2003.
5. Michele Bugliesi and Giuseppe Castagna. Behavioral Typing for Safe Ambients. *Computer Languages,* 28(1):61 – 99, 2002.
6. Michele Bugliesi, Giuseppe Castagna, and Silvia Crafa. Access Control for Mobile Agents: The Calculus of Boxed Ambients. *ACM Transactions on Programming Languages and Systems,* 26(1):57–124, 2004.
7. Michele Bugliesi, Silvia Crafa, Massimo Merro, and Vladimiro Sassone. Communication and Mobility Control in Boxed Ambients. To appear in *Information and Computation.* Extended and revised version of M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication Interference in Mobile Boxed Ambients. In FSTTCS'02, volume 2556 of LNCS, pages 71-84. Springer-Verlag, 2002.
8. Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Mobility Types for Mobile Ambients. In Jiri Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP'99,* volume 1644 of *LNCS,* pages 230–239, Berlin, 1999. Springer-Verlag.
9. Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Ambient Groups and Mobility Types. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, Peter D. Mosses, and Takayasu Ito, editors, *TCS'00,* volume 1872 of *LNCS,* pages 333–347, Berlin, 2000. Springer-Verlag. Extended version to appear in Information and Computation, special issue on TCS'00.
10. Luca Cardelli and Andrew D. Gordon. Mobile Ambients. *Theoretical Computer Science,* 240(1): 177–213, 2000. Special Issue on Coordination, Daniel Le Métayer Editor.
11. Giuseppe Castagna and Jan Vitek. Seal: A Framework for Secure Mobile Computations. In Henri E. Bal, Boumediene Belkhouche, and Luca Cardelli, editors, *Internet Programming Languages,* volume 1686 of *LNCS,* pages 47–77, Berlin, 1999. Springer-Verlag.

12. Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Rosario Pugliese. Dynamic and Local Typing for Mobile Ambients. In *TCS'04,* 2004. to appear.
13. Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Ivano Salvo. M3: Mobility Types for Mobile Processes in Mobile Ambients. In James Harland, editor, *CATS'03,* volume 78 of *ENTCS.* Elsevier, 2003.
14. Rocco De Nicola, GianLuigi Ferrari, and Rosario Pugliese. Klaim: a Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering,* 24(5):315–330, 1998.
15. Elio Giovannetti. Ambient Calculi with Types: a Tutorial. In Corrado Priami, editor, *Global Computing - Programming Environments, Languages, Security and Analysis of Systems,* volume 2874 of *LNCS,* Berlin, 2003. Springer-Verlag.
16. Healfdene Goguen. Typed operational semantics. In Mariangiola Dezani-Ciancaglini and Gordon Plotkin, editors, *TLCA'95,* volume 902 of *LNCS,* pages 186–200, Berlin, 1995. Springer-Verlag.
17. Andrew D. Gordon and Luca Cardelli. Equational properties of mobile ambients. *Mathematical Structures in Computer Science,* 13(3):371–408, 2003.
18. Mattew Hennessy and James Riely. Resource Access Control in Systems of Mobile Agents. *Information and Computation,* 173:82–120, 2002.
19. Matthew Hennessy, Massimo Merro, and Julian Rathke. Towards a behavioural theory of access and mobility control in distributed system (extended abstract). In Andrew D. Gordon, editor, *FOSSACS'03,* volume 2620 of *LNCS,* pages 282–299, Berlin, 2003. Springer-Verlag.
20. Francesca Levi and Davide Sangiorgi. Controlling Interference in Ambients. *Transactions on Programming Languages and Systems,* 25(1):1–69, 2003.
21. Cédric Lhoussaine and Vladimiro Sassone. A Dependently Typed Ambient Calculus. In David Schmidt, editor, *ESOP'04,* volume 2986 of *LNCS,* pages 171–187, Berlin, 2004. Springer-Verlag.
22. Massimo Merro and Matthew Hennessy. Bisimulation Congruences in Safe Ambients. In Neil D. Jones and Xavier Leroy, editors, *POPL'02,* pages 71–80, New York, 2002. ACM Press.
23. Massimo Merro and Vladimiro Sassone. Typing and Subtyping Mobility in Boxed Ambients. In Lubos Brim, Petr Jancar, Mojmir Kretinsky, and Antonin Kucera, editors, *CONCUR'02,* volume 2421 of *LNCS,* pages 304–320, Berlin, 2002. Springer-Verlag.
24. George C. Necula. Proof-carrying code. In Neil D. Jones, editor, *POPL'97,* pages 106–119. ACM Press, 1997.
25. Davide Sangiorgi and Robin Milner. The problem of "Weak Bisimulation up to". In Walter R. Cleaveland, editor, *CONCUR'92,* volume 630 of *LNCS,* pages 32–46, Berlin, 1992. Springer-Verlag.

Algebraic Recognizability of Languages*

Pascal Weil

LaBRI, CNRS and Université Bordeaux-1
pascal.weil@labri.fr

Abstract. Recognizable languages of finite *words* are part of every computer science cursus, and they are routinely described as a cornerstone for applications and for theory. We would like to briefly explore why that is, and how this word-related notion extends to more complex models, such as those developed for modeling distributed or timed behaviors.

Ἐν ἀρχῇ ἦν ὁ λόγος...

In the beginning was the Word...

Recognizable languages of finite *words* are part of every computer science cursus, and they are routinely described as a cornerstone for applications and for theory. We would like to briefly explore why that is, and how this word-related notion extends to more complex models, such as those developed for modeling distributed or timed behaviors.

The notion of recognizable languages is a familiar one, associated with classical theorems by Kleene, Myhill, Nerode, Elgot, Büchi, Schützenberger, etc. It can be approached from several angles: recognizability by automata, recognizability by finite monoids or finite-index congruences, rational expressions, monadic second order definability. These concepts are expressively equivalent, and this leads to a great many fundamental algorithms in the fields of compilation, text processing, software engineering, etc... Moreover, it surely indicates that the class of recognizable languages is central. These equivalence results use the specific structure of words (finite chains, labeled by the letters of the alphabet), and the monoid structure of the set of all words.

Since the beginnings of language theory, there has been an interest for other models than words – especially for the purpose of modeling distributed or timed computation (trees, traces, pomsets, graphs, timed words, etc.) –, and for extending to these models the tools that were developped for words. For many models, some of these tools may not be defined, and those who are defined, may not coincide.

In this paper, we concentrate on the algebraic notion of recognizability: that which, for finite words, exploits the monoid structure of the set of words, and relies on the consideration of monoid morphisms into finite monoids, or equivalently, of finite-index monoid congruences. Our aim is to examine why this

* This paper was prepared in part while the author was an invited Professor at the University of Nebraska-Lincoln. He also acknowledges partial support from project VERSYDIS (*ACI Sécurité Informatique,* Ministère de la Recherche).

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 149–175, 2004.
© Springer-Verlag Berlin Heidelberg 2004

particular approach is fruitful in the finite word case, and how it has been, or can be adapted to other models.

In Sect. 1, we explore the specific benefits of using algebraic recognizability for the study of word languages. It opens the door to a very fine classification of recognizable languages, which uses the resources of the structural theory of finite monoids. This classification of recognizable languages is not only mathematically elegant, it also allows the characterization and the decision of membership in otherwise significant classes of languages.

The emblematic example of such a class is that of star-free languages, which are exactly the first-order definable languages, those that can be defined by a formula in propositional temporal logic, those that are recognized by a counter-free automaton, and those whose syntactic monoid is aperiodic (Schützenberger, McNaughton, Pappert, Kamp). Only the latter two characterizations lead to a decision algorithm, and the algebraic approach makes this algorithm the clearest.

The example of star-free languages is however not the only one; for example, the various notions of locally testable languages are also characterized, and ultimately efficiently decided, by algebraic properties (Simon, McNaughton, Ladner). The power of finite monoid theory leads in fact to an extremely fine classification (Eilenberg), where for instance, natural hierarchies within first-order or temporal logic can be characterized as well (Brzozowski, Knast, Thomas, Thérien, Wilke).

Already in the 1960s, fundamental results appeared on notions of automata for trees and for infinite words, linking them with logical definability and rational expressions (Büchi, Doner, Mezei, Thatcher, Wright). An algebraic approach to automata-recognizable languages of infinite word was introduced in the early 1990s, and as in the case of tree languages, it requires introducing an algebraic framework different from monoid theory (Perrin, Pin, Wilke), see Sect. 4.1.

In fact, very early on, Elgot and Mezei extended the notion of recognizability to subsets of arbitrary (abstract) algebras. But the notion of logical definability for these subsets strongly depends on the combinatorial (relational) structure of the objects chosen to represent the elements of the abstract algebra under consideration. In many situations, the problem is posed in the other direction: we know which models we want to consider (they are posets, or graphs, or traces, or timed words, as arise from, say, the consideration of distributed or timed computation) and we need to identify an algebraic structure on the set of these objects, for which logical definability and algebraic recognizability will be best related. One key objective there, is to be able to decide logical specifications. Note that models of automata, while highly desirable, are not known to exist in all the interesting cases, and especially not for graphs or posets. In contrast, the algebraic and the logical points of view are universal.

A class of relational structures being fixed, Courcelle gave very loose conditions on an algebraic structure on the set of these structures, which guarantee that counting monadic second order definability implies recognizability. The converse is known to hold in a number of significant cases, but not in general.

In Sect. 4, we discuss some of the relational structures that have been studied in the literature, and the *definability vs. recognizability* results that are known for them: trees, infinite words, traces, series-parallel pomsets, message sequence charts and layered diagrams, graphs, etc. The main obstacle to the equivalence between definability and recognizability is the fact that the algebras we consider may not be finitely generated. In contrast, a good number of situations have been identified where this equivalence holds, and each time, a finiteness (or boundedness) condition is satisfied. We will try to systematically point out these finiteness conditions, and we will identify some important questions that are still pending.

When all is said and done, the central question is that of the specification and the analysis of infinite sets by finite means (finite and finitely generated algebras, finite automata, logical formulas, etc.), arguably the fundamental challenge of theoretical computer science. This paper presents a personal view on the relevance of algebraic recognizability for this purpose, beyond its original scope of application (languages of finite words), and an introduction to some of the literature and results that illustrate this view. I do not claim however that it constitutes a comprehensive survey of the said literature and results (in particular, I chose to systematically refer to books and survey papers when available), and I apologize in advance for any omission!...

1 The Finite Word Case

In the beginning were the (finite) words, and loosely following the Biblical analogy, one could say that the spirits of Kleene, Büchi and Schützenberger flew over the abyss, organising it from chaos to beauty.

Throughout this paper, A will denote an alphabet, i.e. a finite, non-empty set. We denote by A^* the set of all finite words on alphabet A.

1.1 The Classical Equivalence Results

We are all familiar with the notion of regular languages, but there are in fact several competing notions, that turn out to be equivalent for finite words. Each is interesting in its own right, as it reveals a fruitful point of view, syntactic or semantic, denotational or operational. The results of this section can be found in many books, and in particular in those of Eilenberg [33], Pin [67,68,69], Straubing [77], Sipser [75] and Sakarovitch [73].

Recognizability by Automata. One can first consider languages *recognized by finite state automata,* whether deterministic or non-deterministic. Every language recognized by a finite state automaton admits a unique *minimal deterministic automaton,* which is effectively computable.

The notion of a deterministic automaton can also be expressed in terms of a finite-index semi-congruence, and in terms of an action of the free monoid on a finite set.

Algebraic Recognizability. One can also consider languages *recognized by a finite monoid.* This exploits the monoid structure of A^*, the set of all words on alphabet A: if M is a monoid and $\varphi: A^* \to M$ is a morphism of monoids, we say that $L \subseteq A^*$ is recognized by φ (or by M) if $L = P\varphi^{-1}$ for some $P \subseteq M$, or equivalently, if $L = L\varphi\varphi^{-1}$. Here too, for every language L recognized by a finite monoid, there exists a least finite monoid recognizing L, called the *syntactic monoid* of L.

Rational Expressions. *Rational expressions* describe languages using the letters of the alphabet, the constant $\emptyset$, and the so-called *rational operations* of union, concatenation and Kleene star (if L is a language, L^* is the submonoid of A^* generated by L).

It should be noted that for a given rational language, there is no notion of a unique minimal rational expression describing it.

Logical Definability. *Büchi's sequential calculus* exploits the combinatorial structure of words, as A-labeled, linearly ordered finite sets: in this logical formalism, individual variables are interpreted to be positions in a word, and the predicates are $i < j$ (to say that position i is to the left of position j) and $R_a i$ (to say that position i is labeled by letter $a \in A$). *First order formulas* (FO) use only individual variables, whereas *monadic second order formulas* (MSO) also use second order variables, interpreted to be sets of positions. To a formula φ in this language, one associates the set $L(\varphi)$ of all words which satisfy φ, that is $L(\varphi)$ is the language of the finite models of φ.

The Kleene-Nerode-Myhill-Büchi Theorem. Theorems by Kleene, Nerode, Myhill and Büchi show that the notions rapidly described above coincide.

Theorem 1. *Let $L \subseteq A^*$. Then L is recognized by a finite state automaton, if and only if L is recognized by a finite monoid, if and only if the syntactic monoid of L is finite, if and only if L is described by a rational expression, if and only if L is defined by an MSO-formula of Büchi's calculus.*

Moreover, there are algorithms to pass from one of these specification formalisms to each other.

It is interesting to note that the many closure properties of the class of recognizable languages are easily established in an appropriate choice of one of these equivalent formalisms. For instance, closure under Boolean operations easily follows from the definition of algebraic recognizability, as does closure under inverse morphism (inverse rewriting). On the other hand, closure under concatenation, star and direct morphism is a triviality for languages described by rational expressions.

None of the equivalences in Theorem 1 is very difficult, but their proofs really use the different points of view on words and languages. If we compare these results with the situation that prevails for other models than words, it is

in fact a very exceptional situation to have these notions be so nicely defined and be equivalent.

1.2 Classification of Recognizable Languages

With each recognizable language L, we can associate a computable canonical finite object – in fact two closely related such objects: the minimal automaton of L, and its syntactic monoid. The connection between them is tight: the syntactic monoid of L is exactly the monoid of transitions of its minimal automaton.

This paves the way for a fine classification of recognizable languages (see [67,69]). Not surprisingly, it is the syntactic monoid, with its natural algebraic structure, which offers the strongest classification tool. In this section, we give a few instances of this classification: some are well-known and open up important applications (star-free languages, locally testable languages), some are more specific, and demonstrate the degree of refinement allowed by this method.

Star-Free Languages. The most illuminating example of a significant subclass of recognizable languages is given by the *star-free languages.* These are the languages which can be described by star-free expressions, *i.e.,* using the letters of the alphabets, the constants $\emptyset$ and 1 (the empty word), the Boolean operations and concatenation (but no star).

The characterization of star-free languages requires the following definitions: a deterministic automaton is said to be *counter-free* if whenever a non-trivial power u^n $(u \neq 1,\ n \neq 0)$ labels a loop at some state q, then the word u also labels a loop at the same state. A finite monoid M is said to be *aperiodic* if it contains no non-trivial group, if and only if for each $x \in M$, $x^n = x^{n+1}$ for all large enough n. Finally, PTL *(propositional temporal logic)* is a modal logic, interpreted on positions in words, with modalities next, eventually and until.

The following statement combines results by Schützenberger, McNaughton, Pappert and Kamp, see [33,49,67,77].

Theorem 2. *Let $L \subseteq A^*$. Then L is star-free, if and only if L is recognized by a finite aperiodic monoid, if and only if the syntactic monoid of L is finite and aperiodic, if and only if L is recognized by a counter-free automaton, if and only if L is defined by an FO-formula of Büchi's calculus, if and only if L is defined by a PTL-formula.*

Moreover, there are algorithms to pass from one of these specification formalisms to each other.

Thus, the class of star-free languages, with its natural definition in terms of generalized rational expressions, ends up having natural characterizations in terms of all the formalisms used in Theorem 1 – to which we can add PTL, a logical formalism considered to be very useful to specify the behavior of complex systems.

The historically first side of this result is the algebraic one, which links star-free languages and aperiodic monoids. It is of particular interest for two reasons.

First because it offers an algorithm to decide whether a language is star-free; and second, because it shows that the algebraic structure of the syntactic monoid of a recognizable language (not just its finiteness) reflects the combinatorial properties of that language. This gave the first hint of Eilenberg's theorem, discussed further in this section.

Variants of FO-Definability. Several refinements and generalizations further reinforce the significance of Theorem 2.

Consider the extension FO+MOD of FO, where we also allow *modulo quantification* of the form $\exists^{\text{mod } q} x\ \varphi(x)$ $(q \geq 1)$. Such a quantification is interpreted to mean that the set of values x for which $\varphi(x)$ holds, has cardinality a multiple of q. Straubing, Thérien and Thomas showed that a language is FO+MOD-definable if and only if the subgroups of its syntactic monoid are solvable [77].

Considering now subclasses of FO, it turns out that every star-free language can be defined by a FO-formula using only 3 variables. Let FO_2 be the class of star-free languages defined by FO-formulas with only 2 variables. Let also **DA** be the class of finite monoids in which every regular element is idempotent (if $xyx = x$ for some y, then $x^2 = x$). Then a combination of results of Etessami, Pin, Schützenberger, Thérien, Vardi, Weil, Wilke [74,70,37,80] shows that a language L is in FO_2, if and only if it is defined by a Σ_2- and by a Π_2-formula, if and only if its syntactic monoid is in **DA**, if and only if L is defined by a PTL formula which does not use the until modality, if and only if L can be obtained from the letters using only disjoint unions and unambiguous products (and the constants $\emptyset$ and A^*).

It is well-known that every FO-formula is equivalent to one in prenex normal form (consisting of a sequence of quantifications, followed by a quantifier-free formula). This gives rise to the classical *quantifier-alternation hierarchy,* based on counting the number of alternated blocks of existential and universal quantifiers. Another natural hierarchy, seen from the point of view of star-free expressions, defines its n+1-st level as the Boolean closure of products of level n languages (and level 0 consists of $\emptyset$ and A^*). This is the so-called *dot-depth hierarchy.* Thomas showed that these two hierarchies coincide, that is, a language can be defined by an FO-formula in prenex normal form with n alternating blocks of quantifiers, if and only if it is in the n-th level of the dot-depth hierarchy [83]. Decidable algebraic characterizations were given for level 1 of these hierarchies, but the decidability of level 2 and the further levels is still an open question. It was however showed (Brzozowski, Knast, Simon, Straubing, see [68]) that the hierarchy is infinite (if $|A| \geq 2$), and that each level is characterized by an algebraic property, in the following sense: if two languages have the same syntactic monoid and one is at level n, then so is the other one.

There is also a natural hierarchy on PTL-formulas, based on the number of nested usage of the until modality. Thérien and Wilke showed that the levels of this infinite hierarchy are characterized by the algebraic properties of the syntactic monoid, and that each is decidable [81].

Communication complexity. The *communication complexity* of a language L is a measure of the amount of communication that is necessary for two partners, each holding part of a word, to determine whether the word lies in L, see [51]. Tesson and Thérien showed that the communication complexity of a recognizable language is entirely determined by its syntactic monoid, and that it can be computed on this basis [78].

Piecewise and Locally Testable Languages. A word v is a *subword* of a word u if $v = a_1 \cdots a_n$ and $u = u_0 a_1 u_1 \cdots a_n u_n$ for some $u_0, \ldots, u_n \in A^*$. It is a *factor* of u if $u = xvy$ for some $x, y \in A^*$.

A language L is said to be n-piecewise testable if whenever u and v have the same subwords of length at most n and $u \in L$, then $v \in L$. The language L is *piecewise testable* if it is n-piecewise testable for some n.

A language L is said to be n-locally testable if whenever u and v have the same factors of length at most n and the same prefix and suffix of length $n-1$, and $u \in L$, then $v \in L$. The language L is *locally testable* if it is n-locally testable for some n. Locally testable languages are widely used in the fields of learning and pattern matching, whereas piecewise testable languages form the first level of the dot-depth hierarchy.

Results of Simon and McNaughton, Ladner (see [33,67,69]) show that both these properties are characterized by algebraic properties of syntactic monoids. More precisely, a language L is piecewise testable if and only if every principal two-sided ideal of its syntactic monoid $S(L)$, admits a single generator. The language L is locally testable if and only if $S(L)$ is aperiodic and $eS(L)e$ is an idempotent commutative monoid, for each idempotent $e \neq 1$ in $S(L)$.

Varieties of Languages. Many more examples can be found in the literature, where natural algebraic properties of finite monoids match natural combinatorial or logical properties of languages (see for instance [1,33,67,69]). The scope of this matching is described in Eilenberg's variety theorem; the latter identifies the closure properties on classes of recognizable languages and on classes of finite monoids, that characterize the classes that can occur in this correspondence. These classes are called, respectively, *varieties of recognizable languages* and *pseudovarieties of finite monoids.*

Decision Procedures. The varieties of recognizable languages thus identified by algebraic means are all the more interesting if they are decidable. Since the syntactic monoid of a recognizable language is computable, this reduces to deciding the membership of a finite monoid in certain pseudovarieties. In fact, in the examples surveyed above, this is the only path known to a decision algorithm.

Let us now assume that we are considering a decidable pseudovariety of monoids (and hence a decidable variety of languages). The syntactic monoid of a language L, which is the transition monoid of the minimal automaton of L, may have a size exponential in the number of states of that automaton. Thus

deciding whether a recognizable language given by a deterministic finite state automaton lies in a given variety, seems to require exponential time and space.

In view of the connection between syntactic monoid and minimal automaton, it is possible to translate the relevant algebraic property of finite monoids to a property of automata, and to check this property on the minimal automaton. This possibility is explicitly stated in Theorem 2, but it is also the underlying reason for the decision procedures concerning piecewise testable (Stern [76]) and locally testable languages (Kim, McNaughton, McCloskey [50]).

In many important situations, this leads to polynomial time membership algorithms: piecewise and locally testable languages, FO_2, certain varieties related to the dot-depth hierarchy [70], etc. One major exception though, is the class of star-free languages, for which the membership problem is PSPACE-complete (Cho, Huynh [14]). In other words, given a deterministic automaton, there is no fundamentally better algorithm to decide whether the corresponding language is star-free, than to verify whether the syntactic monoid is aperiodic.

It must be stressed that even in the cases where we have polynomial membership algorithms, these algorithms are a translation to automata of algebraic properties of the syntactic monoid, they were not discovered until after the corresponding pseudovariety of monoids was identified, and their natural justification is *via* monoid-theoretic considerations.

1.3 Recognizable and Context-Free Languages

Recognizable languages form but the lowest level of the Chomsky hierarchy, where the next level consists of the context-free languages. *Context-free languages* are defined by context-free grammars, which can be viewed, with a more algebraic mindset, as finite systems of polynomial equations of the form $x_i = \sum p(\boldsymbol{x})$ $(1 \leq i \leq n)$ where $\boldsymbol{x} = (x_1, \ldots, x_n)$ is the vector of variables, the summations are finite and each p is a word over the letters of A and the variables (see [11]). A solution of such a system is a vector of languages $\boldsymbol{L} = (L_1, \ldots, L_n)$, and the context-free languages arise as the components of maximal solutions of such systems. Accordingly, context-free languages are also called *equational,* or *algebraic.*

Recognizable languages are components of maximal solutions of certain simpler systems, where each $p(\boldsymbol{x})$ is a word of the form $x_j u$, $1 \leq j \leq n$, $u \in A^*$ (right-linear equation). In particular, not all context-free languages are recognizable. The class of context-free languages is not closed under intersection, but it is closed under intersection with recognizable languages.

2 Almost as Established: The Finite Tree Case

Tree languages were considered in the early 1960s, see [39]. Here we use a *ranked alphabet,* that is, a set Σ, equipped with an arity function $\sigma: \Sigma \to \mathbb{N}$. A Σ-term is defined recursively as follows: every letter of arity 0 (a constant) is a Σ-term, and if $a \in \Sigma$ has arity n and $t_1, \ldots, t_n$ are Σ-terms, then $a(t_1, \ldots, t_n)$ is a Σ-term. Terms are naturally (and unequivocally) represented by Σ-labeled trees,

where an a-labeled node has $\sigma(a)$ linearly ordered children. We let T_Σ be the set of all Σ-terms.

Thatcher and Wright introduced a model of automata for Σ-labeled trees, the so-called *bottom-up automata* [39,79]. To describe their expressiveness, they used the natural algebraic structure on the set of Σ-terms: each element $a \in \Sigma$ is an operation, of arity $\sigma(a)$, and no relation is assumed to hold between these operations. Now, let a Σ-algebra be any set S equipped with a $\sigma(a)$-ary operation a^S for each $a \in \Sigma$. If we use this algebraic notion, we can define recognizable and equational sets of Σ-terms: a subset $L \subseteq T_\Sigma$ is said to be recognizable if there exists a morphism (of Σ-algebras) φ from T_Σ to a finite Σ-algebra, such that $L = L\varphi\varphi^{-1}$; and L is equational if it is a component of a vector of maximal solutions of a system of polynomial equations. These systems are defined as in Sect. 1.3, except that the parameters p are now taken to be terms rather than words. Note that again, given $L \subseteq T_\Sigma$, there exists a unique least Σ-algebra recognizing it, called the *syntactic Σ-algebra* of L.

Thatcher and Wright also described subsets of T_Σ by *generalized rational expressions,* involving the letters, unions, the Σ-operations, and an appropriate notion of iteration.

Finally, Doner considered a logical formalism to be applied to the trees representing Σ-terms [29]: the individual variables are interpreted as nodes in a finite tree and the predicates are interpreted to express the labeling function and the parent-child relation.

Results of Doner, Thatcher and Wright [29,39,79] prove the following statement.

Theorem 3. *Let $L \subseteq T_\Sigma$. Then L is recognized by a bottom-up automaton, if and only if L is recognized by a infinite Σ-algebra, if and only if the syntactic Σ-algebra of L is finite, if and only if L is described by a generalized rational expression, if and only if L is defined by an MSO formula, if and only if L is equational.*

Moreover, there are algorithms to pass from one of these specification formalisms to each other.

Note that the particularity of this setting is that equational sets are recognizable. Another important remark is that deterministic bottom-up automata are really Σ-algebras, so the notions of automata-theoretic and algebraic recognizability are not really distinct. This last point makes Theorem 3 a little less satisfying than its word counterpart. Another (subjective) cause of dissatisfaction is that the generalized rational expression are rather awfully complex. Finally, this result has not made it easy to classify term languages in the spirit of Sect. 1.2. This is maybe due to a less long history of investigating the structural properties of finite Σ-algebras. Some interesting related results on binary trees, expressed in terms of certain context-free languages of words were proved by Beaudry, Lemieux, Thérien [5,6,7,8]. Nevertheless, it is fair to say that no structural theory of Σ-algebras clearly emerges.

An open question which may serve as a benchmark in this direction is the following: given a recognizable tree language, can one decide whether it is FO-definable?

3 The General Notion of Recognizability

Adapting the discussion in Sect. 2, one can easily define *recognizable* and *equational* subsets in any algebra. Recognizable sets are defined in terms of morphisms into finite algebras of the same type (or in terms of finite index congruences), and equational sets in terms of systems of polynomial equations (Mezei, Wright [60], Courcelle [17]). With those definitions, recognizable sets form a Boolean algebra, equational sets are closed under union, recognizable sets are always equational, finite sets are equational (even though they may fail to be recognizable), products of equational sets (using the operations in the algebra under consideration) are equational, but the analogous statement for recognizable sets is not always true, and the intersection of a recognizable and an equational set is equational. Finally, if $\varphi: S \to T$ is a morphism between algebras of the same type, then $L\varphi^{-1}$ is recognizable if $L \subseteq T$ is recognizable, and $L\varphi$ is equational if $L \subseteq S$ is equational.

3.1 Choosing an Algebraic Structure

As discussed above, if the sets we consider are naturally contained in an algebra, the notion of recognizability is straightforward. Sometimes however (frequently maybe), we want to discuss sets of relational structures, and we then design an algebraic signature to combine these structures.

For instance, it is one such abstract construction that has us see trees as terms. Consider even finite words: the interest of the model maybe lies simply in the notion of a totally ordered A-labeled finite set. We chose to view the set of words as a monoid under concatenation, and this gave rise to the notion of algebraically recognizable languages discussed in Sect. 1. We could also consider the following algebraic structure on the set of words: each letter $a \in A$ defines a unary operation $u \mapsto ua$. Then the set of all finite words is the algebra generated by A and the constant 1 (this amounts to considering the set of words as the algebra of $(A \cup \{1\})$-terms). One can verify that the notion of recognizable language is not modified – another sign of the robustness of the model of words. In fact, finite $(A \cup \{1\})$-algebras are naturally identified with deterministic finite state automata, and the equivalence between this notion of algebraic recognizability and the monoid-based one is a rephrasing of Kleene's theorem.

Relational structures, in the sense of this paper, are sets equipped with relations from a given relational signature. For instance, as mentioned earlier, words are A-labeled totally ordered sets: the relational signature consists of the binary order relation and one labeling unary relation R_a for each letter $a \in A$. In trees, the relations are the labeling relations and the parent-child binary relation (or the predecessor relation, or the parent and the sibling relations, etc., – these

choices are equivalent when it comes to expressing properties in monadic second order logic, see Sect. 3.3).

The choice of an algebraic structure can be guided by the natural constructions generating the finite relational structures under consideration (concatenation of words; construction of terms; construction of a word letter by letter), but there is really nothing canonical or unique about the algebraic structure.

Suppose for instance that we consider very few operations: then there are many more recognizable set, maybe to the extent that every set is recognizable (for exemple, consider the set of words with no operations at all: every finite partition, say L and L^c, is a finite index congruence). If on the other hand we have too many operations, then there will be less recognizable sets. For instance, let *sh* (for *shift*) be the unary operations on words that fixes 1, and maps ua to au ($a \in A$, $u \in A^*$). One can verify that the set a^*b is not recognizable for the algebra whose signature consists of the concatenation product and the shift operation. Another extreme example is given by $\mathbb{N}$, equipped with the constant 0 and the unary predecessor and successor operations: then the only recognizable sets are $\emptyset$ and $\mathbb{N}$.

It may also happen that adding certain operations does not change the class of recognizable sets. For instance, adding the mirror operation (defined inductively by $\tilde{1} = 1$ and $\widetilde{ua} = a\tilde{u}$) to the concatenation product, does not alter the notion of recognizability. See also Sect. 4.3.

In Sect. 4, we discuss a number of relational structures for which very interesting notions of recognizability have emerged in the literature.

3.2 Multi-sorted Algebras, Ordered Algebras, etc.

Sometimes, algebras are too constrained: the domain and the range of certain natural operations may consist of certain kinds of elements only. This is taken care of by the definition of *multi-sorted algebras,* see [17,25].

A typical example is provided by the study of languages of infinite words (more details are given in Sect. 4.1). It turns out that the best algebraic framework consists of considering simultaneously the finite and infinite words. One relevant operation is the concatenation product: between two finite words, it is the usual, fundamental operation, yielding a finite word; the product uv where u is a finite word and v is infinite, is an infinite word; and while it is possible to define the product of two infinite words, the outcome of such a product carries no significant information ($uv = u$) and the operation can be discarded. So we find that we need to consider two *sorts* of elements, finite and infinite, and two binary product operations, of type finite $\times$ finite $\to$ finite and finite $\times$ infinite $\to$ infinite. We also need to consider the ω-power, a unary operation of type finite $\to$ infinite (since it turns a finite word into an infinite one).

Another example is discussed in Sect. 4.3, where algebras with infinitely many sorts are considered. We do not want to give here a detailed discussion of congruences in multi-sorted algebra, only pointing out that such congruences can only identify elements of the same sort. If there are finitely many sorts,

recognizability is defined by considering morphisms into finite algebras, or finite-index congruences. If the algebraic signature under consideration has infinitely many sorts, non-trivial algebras are usually not finite, and we consider *locally finite* algebras (in which each sort has a finite number of elements) and *locally finite index* congruences (with a finite number of classes in each sort).

In the mid-1990s, Pin introduced the usage of *ordered semigroups* to refine the classification of recognizable languages [69]. The same idea can as naturally be used in any algebra (and has been for instance in [55]), but we will keep it outside the discussion in this paper to avoid increased complexity.

3.3 Definability Versus Recognizability

Based on the examples of words and trees, the natural language for logical definability of recognizable sets would seem to be MSO, *monadic second order logic.* It is actually more natural to use CMSO, *counting monadic second order logic.* CMSO [16,19] is monadic second order logic, enriched with the modulo quantifiers $\exists^{\bmod q} x$ introduced in Sect. 1. In the case of words, CMSO is equivalent to MSO. In fact, this holds for any relational structure that comes equipped with a linear order, or for which a linear order can be defined by a MSO-formula (*e.g. A*-labeled trees as in Sect. 2, or traces as in Sect. 4.2), but it is not true in general.

For instance, when discussing multisets (subsets of A with multiplicity), we can view them as A-labeled finite discrete graphs (graphs without edges). Then, MSO can only define finite and cofinite sets, and it is strictly weaker than CMSO. Note that, algebraically, the multisets on A under union, form the free commutative monoid on A. The same monoid can be interpreted in terms of traces (with a commutative alphabet), its elements are then viewed as certain directed acyclic graphs with one connected component per letter (see Sect. 4.2 on traces), and MSO is equivalent to CMSO in this context. The recognizable subsets are the same in both interpretations, since their definition is given in terms of the same algebraic structure, that of the free commutative monoid over A, but recognizability is equivalent to CMSO-definability in one interpretation, and to MSO-definability in the other.

Say that a map $\varphi: S \to T$ between sets of relational structures is a MS-transduction if there exist MSO-formulas (in the language of the relational structures in S) that express each $s\varphi$ (its domain and its relations) as a subset of a direct product of a fixed number of copies of s (see Courcelle [19] for a precise definition). For instance, if A_0 is the subset of constants in an alphabet A, the word in A_0^* formed by the leaves of an A-labeled tree t can be easily described by MSO-formulas inside the set of nodes of t.

Now consider a set of relational structures M, equipped with an algebraic structure with signature Σ. A simple example is given by $M = A^*$, the set of words on alphabet A, seen as a monoid: the signature Σ consists of a binary operation (interpreted in A^* as concatenation) and of $|A|$ constant symbols (interpreted in A^* as the letters of A). The valuation morphism val maps every

Σ-term (a Σ-labeled tree) to its interpretation in M. The following result is due to Courcelle [16,19].

Theorem 4. *If the valuation morphism is surjective and is an MS-transduction, then every CMSO-definable subset of M is recognizable.*

The mechanism of the proof is worth sketching: let $L \subseteq M$ be CMSO-definable. The inverse image of a CMSO-definable set by an MS-transduction is CMSO-definable, so $\mathsf{val}^{-1}(L)$ is CMSO-definable in T_Σ. But in the set of Σ-labeled trees, CMSO-definability is equivalent to MSO-definability, and hence to recognizability. And it is easy to show that if $\mathsf{val}^{-1}(L)$ is recognizable, then so is L.

Examining this sketch of proof also sheds light on the decidability of CMSO-defined sets, and on the complexity of such a decision problem. Suppose we have a parsing algorithm, which maps a given relational structure $x \in M$ to a Σ-term $\mathsf{parse}(x)$ describing it. Let $L \subseteq M$ be described by a CMSO-formula φ and let $x \in M$: we want to decide whether $x \in L$. An MSO-formula ψ describing $\mathsf{val}^{-1}(L)$ can be computed from φ and the formulas describing the MS-transduction val. The problem then reduces to deciding whether $\mathsf{parse}(x)$ satisfies ψ, and by Theorem 3, this can be solved (efficiently) by running $\mathsf{parse}(x)$ through a bottom-up tree automaton.

The converse of Theorem 4 does not always hold: there are situations, in particular in the discussion of languages of graphs, where some recognizable sets are not CMSO-definable. However, the two notions are known to be equivalent in important cases: we have already seen it for words or trees; other interesting situations are discussed in Sect. 4.2 and 4.3. It is interesting to note that a common feature of those situations where the notions of definability and recognizability coincide, is that we are able to describe a parsing function parse as an MS-transductions, and this is possible only because some finite generation condition is assumed to hold (which cannot be assumed for the class of all finite graphs).

For some of the specific relational structures discussed in the sequel, there is a notion of automaton that matches the definition of recognizability – but in many other situations, especially when dealing with graphs or posets, no such notion is known. In those cases, the algebraic approach is really the only tool we have to characterize logical definability, and to hope to bring about decision algorithms.

4 Recognizable Sets of Discrete Structures

For the discrete structures discussed in this section, fruitful algebraic structures have been introduced in the literature. The first measure of the interest of such algebraic structures, is whether the corresponding notion of recognizability matches some natural notion of logical definability, or some natural notion of recognizability by automata. A second measure of interest is whether the algebraic theory thus introduced allows us to characterize – and if possible decide –

significant classes of recognizable sets. Typically, deciding FO-definability is a key problem, but other classes may arise naturally depending on the type of discrete structures we consider.

4.1 Infinite Words

We start with infinite words because it is an area where the theory has been developed for a long time (Büchi's theorem goes back to the early 1960s), and has a strong algebraic flavor. Here we are talking of *one-way infinite words,* or *ω-words*, that is, A-labeled infinite chains, or elements of $A^{\mathbb{N}}$. For a detailed presentation of the results surveyed in this section, we refer the readers to Perrin and Pin's book [66] and to the survey papers [65,68].

The notions of *Büchi* and (deterministic) *Muller* or *Rabin automata* were evolved in the 1960s, and they were proved to have the same expressive power as MSO-formulas (on A-labeled infinite chains), and as *ω-rational expressions.* The latter describe every MSO-definable language of ω-words as finite unions of products of the form KL^{ω}, where K, L are recognizable languages of finite words and the ω-power denotes infinite iteration. In particular, this indicates that the sets of ω-words that can be accessed by MSO or automata-theoretic specifications are in a sense ultimately infinite iterations of a recognizable set of finite words.

An algebraic approach to ω-rational languages took longer to evolve. Early work of Arnold, Pécuchet, Perrin emphasized the necessary interplay of relations on $A^{\mathbb{N}}$ (concerning infinite words) and ordinary monoid congruences on A^* (concerning finite words). It also emphasized that nothing much could be expected from the monoid structure of $A^{\mathbb{N}}$, in which every product is equal to its first factor. Eventually, it was recognized that finite and infinite words cannot be considered separately, but they form a two-sorted algebra, as explained in Sect. 3.2. The definition of the binary concatenation product does not pose any problem, but must be split in one operation of type $\text{finite}^2 \to \text{finite}$ and one operation of type $\text{finite} \times \text{infinite} \to \text{infinite}$. But the generation of infinite words from finite one can be envisaged in two fashions: we can consider an ω-ary product, of type $\text{finite}^{\omega} \to \text{infinite}$, or the unary ω-power operation, of type $\text{finite} \to \text{infinite}$ The first choice is termed an *ω-semigroup* (Perrin, Pin), and $A^{\infty} = A^{+} \cup A^{\mathbb{N}}$ is (freely) generated by A as an ω-semigroup. The second choice is termed a *Wilke algebra* (Wilke), and the sub-Wilke algebra of A^{∞} generated by A consists in the finite and ultimately periodic ω-words only. A Ramsey theorem shows however that on a finite set, a Wilke algebra structure can be canonically extended to an ω-semigroup structure, so that the consideration of these two algebraic structures yields the same class of recognizable languages.

The robustness of this algebraic approach to recognizable subsets of A^{∞} (and not $A^{\mathbb{N}}$!) is such that an Eilenberg-style theory of varieties was developped (see Sect. 1), and that a good number of combinatorially or logically interesting classes of recognizable sets have been characterized algebraically (Perrin and Pin [66], Carton [22]).

From the algorithmic point of view, note that passing from a Büchi automaton to a deterministic Muller or Rabin automaton (say, for the purpose of complementation) is notoriously difficult, see Safra's exponential time algorithm, but no significantly better algorithm is possible [85].

Elegant results generalize this discussion to transfinite words, that is, A-labeled ordinals longer than ω, see the work of Bedon, Bruyère, Carton, Choueka [15,9,10,13].

For infinite trees, we know models of automata that are equivalent to MSO-decidability (Rabin, see [85]), but the extension of the algebraic ideas sketched above remains to be done. The finiteness results implied by Ramsey's theory seem much harder to obtain for trees.

4.2 Poset-Related Models

A pomset (partially ordered multiset) is an A-labeled poset. The first example, of course, is that of words, which are A-labeled chains. Other examples were considered, and first of all the case of traces.

Traces. There the alphabet is equipped with a structure – which can be viewed as an independence relation, or a dependence relation, or a distributed structure. A *trace* can then be viewed in several fashions: as an equivalence class of words in the free monoid A^*, in the congruence induced by the commutation of independent letters (so traces form a monoid); or as a so-called *dependence graph,* that is, an A-labeled poset where the order is constrained by the distributed structure of the alphabet, see Diekert and Rozenberg's book [27]. The latter is the more significant model, from the point of view of the original motivation of traces as a model of distributed computation.

The power of MSO-definability – interpreted on the dependence graph model – was proved to be equivalent to the power of *Zielonka's automata* (a model of automata which which incorporates information on the distributed structure of the alphabet), and to algebraic recognizability in the trace monoid [27].

Note that, as discussed in Sect. 3.3, in the particular case where the letters are independent from one another, the trace monoid is the free commutative monoid. When elements of this monoid are represented by trace dependence graphs, where for each letter $a \in A$, the set of A-labeled elements is a chain, then antichains have bounded cardinality (that of A), and a linearization of the poset can be defined by a MSO-formula, so MSO-definability is equivalent to CMSO-definability. When the elements of the same monoid are represented by finite discrete A-labeled graphs, without any edges, then MSO-definability is strictly weaker than CMSO-definability. In both cases however, recognizability is equivalent to CMSO-definability.

Good results are also known for FO-definable trace languages: they are characterized by star-free rational expressions, and by the aperiodicity of their syntactic monoid (Guaiana, Restivo, Salemi [42]), and important temporal logics with the same expressive power have been developed (see Thiagarajan and Walukiewicz [82] and Diekert and Gastin [26]).

There is a large body of literature on recognizable trace languages, and the results summarized above point to a rather well understood situation. Some questions however are not solved in a completely satisfactory fashion. For instance, the question of rational expressions for trace languages remains unclear (see the star problem): the difficulty comes from the fact that the star of a recognizable trace language may not be recognizable; the notion of concurrent star, which takes care of that obstacle, retains an *ad hoc* flavor [27]. Similarly, with the remarkable exception of FO-definable trace languages, the task of identifying, characterizing and deciding interesting subclasses of recognizable languages has eluded efforts.

One can argue that this is due to the loss of information that occurs if we consider the set of traces as a monoid – which we must do if the algebraic structure on the set of traces is that of a monoid: in the resulting definition of recognizability, a set of traces is recognizable if and only if its set of linearizations (in A^*) is recognizable. From an algebraic point of view, this puts the emphasis on commutation, but two traces may commute because they are independent, or because they are powers of a third one, in which case they are deeply dependent. From a more algorithmic point of view, what is done there is to reduce the study of a trace language to the study of the language of all its linearizations.

On the other hand, Zielonka's automata succeed in taking into account the distributed structure of the computation model, and are well-adapted to traces. Since they match monoid recognizability all the same, this points to the following problem: to find an alternative algebraic structure on the set of traces, which does not change the family of recognizable sets, yet better accounts for the distributed nature of that model, and hence (hopefully) naturally connects with Zielonka's automata (*i.e.,* provides an algebraic proof of Zielonka's theorem) and allows the identification and characterization of structurally significant subclasses of recognizable trace languages.

Infinite traces exhibit interesting properties, from the point of view of automata-recognizability and logical definability, see [27,30].

Message Sequence Charts and Communication Diagrams. *Message sequence charts (MSCs)* form a specification language for the design of communication protocols, that has attracted a lot of attention in the past few years. They can also be considered as specifications of particular pomsets, that are disjoint unions of k chains. An abstraction of this model is given by *Lamport Diagrams* (LDs) and by *Layered Lamport Diagrams* (LLDs), which are LDs subject to a boundedness condition.

Henriksen, Kumar, Mukund, Sohoni, Thiagarajan [43,44,62] considered the class of *bounded* finite MSC languages, defined by so-called *bounded* (Alur, Yannakakis [3]) or *locally synchronised* (Muscholl, Peled [63]) MSC-graphs. For bounded MSC languages, MSO-definability is equivalent to rationality of the language of all linearizations, and to recognizability by deterministic (resp. nondeterministic) *message-passing automata.* Kuske extended these results to FO-definable MSC languages, and to infinite bounded MSCs [53].

The restriction to classes of posets with a rational language of linearizations is rather severe, but little work so far has discussed definability or recognizability outside this hypothesis. Meenakshi and Ramanujam [59] and Peled [64] investigated decidable logics for MSCs and LLDs, that are structural, *i.e.,* not defined on the language of linearizations. There does not seem yet to exist an algebraic approach of (a subclass of LDs) that would match the power of MSO-definability.

Series-Parallel Pomsets. Sets of series-parallel pomsets (or *sp-languages*) were investigated by Lodaya, Weil, Kuske [52,55]. A poset is *series-parallel* if it can be obtained from singletons by using the operations of sequential and parallel product. There is a combinatorial characterization of these posets (*N*-free posets [41,87]), but the definition above naturally leads to the consideration of the so-called *series-parallel algebras* [55], that is, sets equipped with two binary associative operations, one of which is commutative. Kuske showed that an *sp*-language is recognizable if and only if it is CMSO-definable [52]. Lodaya and Weil introduced a model of branching automata and a notion of rational expressions, which they proved had the same expressive power [55]. However these automata accept not only the recognizable *sp*-languages, but also some non-recognizable ones.

The *bounded-width* condition is a natural constraint on *sp*-languages: a set L of series-parallel pomsets has bounded-width if there is a uniform upper bound on the cardinality of an anti-chain in the element so L. Results of Kuske, Lodaya and Weil [52,55] show that when we consider only bounded-width *sp*-languages, then recognizability is equivalent to automata-recognizability, to MSO-decidability, and to expressibility by a so-called *series-rational expression.* Under the bounded-width hypothesis, FO-definable *sp*-languages are characterized by a notion of star-free rational expressions, and by an algebraic condition on the syntactic *sp*-algebra which is analogous to the aperiodicity of monoids [52].

Texts and n-Pomsets. An *A-labeled text* is a finite A-labeled set, equipped with 2 linear orders. Texts form a particular class of the *2-structures* studied by Ehrenfeucht, Engelfriet, Harju, Proskurowski and Rozenberg [31,32,34]. Hoogeboom and ten Pas introduced an algebraic structure on the set of all texts [47]. This algebra has an infinite signature, but within any finitely generated subalgebra (generated by A and any finite subset of the signature, the hypothesis of *bounded primitivity* in [47,48]), recognizability is equivalent to MSO-definability.

The class of texts generated by the alphabet and the two arity 2 operations on texts *(alternating texts)* is of particular interest, as we now discuss.

A pair of linear orders $(\leq_1, \leq_2)$ on a finite set specifies and can be specified by a pair of partial orders $(\sqsubseteq_1, \sqsubseteq_2)$ such that every pair of distinct elements is comparable in exactly one of these partial orders (this defines a *2-poset*): it suffices to take $\leq_1 = \sqsubseteq_1 \cup \sqsubseteq_2$ and $\leq_2 = \sqsubseteq_1 \cup \sqsupseteq_2$; and conversely $\sqsubseteq_1 = \leq_1 \cap \leq_2$ and $\sqsubseteq_2 = \leq_1 \cap \geq_2$. Since the translation between texts and 2-posets is described by quantifier-free formulas, MSO-definability is preserved under this translation. On 2-posets, one can consider two natural operations: one behaves like a sequential

product on $\sqsubseteq_1$ and a parallel product on $\sqsubseteq_2$; and the other is defined dually, exchanging the roles of the two partial orders. Let *SPB(A)* be the algebra of 2-posets generated by A and these two operations. Ésik and Németh observed that these two operations on 2-posets translate to the two arity 2 operations of the text algebra; moreover, they introduced a simple model of automata for subsets of *SPB(A),* whose power is equivalent to recognizability and to MSO-definability [35].

Ésik and Németh's automata can also be defined for n-posets, where they are also equivalent to recognizability and to MSO-definability.

Pomsets in General. There does not seem to be a natural model of automaton that makes sense on all pomsets. However, since pomsets can be represented by A-labeled directed acyclic graphs *(dags),* they are directly concerned by the discussion in the next section. In particular, and getting ahead of ourselves, let us observe that the subsignature of the modular signature consisting of the operations defined by graphs that are posets (resp. dags), generates the class of all finite posets (resp. dags) - and the results on CMSO-definability discussed in Sect. 4.3 therefore apply to pomset languages.

4.3 Graphs and Relational Structures

Graphs (edge- or vertex-labeled, colored, with designated vertices, etc.), and beyond them, relational structures (*i.e.,* hypergraphs) are the next step, and they occur indeed in many modeling problems. The notion of logical definability is rather straightforward, although it may depend on the logical structure we consider on graphs (whether a graph is a set of vertices with a binary edge predicate, or two sets of vertices and edges with incidence predicates). From the algebraic point of view, there is no prominent choice for a signature to describe graphs. However, three signatures emerge from the literature. One of them, the modular signature, arises from the theory of modular decomposition of graphs, the other two (the *HR*- and the *VR*-signature) arise from the theory of graph grammars. We will also consider a fourth signature, on the wider class of relational structures. We will see that under suitable finiteness conditions, the resulting notions of recognizability are equivalent.

After briefly describing these signatures, and comparing the notions of recognizability which they induce, we rapidly survey known definability results. We conclude with the discussion of a couple of situations where automata-theoretic models have been introduced.

The Modular Signature. A concrete graph H with vertices $\{1, \ldots, n\}$ and edge set E_H, induces an n-ary operation on graphs as follows: the vertex set of the graph $H\langle G_1, \ldots, G_n\rangle$ is the disjoint union of the vertex sets of the G_i, it contains all the edges of the G_i, and for each edge $(i, j) \in E_H$, it also has all the edges from a vertex of G_i to a vertex of G_j. A graph is said to be *prime* if it cannot be decomposed non-trivially by such an operation. The *modular signature*

$\mathcal{F}_\infty$ consists of the set of all prime graphs, or rather, of one representative of each isomorphism class of prime graphs. It is an infinite signature. In particular, $\mathcal{F}_\infty$ contains a finite number of operations of each arity. The operations of arity 2 are the parallel product, the sequential product and the clique product: they are defined by the graphs with 2 vertices and no edge, 1 edge and 2 edges, respectively [18,88].

The theory of modular decomposition of graphs [57,61] shows that the class of all finite graphs is generated by the singleton graph and $\mathcal{F}_\infty$, and describes the relations between the operations in $\mathcal{F}_\infty$. Finite A-labeled graphs are generated by A and $\mathcal{F}_\infty$. If $\mathcal{F}$ is a finite subset of $\mathcal{F}_\infty$, the algebra generated by A and $\mathcal{F}$ is called the class of A-labeled $\mathcal{F}$-graphs.

For instance, if $\mathcal{F}$ consists of the sequential product, the $\mathcal{F}$-graphs are the finite words. If $\mathcal{F}$ consists of the parallel (resp. clique) product, they are the discrete graphs (resp. cliques). If $\mathcal{F}$ consists of the parallel and the sequential products, we get the series-parallel posets (see Sect. 4.2), and if it consists of the parallel and clique products, we get the cographs (see below).

The Signature *HR*. Here, graphs are considered as sets equipped with a binary edge predicate, and a finite number of constants (*i.e.,* designated vertices), called *sources.* Each finite set of source names defines a sort in the *HR*-algebra $\mathcal{GS}$ of *graphs with sources.* The operations in the algebra are the disjoint union of graphs with disjoint sets of source names, the renaming of sources, forgetting sources, and the fusion of two sources [19]. A number of variants can be considered, which do not affect the class of *HR*-recognizable subsets, see Courcelle and Weil [21]: the disjoint union can be replaced with parallel composition (source name sets need not be disjoint, and sources with the same name get identified), the sources may be assumed to be pairwise distinct *(source separated graphs),* the source renaming operations can be dropped, or the source forgetting operations, etc.

The signature *HR* emerged from the literature on graph grammars, and the acronym *HR* stands for *Hyperedge Replacement.* More precisely, the equational sets of graphs, relative to the signature *HR,* are known to enjoy good closure properties, and can be elegantly characterized in terms of recognizable tree languages and MS-transductions where both vertex and edge sets can be quantified (see Courcelle [19]).

The Signature *VR*. Now graphs are considered as sets equipped with a binary edge predicate, and a finite number of unary predicates (*i.e.,* colors on the set of vertices), called *ports.* Each finite set of port names defines a sort in the *VR*-algebra $\mathcal{GP}$ of *graphs with ports.* The operations in the algebra are the disjoint union, the edge adding operation (adding an edge from each p-port to each q-port for designated port names p, q), and the renaming and forgetting of port names. Again, the class of *VR*-recognizable subsets is not affected by variants such as the consideration of graphs where ports must cover the vertex set, or

must partition it. It also coincides with $\mathcal{NLC}$-recognizable graphs, see Courcelle and Weil [21].

The signature *VR* (standing for *Vertex Replacement*) also emerged from the literature on graph grammars, and the equational sets of graphs, relative to the signature *VR,* enjoy good closure properties, and are characterized in terms of recognizable tree languages and MS-transductions where only vertex sets can be quantified (see Courcelle [19]).

The Signature $\mathcal{S}$ on Relational Structures with Sources. Subsuming the algebras of graphs with sources and with ports, we can consider the class of *relational structures with sources* $\mathcal{S}t\mathcal{S}$. These are sets equipped with a finite relational structures, and a finite number of constants *(sources).* Each pair consisting of a relational signature and a set of source names defines a sort, and the operations in the signature $\mathcal{S}$ are the disjoint union between sorts with disjoint sets of source names, and all the unary operations that can be defined on a given sort using quantifier-free formulas, see [21]. The operations in the signatures *VR* and *HR* are particular examples of such *quantifier-free definable operations.* The notion of $\mathcal{S}$-recognizability is not affected if we consider parallel composition instead of disjoint union (as for the signature *HR*), nor if we consider only structures where sources are separated [21].

Comparing the Notions of Recognizability. Combining a number of results of Courcelle and Weil [21], we find that a set of graphs is *VR*-recognizable if and only if it is $\mathcal{S}$-recognizable. Moreover, a *VR*-recognizable set of graphs is $\mathcal{F}_\infty$-recognizable, and it is also *HR*-recognizable (the implication for *HR*- and *VR*-equational sets goes in the other direction). Finally, $\mathcal{F}_\infty$-, *HR*- and *VR*-recognizability (resp. equationality) are equivalent under certain boundedness conditions.

In particular if we consider a set L of graphs without $\boldsymbol{K}_{n,n}$ for some n ($\boldsymbol{K}_{n,n}$ is the complete bipartite directed graph with $n + n$ vertices), then L is *VR*-recognizable if and only if it is *HR*-recognizable [21]. This sufficient condition is implied by the following boundedness properties (in increasingly general order): the graphs in L have uniformly bounded degree, they have bounded tree-width, they are sparse. The notion of bounded tree-width can be seen as a finite generation property, relative to the signature *HR* [19].

If $\mathcal{F}$ is a finite subset of the modular signature $\mathcal{F}_\infty$ and L is a set of $\mathcal{F}$-graphs, then L is $\mathcal{F}$-recognizable if and only if it is $\mathcal{F}_\infty$-recognizable (resp. *VR*-recognizable, $\mathcal{S}$-recognizable) [18,21].

Monadic Second-Order Definability. From the logical definability point of view, graphs can be seen as sets (of vertices) equipped with an edge predicate, or as pairs of sets (of vertices and edges respectively) with incidence predicates. Let us denote by CMSO[E] the CMSO emerging from the first point of view, and by CMSO[inc] the second one. It is easily verified that CMSO[E]-definable sets

of graphs are also CMSO[inc]-definable. Moreover CMSO[E]-definability implies *VR*-recognizability, and CMSO[inc]-definability implies *HR*-recognizability, see Courcelle [19].

Lapoire showed that if L is a set of graphs with bounded tree-width, then CMSO[inc]-definability is equivalent to *HR*-recognizability [54]. Moreover, if L is uniformly sparse, then CMSO[E]- and CMSO[inc]-definability are equivalent (Courcelle [20]). In view of the equivalence result between *HR*- and *VR*-recognizability mentioned above, it would be interesting to find out whether both definabilities are also equivalent if L is without $\boldsymbol{K}_{n,n}$ for some n.

Returning to the modular signature, CMSO[E]-definability implies $\mathcal{F}_\infty$-recognizability, by general reasons (see Sect. 3.3). Weil showed that the converse holds for sets of $\mathcal{F}$-graphs, provided $\mathcal{F}$ is finite and the operations of $\mathcal{F}$ enjoy a limited amount of commutativity (*weakly rigid signature,* see [88] for details). This assumption is rather general, and covers in particular all the cases where $\mathcal{F}$-graphs are dags or posets, and notably the case of *sp*-languages.

A typical example of a subsignature of $\mathcal{F}_\infty$ which is not weakly rigid, consists of the parallel and the clique products, two binary commutative, associative operations which generate the cographs. *Cographs* form a class of undirected graphs, closely related with comparability graphs (Corneil, Lerchs, Stewart [23]) and can be characterized as follows: an undirected graph is a cograph if and only if it does not contain P_4 (P_4 has vertex set $\{1, \ldots, 5\}$ and edges between i and $i+1$, $1 \leq i \leq 4$). The arguments that show that $\mathcal{F}$-recognizability is equivalent to CMSO[E]-definability when $\mathcal{F}$ is weakly rigid, fail for cographs. Courcelle [18] asks whether CMSO-definability is strictly weaker than $\mathcal{F}$-recognizability for a general finite subsignature $\mathcal{F} \subseteq \mathcal{F}_\infty$: the first place to look for a counter-example seems to be cographs.

Series-Σ Algebras. In their investigation of *sp*-languages, Lodaya and Weil introduced series-Σ-algebras and their subsets ($s\Sigma$-languages): here Σ is a ranked alphabet (as in the study of tree languages, Sect. 2) and $\bullet$ is a binary associative operation not in Σ. The *$s\Sigma$-terms*, that is, the elements of the algebra freely generated by A, Σ and $\bullet$ (called the *free series-Σ-algebra over A*), can be viewed as finite sequences of Σ-trees, where each child of the root is in fact a smaller $s\Sigma$-term. Lodaya and Weil introduced a model of automata and a notion of rational expressions, both of which are equivalent to recognizability [56] – a result which generalizes the characterization of recognizability by finite automata for both words and trees. They showed that their result could be adapted if some of the operations in Σ were assumed to be commutative, but not if some amount of associativity was introduced (e.g. *sp*-languages, cographs). The logical dimension of $s\Sigma$-languages was not developped.

Automata for Graph Languages. We have seen some automata, designed for specific situations (finite and infinite words, traces, series-parallel pomsets, MSC languages, n-pomsets), see Sect. 4.2. As discussed there, these automata mod-

els match the expressiveness of recognizability and MSO-definability, sometimes under additional boundedness hypothesis.

For general graph languages, Thomas introduced the notion of *graph acceptor* [84,86], generalizing the *tiling systems* introduced earlier by Giammarresi and Restivo [40] for *pictures* (A-labeled rectangular grids). Recognizability by a graph acceptor was shown to be equivalent to EMSO-definability, where EMSO is the extension of FO by existential quantification of monadic second order variables.

4.4 Revisiting Trees

As mentioned in Sect. 2, the problem of deciding whether a given recognizable set of Σ-trees is FO-definable is still open, and various attempts to use the structure of Σ-algebras described in Sect. 2 to solve it in the spirit of Schützenberger's theorem (Theorem 2), have failed [46,71,72]. Recently, Ésik and Weil introduced a new algebraic framework to investigate this particular problem on tree languages [36]. The point was to enrich the algebraic framework, without modifying the notion of a recognizable subset, but introducing additional algebraic structure.

Ésik and Weil's algebras, called *preclones,* are multi-sorted algebras with one sort for each integer n, and Σ-trees form the 0-sort of the free Σ-generated preclone. As indicated, a set of Σ-trees is preclone-recognizable if and only if it is recognizable with respect to Σ-algebras; moreover, if L is a recognizable Σ-tree language, its syntactic preclone (which is finitary but not finite due to the infinite number of sorts) admits a finite presentation, encoded in the minimal bottom-up automaton of L. This is naturally important if we want to use syntactic preclones in algorithms.

The main result of [36] states that L is FO-definable if and only if its syntactic preclone lies in the least pseudovariety of preclones closed under *two-sided wreath products,* and containing a certain very simple 1-generated preclone. The two-sided wreath product is a generalization of the operation of the same name on monoids (Rhodes, Tilson, see [77]), and this result generalizes Schützenberger's theorem on finite words. It is the first algebraic characterization of FO-definable tree languages, but unfortunately, it is not clear at this point whether this characterization can be used to derive a decision algorithm.

The approach in [36] also applies to FO+MOD-definable tree languages, and other similarly defined languages.

4.5 Timed Models

Timed automata appeared in the 1990s, to represent the behavior of finite state systems subjected to explicit time constraints (Alur, Dill [2]). While they are already widely used, the foundations of the corresponding theory are still under development. There are several variants of these automata, such as event-clock automata, and of the models of timed computations (timed words, clock words, etc.). There have also been several attempts to develop appropriate notions of rational expressions, that would be equivalent to the expressive power of timed

automata, see Henzinger, Raskin, Schobbens [45], Asarin, Maler, Caspi [4], Dima [28], Maler, Pnueli [58] among others. At the same time, timed automata and timed languages may exhibit paradoxical behaviors, due to the continuous nature of time, so the central ideas and techniques from the classical theory cannot simply be enriched with timed constraints to account for the behavior of timed automata.

The development of an algebraic apparatus and of a logical formalism is also still in its infancy. One should mention however the recent work of Maler and Pnueli [58], and the results of Francez and Kaminski [38] and Bouyer, Petit and Thérien [12] on generalizations of timed languages and automata, to *automata on infinite alphabets* and to *data languages,* respectively. In both cases, an interesting notion of algebraic recognizability is introduced, that is at least as powerful as timed automata. languages. Moreover, several logics have been introduced (see for instance Demri, D'Souza [24]), but none is completely satisfactory with respect to the motivation of formulating and solving the controller synthesis problem, and none is connected in a robust way to an algebraic approach of recognizability.

The development of a complete theory of timed systems, incorporating automata-theoretic, algebraic and logical aspects, appears to be one of the more difficult challenges of the moment.

References

1. J. Almeida. *Finite semigroups and universal algebra.* Series in Algebra **3**, World Scientific, Singapore (1994).
2. R. Alur, D. Dill. A Theory of Timed Automata, *Theoret. Computer Science* **126** (1994) 183-235.
3. R. Alur, M. Yannakakis. Model Checking of Message Sequence Charts. In: *Proc. CONCUR 99,* Lect. Notes in Computer Science **1664**, 114-129, Springer, 1999.
4. E. Asarin, O. Maler, P. Caspi Timed Regular Expressions, *Journal of the ACM* **49** (2002) 172-206.
5. M. Beaudry. Finite idempotent groupoids and regular languages, *RAIRO Informatique théorique* **32** (1998) 127-140.
6. M. Beaudry. Languages recognized by finite aperiodic groupoids, *Theoret. Computer Science* **209** (1998) 299-317.
7. M. Beaudry, F. Lemieux. D. Thérien. Finite loops recognize exactly the regular open languages. In *Proc. ICALP 97,* Lect. Notes in Computer Science (1997), pp. 110-120.
8. M. Beaudry, F. Lemieux. D. Thérien. Star-free open languages and aperiodic loops. In *Proc. STACS 01,* Lect. Notes in Computer Science **2010**, 87-98, Springer, 2001.
9. N. Bedon. Logic over words on denumerable ordinals, *J. Computer and System Science* **63** (2001) 394-431.
10. N. Bedon. Automata, semigroups and recognizability of words on ordinals, *Intern. J. Algebra and Computation* **8** (1998) 1-21.
11. J. Berstel. *Transductions and context-free languages,* Teubner, Stuttgart (1979).
12. P. Bouyer, A. Petit, D. Thérien. An algebraic characterization of data and timed languages, *Information and Computation* **182** (2003) 137-162.

13. V. Bruyère, O. Carton. Automata on linear orderings. In: *Proc. DLT 02* (M. Ito, M. Toyama, eds.), Lect. Notes in Computer Science **2450**, 103-115, Springer 2002.
14. S. Cho, D.T. Huynh. Finite automaton aperiodicity is PSAPCE-complete. *Theoret. Computer Science* **88** (1991) 99-116.
15. Y. Choueka. Finite automata, definable sets, and regular expressions over ω^n-tapes, *J. Computer and System Sciences* **17** (1978) 81-97.
16. B. Courcelle. The monadic second-order logic of graphs I: recognizable sets of finite graphs, *Information and Computation* **85** (1990) 12-75.
17. B. Courcelle. Basic notions of universal algebra for language theory and graph grammars, *Theoret. Computer Science* **163** (1996) 1-54.
18. B. Courcelle. The monadic second-order logic of graphs X: Linear orders, *Theoret. Computer Science* **160** (1996) 87-143.
19. B. Courcelle. The expression of graph properties and graph transformations in monadic second order logic. In: *Handbook of Graph Grammars and Computing by Graph Transformations* (G. Rozenberg, ed.), vol. 1, chapter 5. World Scientific, 1997, 313-400.
20. B. Courcelle. The monadic second-order logic of graphs XIV: Uniformly sparse graphs and edge set quantifications, *Theoret. Computer Science* **299** (2003) 1-36.
21. B. Courcelle, P. Weil. The recognizability of sets of graphs is a robust property, to appear.
22. O. Carton. Wreath product and infinite words, *J. Pure and Applied Algebra* **153** (2000) 129-150.
23. D. Corneil, H. Lerchs, L. Stewart. Complement reducible graphs, *Discrete Applied Mathematics* **3** (1981) 163-174.
24. S. Demri, D. D'Souza. An Automata-Theoretic Approach to Constraint LTL. In: *Proc. FSTTCS 02,* Lect. Notes in Computer Science **2256**, 121-132, Springer, 2002.
25. K. Denecke, S. L. Wismath. *Universal Algebra and* Applications *in Theoretical Computer Science,* Chapman and Hall (2002).
26. V. Diekert, P. Gastin. Pure future local temporal logics are expressively complete for Mazurkiewicz traces. In: *Proceedings of LATIN'04*, Lect. Notes in Computer Science **2976**, 232-241, Springer, 2004.
27. V. Diekert, G. Rozenberg (eds.). The *book of traces,* World Scientific (1995).
28. C. Dima. Real-time automata and the Kleene algebra of sets of real numbers, *J. Automata, Languages and Computation* **6** (2001) 3-23.
29. J. Doner. Tree acceptors and some of their applications, *J. Comput. System Sci.* **4** (1970) 406-451.
30. W. Ebinger, A. Muscholl. On logical definability of omega-trace languages. *Theoret. Computer Science* **154** (1996) 67-84.
31. A. Ehrenfeucht, G. Rozenberg. Theory of 2-structures, Part 1: clans, basic subclasses and morphism. Part 2: representation through labeled tree families, *Theoret. Computer Science* **70** (1990) 277-303, 305-342.
32. A. Ehrenfeucht, G. Rozenberg. Angular 2-structures, *Theoret. Computer Science* **92** (1992) 227-248.
33. S. Eilenberg. *Automata, languages and machines,* vol. A (1974) and vol. B (1976), Academic Press, New York.
34. J. Engelfriet, T. Harju, A. Proskurowski, G. Rozenberg. Characterization and complexity of uniformly nonprimitive labeled 2-structures, *Theoret. Computer Science* **154** (1996) 247-282.
35. Z. Ésik, Z. Németh. Higher dimensional automata, to appear.

36. Z. Ésik, P. Weil. On logically defined recognizable tree languages. In: *Proc. FST TCS 2003* (P. K. Pandya, J. Radhakrishnan eds.), Lect. Notes in Computer Science **2914** (2003), Springer, 195-207.
37. K. Etessami, M. Y. Vardi, Th. Wilke. First-order logic with two variables and unary temporal logic. *Inform. and Computation* **179** (2002) 279-295.
38. N. Francez, M. Kaminski. An algebraic characterization of deterministic regular languages over infinite alphabets, *Theoret. Computer Science* **306** (2003) 155-175.
39. F. Gécseg, M. Steinby. *Tree automata*, Akadémiai Kiadó, Budapest (1994).
40. D. Giammarresi, A. Restivo. Two-dimensional languages. In: Handbook of Formal Languages (G. Rozenberg and A. Salomaa, eds), volume III, Springer, New York, 1997.
41. J. Grabowski. On partial languages, *Fundamenta Informatica* **4** (1981) 427-498.
42. G. Guaiana, A. Restivo, S. Salemi. Star-Free Trace Languages, *Theoret. Computer Science* **97** (1992) 301-311.
43. J.G. Henriksen, M. Mukund, K. Narayan Kumar, P.S. Thiagarajan. Regular Collections of Message Sequence Charts. In: *Proc. MFCS 00,* Lect. Notes in Computer Science **1893**, 405-414, Springer, 2000.
44. J.G. Henriksen, M. Mukund, K. Narayan Kumar, P.S. Thiagarajan. On Message Sequence Graphs and Finitely Generated Regular MSC Languages. In: *Proc. ICALP 00,* Lect. Notes in Computer Science **1853**, 675-686, Springer, 2000.
45. T. Henzinger, J.-F. Raskin, P.-Y. Schobbens. The Regular Real-Time Languages. In: *Proc. ICALP'98,* Lect. Notes in Computer Science **1443**, 580-591, Springer, 1998.
46. U. Heuter. First-order properties of trees, star-free expressions, and aperiodicity. In: *Proc. STACS 88,* Lect. Notes in Computer Science **294**, 136-148, Springer, 1988.
47. H.J. Hoogeboom, P. ten Pas. Text languages in an algebraic framework, *Fundamenta Informatica* **25** (1996) 353-380.
48. H.J. Hoogeboom, P. ten Pas. Monadic second order definable text languages, *Theory Comput. Syst.* **30** (1997) 335-354.
49. J. A. Kamp. *Tense logic and the theory of linear order*, Ph. D. Thesis, UCLA, 1968.
50. S. Kim, R. McNaughton, R. McCloskey. A polynomial time algorithm for the local testability problem of deterministic finite automata, *IEEE Trans. Comput.* **40** (1991) 1087-1093.
51. E. Kushilevitz, N. Nisan. *Communication Complexity*, Cambridge University Press (1997).
52. D. Kuske. Towards a language theory for infinite *N*-free pomsets, *Theoret. Computer Science* **299** (2003) 347-386.
53. D. Kuske. Regular sets of infinite message sequence charts, *Information and Computation* **187** (2003) 90-109.
54. D. Lapoire. Recognizability equals Monadic Second-Order definability, for sets of graphs of bounded tree-width. In: *Proc. STACS 98.* Lect. Notes in Computer Science **1373**, 618-628, Springer, 1998.
55. K. Lodaya, P. Weil. Series-parallel languages and the bounded-width property, *Theoret. Computer Science* **237** (2000) 347-380.
56. K. Lodaya, P. Weil. Rationality in algebras with a series operation, *Information and Computation* **171** (2001) 269-293.
57. R. McConnell, J. Spinrad. Modular decomposition and transitive orientation, *Discrete Mathematics* **201** (1999) 189-241.

58. O. Maler, A. Pnueli. On Recognizable Timed Languages. In: *Proc. FOSSACS 04,* Lect. Notes in Computer Science **2987**, 348-362, Springer, 2004.
59. B. Meenakshi and R. Ramanujam. Reasoning about layered message-passing systems. In: *Proc. VMCAI 03,* of Lect. Notes in Computer Science **2575**, 268-282, Springer, 2003.
60. J. Mezei and J. B. Wright. Algebraic automata and context-free sets, *Information and Control* **11** (1967) 3-29.
61. R.H. Möhring, F.J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Mathematics* **19** (1984) 257-356.
62. M. Mukund, K. Narayan Kumar, M. Sohoni. Synthesizing distributed finite-state systems from MSCs. In: *Proc. CONCUR 00,* Lect. Notes in Computer Science **1877**, 521-535, Springer, 2000.
63. A. Muscholl, D. Peled. Message sequence graphs and decision problems on Mazurkiewicz traces. In: *Proc. MFCS 99* (M. Kutylowski, L. Pacholski, eds.), Lect. Notes in Computer Science **1672**, 81-91, Springer, 1999.
64. D. Peled. Specification and Verification of Message Sequence Charts. In: *Proc. FORTE 2000,* IFIP Conference Proceedings **183**, 139-154, Kluwer, 2000.
65. D. Perrin, J.-E. Pin. Semigroups and automata on infinite words. In: *NATO Advanced Study Institute Semigroups, Formal Languages and Groups* (J. Fountain, ed.), pp. 49-72, Kluwer, 1995.
66. D. Perrin, J.-E. Pin. *Infinite words,* Pure and Applied Mathematics Vol. 141, Elsevier (2004).
67. J.-E. Pin. *Variétés de langages formels,* Masson, Paris (1984); English translation: Varieties *of formal languages,* Plenum, New-York (1986).
68. J.-E. Pin. Logic, Semigroups and Automata on Words. *Annals of Mathematics and Artificial Intelligence* **16** (1996) 343–384.
69. J.-E. Pin. Syntactic semigroups. Chap. 10 in: *Handbook of formal languages,* Vol. I (G. Rozenberg et A. Salomaa, eds.), Springer Verlag, 1997, 679-746.
70. J.-E. Pin, P. Weil. Polynomial closure and unambiguous product. *Theory Comput. Systems* **30** (1997) 383-422.
71. A. Potthoff. Modulo counting quantifiers over finite trees. In: *Proc. CAAP '92,* Lect. Notes in Computer Science **581**, 265-278, Springer, 1992.
72. A. Potthoff. First order logic on finite trees. In: *Proc. TAPSOFT '95,* Lect. Notes in Computer Science **915**, Springer, 1995.
73. J. Sakarovitch. *Élements de théorie des automates,* Vuibert, Paris (2003).
74. M. P. Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum* **13** (1976) 47-75.
75. M. Sipser. *Introduction to the theory of computation,* International Thompson Publishing (1996).
76. J. Stern. Characterization of some classes of regular events. *Theoret. Computer Science* **35** (1985) 17-42.
77. H. Straubing. *Finite automata, formal logic, and circuit complexity,* Birkhaüser (1994).
78. P. Tesson, D. Thérien. Complete classifications for the communication complexity of regular languages. In *STACS 2003,* H. Alt, M. Habib (Eds.), Lect. Notes in Computer Science **2607**, pp. 62-73, Springer (2003).
79. J. W. Thatcher, J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Math. Systems Theory* **2** (1968) 57-81.

80. D. Thérien, Th. Wilke. Over words, two variables are as powerful as one quantifier alternation: $\mathsf{FO}^2 = \Sigma_2 \cap \Pi_2$. In *Proc. 13th ACM SToC,* 41-47, 1998.
81. D. Thérien, Th. Wilke. Temporal logic and semidirect products: An effective characterization of the until hierarchy. *SIAM Journal on Computing* **31** (2002) 777-798.
82. P.S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *Proc. LICS 97,* 183-194, 1997.
83. W. Thomas. Classifying regular events in symbolic logic. *J. Comput. Syst. Sci.* **25** (1982) 360–375.
84. W. Thomas. Automata theory on trees and partial orders. In: *Proc. TAPSOFT 97* (M. Bidoit, M. Dauchet, eds), Lect. Notes in Computer Science **1214**, 20-38, Springer, 1997.
85. W. Thomas. Automata on infinite objects. In: *Handbook of Theoretical Computer Science* (J. van Leeuwen ed.), vol. B. Elsevier, 1990, 133-192.
86. W. Thomas. Languages, automata, and logic. In: *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, eds), volume III, 389-455, Springer, New York, 1997.
87. J. Valdes, R.E. Tarjan and E.L. Lawler. The recognition of series-parallel digraphs, *SIAM J. Comput.* **11** (1981) 298-313.
88. P. Weil. On the logical definability of certain graph and poset languages, *J. Automata, Languages and Computation* **9** (2004) 147-165.

Geometric Optimization and Unique Sink Orientations of Cubes

Emo Welzl

Institute of Theoretical Computer Science
ETH Zürich
CH-8092 Zürich, Switzerland
emo@inf.ethz.ch

Abstract. There is an ongoing attempt of designing a provably fast method for solving Linear Programs and related geometric optimization problems by combinatorial methods in the unit cost model (as opposed to the bit-model, where polynomial methods are known). Among others, this research has brought forth randomized methods with subexponential running time, but also a number of interesting combinatorial models like Oriented Matroids, Abstract Objective Functions (AOF), Unimodal Numberings, LP-type Problems, Abstract Optimization Problems (AOP), Unique Sink Orientations of Cubes (USO), et cetera. Although many of these models are quite general, so far there has been little success in showing good lower bounds even in these generalized abstractions. As for the simplex method, there are a number of open questions concerning several pivoting rules, notably randomized ones.

After a general introduction, we will focus on one particular model: *unique sink orientations of cubes (USO).* To this end consider (the edge graph of) an n-dimensional hypercube with its edges oriented so that every face has a unique sink. Such an orientation is called a unique sink orientation, and we are interested in finding the unique sink of the whole cube when the orientation is given implicitly. The basic operation available is the so-called vertex evaluation, where we can access an arbitrary vertex of the cube, for which we obtain the orientations of the incident edges.

Unique sink orientations occur when the edges of a deformed geometric n-dimensional cube (i.e., a polytope with the combinatorial structure of a cube) are oriented according to some generic linear function. These orientations are easily seen to be acyclic. The main motivation for studying unique sink orientations are certain linear complementarity problems, which allow this combinatorial abstraction (due to Alan Stickney and Layne Watson), where orientations with cycles can arise. Similarly, linear programming and some quadratic optimization problems, like computing the smallest enclosing ball of a finite point set, are polynomial time reducible (in the unit cost model) to finding a sink in a unique sink orientation (possibly with cycles).

The talk surveys some results concerning upper and lower bounds for algorithms finding the sink in a USO (acyclic or possibly with cycles).

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, p. 176, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Congestion Games and Coordination Mechanisms*

Elias Koutsoupias

Department of Informatics, University of Athens,
Panepistimiopolis Ilisia, Athens 15784, Greece.
elias@di.uoa.gr

Abstract. In recent years we witness a fusion of ideas coming from the fields of Game Theory, Networks, and Algorithms. One of the central ideas in this new area is the notion of the price of anarchy [5,7] which is an attempt to measure the deterioration in performance of a system due to selfish behavior of its users or components. The study of the price of anarchy in general games is pointless; it can be arbitrary even for simple 2×2 games. But it becomes interesting when we restrict our attention to congestion games and to their natural generalizations and variants.

Ideally when users share the resources of a system they behave in a way that optimizes the performance of the system. However, when the users are selfish, they will act in a way that optimizes their own individual and usually conflicting objectives. As an example, consider a set of users that compete for the links of a network. The situation results in a network congestion game [8]: In a congestion game there are users/players and facilities (links in the case of networks). To each user we associate some collections of facilities (paths from a source to a destination in the case of networks); these are the pure strategies of the user. The cost of each facility depends on the number of users that use it; the cost of a user is the sum of the cost of the facilities in its selected strategy. One nice property of congestion games is the not-so-trivial fact that they all have at least one *pure* Nash equilibrium. Special case of congestion games of particular interest are the singleton games when each strategy is a singleton set (each user uses only one facility) and when costs are linear (the cost of each facility is proportional to the number of users using it). This is essentially the selfish version of the Graham's classical scheduling problem of identical tasks. Also of particular interest are symmetric network congestion games with a continuum of users; these are the games studied by Roughgarden and Tardos [9].

Given a set of strategies of a congestion game, we can define the social cost (or system cost) as either the maximum or the average cost among the users. Then the (pure) price of anarchy of a congestion game is the maximum ratio among all (pure) Nash equilibria of the social cost of a Nash equilibrium over the social cost of the optimal set of strategies. What is the pure and mixed price of anarchy of congestion games? of singleton and linear games? We know the answers to most of these questions but not to all of them [5,3,4].

* Supported in part by the IST (FLAGS, IST-2001-33116) program.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 177–179, 2004.
© Springer-Verlag Berlin Heidelberg 2004

There are two natural generalizations of congestion games: when users have weights—in this case, the cost of each facility depends on the total weight of the users using it—and when facilities differentiate between users. Of particular interest is the class of the singleton games with weights, which corresponds essentially to selfish scheduling. Again we know the price of anarchy of some of these cases but many cases are still open.

Given that in many cases the price of anarchy is high, one can ask how to redesign a game to improve its price of anarchy. There are interesting suggestions such as adding taxes [1] or more choices to players [6]. A natural way to address the problem of redesigning a congestion game is the notion of coordination mechanisms: Given a congestion game, consider the class of congestion games when we are allowed to increase the cost on the facilities and when we can differentiate among players (with player-specific costs). The idea is that in many practical situations, increasing the cost of a facility corresponds to "slowing down" which can be implemented easily; similarly, differentiating between players can be naturally done in many cases. A coordination mechanism then simply selects among these games one that has small price of anarchy.

The situation becomes more interesting when we consider congestion games with weights. In this case, we can naturally assume that when we redesign a game we don't know the weights of the users. For example, in the special case of selfish scheduling we want to have a scheduling policy on each facility which is fixed in advance (i.e., independent of the weights of the jobs/users). The situation resembles very much the framework of competitive analysis: We, the designers, select a scheduling policy for each facility. Then an adversary selects the weights of jobs and we compute the price of anarchy.

The immediate problem is to estimate or bound the optimal price of anarchy achieved by coordination mechanisms. For most classes of congestion games, their special cases, and their variants, the problem is open. We know only partial answers to special cases, such as the case of singleton games [2].

References

1. R. Cole, Y. Dodis, and T. Roughgarden. Pricing network edges for heterogeneous selfish users. In *ACM STOC,* pages 521–530, 2003.
2. G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. In *ICALP,* 2004.
3. A. Czumaj and B. Vöcking. Tight Bounds for Worst-case Equilibria. In *ACM-SIAM SODA,* pp. 413–420, 2002.
4. M. Gairing, T. Lücking, M. Mavronikolas, and B. Monien. Computing Nash equilibria for scheduling on restricted parallel links. In *STOC,* 2004.
5. E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *STACS,* pages 404-413, 1999.
6. D. Monderer and M. Tennenholtz. k-Implementation. In *ACM EC,* pp 19–28, 2003.
7. C. H. Papadimitriou. Algorithms, games, and the Internet. In *ACM STOC,* pp 749-753, 2001.

8. R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory,* 2:65-67, 1973.
9. T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM,* 49(2):236-259, 2002.

Equitable Colorings of Bounded Treewidth Graphs

Hans L. Bodlaender[1] and Fedor V. Fomin[2*]

[1] Institute of Information and Computing Sciences, Utrecht University, 3508 TA Utrecht, The Netherlands,
hansb@cs.uu.nl
[2] Department of Informatics, University of Bergen, N-5020 Bergen, Norway,
fomin@ii.uib.no

Abstract. A proper coloring of a graph G is *equitable* if the sizes of any two color classes differ by at most one. A proper coloring is ℓ-bounded, when each color class has size at most ℓ. We consider the problems to determine for a given graph G (and a given integer ℓ) whether G has an equitable (ℓ-bounded) k-coloring. We prove that both problems can be solved in polynomial time on graphs of bounded treewidth, and show that a precolored version remains NP-complete on trees.

1 Introduction

There is a wide believe that almost every natural hard problem can be solved efficiently on graphs of bounded treewidth. Of course this is not true, a nice example is the BANDWIDTH MINIMIZATION problem which is NP-hard even on trees of degree three [11,25]. Another part of 'folklore' in the Graph Algorithms community is that if some (natural) problem can be solved in polynomial time on trees, one should be able to solve it in polynomial time on graphs of bounded treewidth. However, there are some striking and frustrating examples, like $L(2, 1)$-COLORING, where an efficient algorithm for trees is known [7], but the complexity of the problem on graphs of treewidth ≥ 2 is still open. For more than ten years, EQUITABLE k-COLORING and ℓ-BOUNDED k-COLORING were also examples of such problems. Both problems can be solved in polynomial time on trees and forests [9,2,17], i.e. graphs of treewidth 1 and the existence of a polynomial time algorithm for graphs of treewidth ≥ 2 was an open question. In this paper we introduce the first polynomial time algorithm on graphs of bounded treewidth for both versions of coloring. Due to enormous exponents in the running time, our algorithm is mainly of theoretical interest. Our main technique is quite far from the standard dynamic programming on graphs of bounded treewidth. To convince the reader (and ourselves) that the standard dynamic programming approach is unlikely to be implemented for EQUITABLE k-COLORING on graphs of bounded treewidth, we prove that a precolored version

* Fedor Fomin is supported by Norges forskningsråd projects 160233/V30 and 160778/V30.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 180–190, 2004.
© Springer-Verlag Berlin Heidelberg 2004

of the problem is NP-complete on trees, hence on graphs of treewidth 1. The main idea behind our polynomial time algorithm is to use recent combinatorial results of Kostochka et al. [21] that allow us to handle graphs with 'large' vertex degrees separately.

Previous results. The EQUITABLE k-COLORING problem has a long history. The celebrated theorem of Hajnal & Szemerédi [13] says that any graph G has an equitable k-coloring for $k \geq \Delta(G)+1$. This bound is sharp. One of the directions of research in this field was to obtain better upper bounds than $\Delta(G)+1$ for special graph classes. See the survey [23] for a review of the results in this field.

The k-COLORING problem can be trivially reduced to EQUITABLE k-COLORING problem and thus EQUITABLE k-COLORING is NP-hard. Polynomial time algorithms are known for split graphs [8] and trees [9].

ℓ-BOUNDED k-COLORING has a number of applications. It is also known as the MUTUAL EXCLUSION SCHEDULING problem (MES) which is the problem of scheduling unit-time tasks non-preemptively on m processors subject to constraints, represented by a graph G, so that tasks represented by adjacent vertices in G must run in disjoint time intervals. This problem arises in load balancing the parallel solution of partial differential equations by domain decomposition. (See [2,26] for more information.) Also the problems of this form have been studied in the Operations Research literature [3,22]. Other applications are in scheduling in communication systems [15] and in constructing school timetables [19].

The ℓ-BOUNDED k-COLORING problem can be solved in polynomial time on split graphs, complements of interval graphs [24,8], forests and in linear time on trees [2,17]. This is almost all what is known about graph classes where the ℓ-BOUNDED k-COLORING problem is efficiently solvable. When one of the parameters ℓ or k is fixed the situation is different. For example, for fixed ℓ or k the problem is solved on cographs [4,24] and for fixed ℓ on bipartite graphs [4,14] and line graphs [1]. For $\ell = 2$ the problem is equivalent to the MAXIMUM MATCHING problem on the complement graphs and is polynomial. Notice that for fixed ℓ the problem can be expressed in the counting monadic second-order logic and for graphs of bounded treewidth linear time algorithm for *fixed* ℓ can be constructed [18]. When ℓ is not fixed (i.e. ℓ is part of the input) even for trees the situation is not simple and the question of existence of a polynomial time algorithm for trees [14] was open for several years.

The problem remains NP-complete on cographs, bipartite and interval graphs [4], on cocomparability graphs and fixed $\ell \geq 3$ [24], on complements of line graphs and fixed $\ell \geq 3$ [10], and on permutation graphs and $\ell \geq 6$ [16]. For $k = 3$ the problem is NP-complete on bipartite graphs [4].

Almost all NP-completeness results for ℓ-BOUNDED k-COLORING for different graph classes mentioned above can also be obtained for EQUITABLE k-COLORING by making use of the following observation.

Proposition 1. *A graph G on n vertices is ℓ-bounded k-colorable if and only if the graph G' obtained by taking disjoint union of G and an independent set of size $\ell k - n$ is equitable k-colorable.*

Our contribution. A standard dynamic programming approach for the COLORING problem needs to keep $O(w^k n)$ entries, where w is the treewidth of a graph and k is the number of colors. Since the chromatic number of a graph is at most $w+1$ this implies that the classical coloring problem can be solved in polynomial time on graphs of bounded treewidth. Clearly such a technique does not work for EQUITABLE k-COLORING because the number of colors in an equitable coloring is not bounded by a function of treewidth. For example, a star on n vertices has treewidth 1 and it cannot be equitable k-colored for any $k < (n-1)/2$. One of the indications that the complexity of EQUITABLE k-COLORING for graphs of bounded treewidth can be different from 'classical' is that by Proposition 1 and [4], the problem is NP-hard on cographs and thus on graphs of bounded clique-width. (Note that chromatic number is polynomial on graphs of bounded clique-width [20].)

However, one of the properties of equitable colorings making our approach possible is the phenomena observed first by Bollobás & Guy [5] for trees: 'Most' trees can be equitable 3-colored. In other words, for almost all trees the difference between the numbers of colors in an equitable coloring is not 'far' from the chromatic number. Recently Kostochka et al. [21] succeeded to generalize the result of Bollobás & Guy for degenerate graphs and our main contribution — the proof that EQUITABLE k-COLORING can be solved in polynomial time on graphs of bounded treewidth (Section 3) — strongly uses this result. Very roughly, we use the results of Kostochka et al. to establish the threshold when the problem is trivially solved and when it becomes to be solvable in polynomial time by dynamic programming developed in Section 2. In Section 4 we show that such an approach cannot be extended to PRECOLORED EQUITABLE k-COLORING by showing that the precolored version of the problem is NP-hard on trees.

1.1 Definitions

We denote by $G = (V,E)$ a finite, undirected, and simple graph. We usually use n to denote the number of vertices in G. For every nonempty $W \subseteq V$, the subgraph of G induced by W is denoted by $G[W]$. The maximum degree of G is $\Delta(G) := \max_{v \in V} d_G(v)$. A graph G is *d-degenerate* if each of its nonempty subgraphs has a vertex of degree at most d. A nonempty subset of vertices $I \subseteq V$ is *independent* in G if no two of its elements are adjacent in G.

Definition 1. *A* tree decomposition *of a graph $G = (V,E)$ is a pair $(\{X_i \mid i \in I\}, T = (I,F))$, with $\{X_i \mid i \in I\}$ a family of subsets of V and T a tree, such that*

- $\bigcup_{i \in I} X_i = V$.
- *For all $\{v,w\} \in E$, there is an $i \in I$ with $v,w \in X_i$.*
- *For all $i_0, i_1, i_2 \in I$: if i_1 is on the path from i_0 to i_2 in T, then $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$.*

The width *of tree decomposition $(\{X_i \mid i \in I\}, T = (I,F))$ is $\max_{i \in I} |X_i| - 1$. The* treewidth *of a graph G is the minimum width of a tree decomposition of G.*

Lemma 1 (Folklore). *Every graph on n vertices and of treewidth $\leq w$ has a vertex of degree at most w, and has at most wn edges.*

A *k-coloring* of the vertices of a graph $G = (V,E)$ is a partition $A_1, A_2, \ldots, A_k$ of V into independent sets (in which some of the A_j may be empty); the k sets A_j are called the *color classes* of the k-coloring. The *chromatic number* $\chi(G)$ is the minimum value k for which a k-coloring exists. A k-coloring $A_1, A_2, \ldots, A_k$ is *ℓ-bounded* if $|A_i| \leq l$, $1 \leq i \leq k$. A k-coloring $A_1, A_2, \ldots, A_k$ is *equitable* if for any $i, j \in \{1, 2, \ldots, k\}$, $|A_i| - |A_j| \leq 1$.

In the EQUITABLE k-COLORING problem, we are given a graph $G = (V,E)$, and an integer k, and we ask whether G has an equitable k-coloring.

Theorem 1 ([21]). *Every n-vertex d-degenerate graph G is equitable k-colorable for any $k \geq \max\{62d, 31d\frac{n}{n-\Delta(G)+1}\}$.*

Every graph of treewidth $\leq d$ is d-degenerate and Theorem 1 implies the following corollary.

Corollary 1. *Every n-vertex graph G of treewidth w is equitable k-colorable for any $k \geq \max\{62w, 31w\frac{n}{n-\Delta(G)+1}\}$.*

2 Covering by Equitable Independent Sets

Let $S \subseteq V$ be a set of vertices of a graph $G = (V, E)$. We say that S can be *covered by independent sets of sizes* $[n/k]$ if there is a set of subsets $A_i \subseteq V$, $i \in \{1, 2, \ldots, p\}$, $p \leq |S|$, such that

(i) For every $i \in \{1, 2, \ldots, p\}$, A_i is an independent set;
(ii) For every $i, j \in \{1, 2, \ldots, p\}$, $i \neq j$, $A_i \cap A_j = \emptyset$;
(iii) For every $i \in \{1, 2, \ldots, p\}$, either $|A_i| = \lceil n/k \rceil$, or $|A_i| = \lfloor n/k \rfloor$;
(iv) $S \subseteq \cup_{1 \leq i \leq p} A_i$.

Covering by independent sets is a natural generalization of an equitable coloring: A graph G has an equitable k-coloring if and only if V can be covered by independent sets of sizes $[n/k]$. We use the following observations in our proof.

Lemma 2. *Let $S \subseteq V$ be a vertex subset of a graph G.*

(a) If S can not be covered by independent sets of sizes $[n/k]$, the graph G is not equitable k-colorable;
(b) If S can be covered by p independent sets $A_1, \ldots, A_p$ of sizes $[n/k]$ and the graph $G' = G[V - \cup_{1 \leq i \leq p} A_i]$ is equitable $(k-p)$-colorable, the graph G is equitable k-colorable.

Proof. (a) Let $B_1, \ldots, B_k$ be the color classes of an equitable k-coloring of G. Consider the collection of sets $\{B_i \mid 1 \leq i \leq k,\ B_i \cap S \neq \emptyset\}$, and and number them $A_1, A_2, \ldots$.

(b) Use color classes $A_1, \ldots, A_p$, and partition the vertices of G' as in the equitable coloring into color classes $A_{p+1}, \ldots, A_k$. □

Let G be a graph of treewidth w. The next theorem implies that when the cardinality of $S \subseteq V$ or the number k is at most $f(w)$, where f is a function of w, the question if S can be covered by independent sets of sizes $\lceil n/k \rceil$ can be answered in polynomial time. Because there are graphs that need $\Omega(n)$ colors in an equitable coloring, Theorem 2 does not imply directly that for graphs of bounded treewidth the EQUITABLE k-COLORING problem can be solved in polynomial time.

Theorem 2. *Let $G = (V, E)$ be an n-vertex graph of treewidth $\leq w$, let S be a subset of V, and let k be an integer. When k or $|S|$ is bounded by a constant, one can either find in polynomial time a covering of S by independent sets of sizes $\lceil n/k \rceil$, or conclude that there is no such a covering.*

Proof. (Sketch.) This can be shown using standard dynamic programming techniques for graphs of bounded treewidth. Note that we can check for a covering of at most $\min\{k, |S|\}$ independent sets of sizes $\lceil n/k \rceil$. An algorithm comparable to those e.g. shown in [6,27], that also has different table entries / homomorphism classes when sets have different sizes solves the problem in polynomial time on graphs of bounded treewidth. □

3 Bounded Treewidth

The main result of this paper is the following theorem.

Theorem 3. *The* EQUITABLE k-COLORING *problem can be solved in polynomial time on graphs of bounded treewidth.*

Proof. Let $G = (V, E)$ be a graph of treewidth w and let k be an integer. To determine if G has an equitable k-coloring, we consider the following cases.

Case 1. $\Delta(G) \leq n/2 + 1$ and $k \geq 62w$. Since

$$\max_{0 \leq \Delta(G) \leq n/2+1} \frac{n}{n - \Delta(G) + 1} = 2,$$

we have that

$$k \geq 62w = \max\{62w, 2 \cdot 31w\} \geq \max\{62w, 31w \frac{n}{n - \Delta(G) + 1}\}$$

and by Corollary 1, G is equitable k-colorable.

Case 2. $\Delta(G) \leq n/2 + 1$ and $k \leq 62w$. In this case, it follows from Theorem 2 that the question whether G has an equitable k-coloring can be solved in polynomial time.

Case 3. $\Delta(G) > n/2 + 1$. Let $S \subset V$ be the set of vertices in G of degree $\geq n/2 + 2$. By Lemma 1, G has at most wn edges, so $|S| \leq 4w$. Thus by Theorem 2, it can be checked in polynomial time whether S can be covered

by independent sets of sizes $[n/k]$. If S cannot be covered, then by part (a) of Lemma 2, G has no equitable k-coloring.

Let $A_i \subset V$, $i \in \{1,2,\ldots,p\}$, $p \leq |S|$, be an equitable covering of S by independent sets of sizes $[n/k]$. We define a new graph $G' = G[V - \cup_{1\leq i\leq p} A_i]$. The maximum vertex degree in G' is at most $n/2+1$ and the treewidth of G' is $\leq w$. Graph G' has

$$n' = |V - \cup_{1\leq i\leq p} A_i| \geq n - p\left\lfloor \frac{n}{k} \right\rfloor \geq n - 4w(\frac{n}{k} - 1) > (1 - \frac{4w}{k})n$$

vertices.

Let $k' = k - p$. We need again case distinction.

Case A. $k' \geq \max\{62w, 31w\frac{n'}{n'-n/2}\}$. Then

$$k' \geq \max\{62w, 31w\frac{n'}{n' - n/2}\} \geq \max\{62w, 31w\frac{n'}{n' - \Delta(G') + 1}\}$$

and by Corollary 1, G' is equitable k'-colorable. By part (b) of Lemma 2, G has equitable k-coloring.

Case B. $k' < \max\{62w, 31w\frac{n'}{n'-n/2}\}$ and $k' < 62w$. Since $p \leq 4w$, we have that $k = k' + p < 66w$. Then by Theorem 2, the question whether G has an equitable k-coloring, can be solved in polynomial time.

Case C. $k' < \max\{62w, 31w\frac{n'}{n'-n/2}\}$ and $k' \geq 62w$. Then

$$k' < 31w\frac{n'}{n' - \frac{n}{2}} < 31w\frac{n}{(1 - \frac{4w}{k})n - \frac{n}{2}} = \frac{31w}{\frac{1}{2} - \frac{4w}{k}}. \tag{1}$$

Using $k = k' + p \geq 62w$, we have that

$$\frac{4w}{k} \leq \frac{4w}{62w} = \frac{2}{31}$$

and

$$\frac{1}{2} - \frac{4w}{k} \geq \frac{27}{62}. \tag{2}$$

By (1) and (2),

$$k' < 31w\frac{62}{27} < 72w$$

and we conclude that $k = k' + p \leq 76w$. Again, by Theorem 2 the question if G has an equitable k-coloring, can be solved in polynomial time. □

By Proposition 1, Theorem 3 implies directly that there is a polynomial time algorithm for the ℓ-BOUNDED k-COLORING problem restricted to graphs of bounded treewidth.

4 Equitable Coloring with Precoloring

For a graph $G = (V, E)$, a *precoloring* π of a subset $V' \subset V$ in k colors is a mapping $\pi: V' \to \{1, 2, \ldots, k\}$. We say that a coloring $A_1, A_2, \ldots, A_k$ of G *extends* the pre-coloring π if $u \in A_{\pi(u)}$ for every $u \in V'$. We consider the following problem: EQUITABLE k-COLORING WITH PRECOLORING: For a given graph G, integer k and a given precoloring π of G, determine whether there exists an equitable k-coloring of G extending π.

Theorem 4. EQUITABLE k-COLORING WITH PRECOLORING *is NP-complete on trees.*

Proof. We use a reduction from the problem

3-PARTITION

Instance: A set A of non-negative integers $a_1, \ldots, a_{3m}$, and a bound B, such that for all i with $1 \leq i \leq 3m$, $(B+1)/4 < a_i < B/2$ and $\sum_{1 \leq i \leq 3m} a_i = mB$.

Question: Can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{a_i \in A_j} a_i = B$ for every j with $1 \leq j \leq m$?

3-PARTITION is NP-complete in the strong sense (Problem SP15 in Garey & Johnson [12]).

Let the set $A = \{a_1, \ldots, a_{3m}\}$ and the bound B be an instance of 3-PARTITION. We construct a tree G and a precoloring of G such that G is equitable $(m+1)$-colorable if and only if A can be 3-partitioned.

For every $i \in \{1, 2, \ldots, m\}$ we define the set $N_i = \{1, 2, \ldots, m\} - \{i\}$ and the precolored star S_i as a star with one non-precolored central vertex v adjacent to $m-1$ leaves which are precolored with all colors from N_i. Thus vertex v can be colored only with color i or $m+1$. For every $i \in \{1, 2, \ldots, m\}$ and $j \in \{1, 2, \ldots, 3m\}$, we define the precolored tree $G_{i,j}$ as a tree obtained by taking the disjoint union of $a_j + 1$ precolored stars S_i and by making the central vertex v of one of them to be adjacent to the central vertices of the others a_j stars. We call the vertex v the central vertex of $G_{i,j}$. Thus $G_{i,j}$ has $m(a_j+1)$ vertices; for every color $\ell \in N_i$ there are (a_j+1) vertices of $G_{i,j}$ precolored with ℓ. Every $(m+1)$-coloring of $G_{i,j}$ either colors v with $m+1$ and the remaining a_j non-precolored vertices with i, or it colors v with i and the remaining a_j non-precolored vertices with $m+1$. (See Fig. 1.)

For every $j \in \{1, 2, \ldots, 3m\}$, we define a precolored tree G_j as follows: We take the disjoint union of precolored trees $G_{i,j}$, $i \in \{1, 2, \ldots, m\}$, add one vertex c_j adjacent to all central vertices of trees $G_{i,j}$ and add one leaf adjacent to c_j precolored with $m+1$. Thus G_j has $m^2(a_j+1)+2$ vertices; for every color $\ell \in \{1, 2, \ldots, m\}$ there are $(m-1)(a_j+1)$ vertices of G_j precolored with ℓ and there is 1 vertex precolored with $m+1$. Also in any coloring of G_j vertex c_j cannot be colored with $m+1$. Thus for every $(m+1)$-coloring of G_j the spectra of colors used on neighbors of c_j does not contain all colors from $\{1, 2, \ldots, m\}$.

Finally G is obtained by taking the disjoint union of precolored trees $G_1, G_2, \ldots, G_{3m}$ and an independent set of cardinality $3m(m-2)+B$ precolored

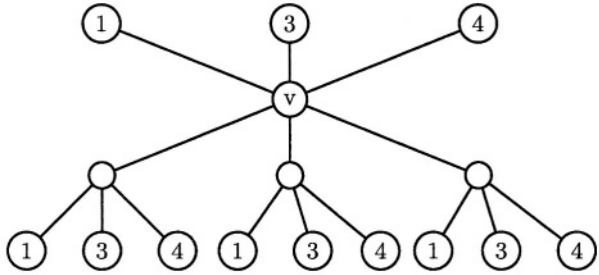


Fig. 1. Tree $G_{i,j}$. Here $a_j = 3$, $m = 4$, $i = 2$ and $N_2 = \{1, 3, 4\}$. In any 5-coloring of $G_{i,j}$, either v should be colored with 5 and all its non-precolored neighbors with 2, or v should be colored with 2 and the neighbors with 5.

with color $m+1$, and then making this forest a tree by adding appropriate edges between precolored vertices of the same color in different trees. Thus G has

$$\sum_{1 \le j \le 3m} m^2(a_j + 1) + 3(m-2) + B = m^2(mB + 3m) + 3(m-2) + B$$

vertices. For every color $\ell \in \{1, 2, \ldots, m\} - \{i\}$ there are

$$\sum_{1 \le j \le 3m} (m-1)(a_j + 1) = (m-1)(mB + 3m)$$

vertices of G precolored with ℓ and $3m(m-1) + B$ vertex are precolored with $m+1$.

Claim. A can be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{a_i \in A_j} a_i = B$, if and only if G has an equitable $(m+1)$-coloring that extends the precoloring.

Proof of Claim. Suppose that A can be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{a_i \in A_j} a_i = B$. We define an extension of the precoloring of G as follows. For every fixed $j \in \{1, 2, \ldots, 3m\}$, we choose i such that $a_j \in A_i$. We color the central vertex of $G_{i,j}$ with color $m+1$ and the remaining non-colored a_j vertices of $G_{i,j}$ with color i. In each graph $G_{\ell,j}$, $\ell \neq i$, a_j vertices are colored with $m+1$ and one vertex with ℓ. Also we color vertex c_j with i. Thus in every graph G_j on the set of non-precolored vertices color i is used $a_j + 1$ times. Any color $\ell \in \{1, 2, \ldots, m\} - \{i\}$ is used one time and color $m+1$ is used

$$a_j(m-1) + 1$$

times on non-precolored vertices. Thus in graph G the number of vertices colored with color $\ell \in \{1, 2, \ldots, m\}$ is

$$(m-1)(mB+3m) + \sum_{a_j \in A_\ell} (a_j+1) + \sum_{\{1 \le j \le 3m \mid a_j \notin A_\ell\}} 1 = (m-1)(mB+3m) + B + 3m.$$

The number of vertices colored with $m+1$ is

$$3m(m-2)+B+\sum_{1\leq j\leq 3m}(a_j(m-1)+1)=(m-1)(mB+3m)+B+3m$$

and we conclude that the obtained coloring is an equitable $(m+1)$-coloring.

Suppose now that G has an equitable $(m+1)$-coloring that extends the given precoloring. The main observation here is that for every $j \in \{1,2,\ldots,3m\}$, at most $a_j(m-1)+1$ vertices of a graph G_j are colored with $m+1$. (Otherwise, the coloring of the central vertices of graphs $G_{i,j}$, $i \in \{1,2,\ldots,m\}$, uses the whole spectra $\{1,2,\ldots,m\}$, thus leaving no space for the color of c_j.) If for some $j \in \{1,2,\ldots,3m\}$, less than $a_j(m-1)+1$ vertices of a graph G_j are colored with color $m+1$, then (the coloring is equitable) for some $j' \in \{1,2,\ldots,3m\}$, at least $a_j m$ vertices of the graph $G_{j'}$ are colored with $m+1$, which is a contradiction.

Thus we can conclude that for every $j \in \{1,2,\ldots,3m\}$, there is exactly one subgraph $G_{i,j}$ such that a_j non-precolored vertices of $G_{i,j}$ are colored with i. For all other $i' \in \{1,2,\ldots,m\}$, $i \neq i'$, a_j non-precolored vertices of $G_{i',j}$ are colored with $m+1$. We define

$$A_i = \{a_j \mid a_j \text{ non-precolored vertices of } G_{i,j} \text{ are colored with } i\}.$$

In G, the number of vertices colored with color $i \in \{1,2,\ldots,m\}$ is

$$(m-1)(mB+3m)+B+3m=(m-1)(mB+3m)+\sum_{a_j\in A_i}(a_j+1)+\sum_{\{1\leq j\leq 3m \mid a_j\notin A_i\}}1.$$

Thus for every $i \in \{1,2,\ldots,m\}$

$$\sum_{a_j\in A_i}(a_j+1)=B$$

and $A_1, A_2, \ldots, A_m$ is a 3-partition of A. $\triangle$

So we have a polynomial reduction from 3-PARTITION to EQUITABLE k-COLORING WITH PRECOLORING. As EQUITABLE k-COLORING WITH PRECOLORING trivially belongs to NP, we can conclude it is NP-complete. □

A direct corollary is that ℓ-BOUNDED k-COLORING WITH PRECOLORING is NP-complete for trees.

We can also formulate the related EQUITABLE COLORING WITH PRECOLORING problem. Here, we are given a precolored graph G, and ask whether there exists an integer k, and an equitable k-coloring of G that extends the precoloring. I.e., k is not part of the instance of the problem.

Proposition 2. EQUITABLE COLORING WITH PRECOLORING *is NP-complete on trees.*

Proof. Clearly, the problem is in NP. We transform from EQUITABLE k-COLORING WITH PRECOLORING on trees. Take a tree instance T of EQUITABLE k-COLORING WITH PRECOLORING. We may assume, by the proof above, that for each color i, $1 \leq i \leq k$, T has at least one precolored vertex with color i. Suppose T has n vertices. We construct a new tree as follows. For each i, $1 \leq i \leq k$, we add $n+1$ new vertices, precolored with i. Each of the new vertices is made adjacent to a precolored vertex in T with a different color. We obtain a new tree T'. T' cannot have an equitable r coloring for some $r \neq k$. (An equitable coloring must use at least k colors. It cannot use more than k colors, as we have at most $n-k$ non-precolored vertices, and at least $n+2$ vertices are already precolored with color 1.) Also, T' has an equitable k-coloring if and only if T has an equitable k-coloring. (Use the same colors for the non-precolored vertices in both cases.) So, NP-hardness follows. □

References

1. N. ALON, *A note on the decomposition of graphs into isomorphic matchings,* Acta Math. Hungar., 42 (1983), pp. 221–223.
2. B. S. BAKER AND E. G. COFFMAN, JR., *Mutual exclusion scheduling,* Theoret. Comput. Sci., 162 (1996), pp. 225–243.
3. J. BLAZEWICZ, K. H. ECKER, E. PESCH, G. SCHMIDT, AND J. WEGLARZ, *Scheduling Computer and Manufacturing Processes,* Springer, Berlin, 2001. 2nd ed.
4. H. L. BODLAENDER AND K. JANSEN, *Restrictions of graph partition problems. I,* Theoret. Comput. Sci., 148 (1995), pp. 93–109.
5. B. BOLLOBÁS AND R. K. GUY, *Equitable and proportional coloring of trees,* J. Combin. Theory Ser. B, 34 (1983), pp. 177–186.
6. R. B. BORIE, *Generation of polynomial-time algorithms for some optimization problems on tree-decomposable graphs,* Algorithmica, 14 (1995), pp. 123–137.
7. G. J. CHANG AND D. KUO, *The $L(2,1)$-labeling problem on graphs,* SIAM J. Discrete Math., 9 (1996), pp. 309–316.
8. B.-L. CHEN, M.-T. KO, AND K.-W. LIH, *Equitable and m-bounded coloring of split graphs,* in Combinatorics and computer science, vol. 1120 of Lecture Notes in Comput. Sci., Springer, Berlin, 1996, pp. 1–5.
9. B.-L. CHEN AND K.-W. LIH, *Equitable coloring of trees,* J. Combin. Theory Ser. B, 61 (1994), pp. 83–87.
10. E. COHEN AND M. TARSI, *NP-completeness of graph decomposition problems,* J. Complexity, 7 (1991), pp. 200–212.
11. M. R. GAREY, R. L. GRAHAM, D. S. JOHNSON, AND D. E. KNUTH, *Complexity results for bandwidth minimization,* SIAM J. Appl. Math., 34 (1978), pp. 477–495.
12. M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability, A guide to the theory of NP-completeness,* W. H. Freeman and Co., San Francisco, Calif., 1979.
13. A. HAJNAL AND E. SZEMERÉDI, *Proof of a conjecture of P. Erdős,* in Combinatorial theory and its applications, II, North-Holland, Amsterdam, 1970, pp. 601–623.
14. P. HANSEN, A. HERTZ, AND J. KUPLINSKY, *Bounded vertex colorings of graphs,* Discrete Math., 111 (1993), pp. 305–312.
15. S. IRANI AND V. LEUNG, *Scheduling with conflicts, and applications to traffic signal control,* in Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96), New York, 1996, ACM, pp. 85–94.

16. K. JANSEN, *The mutual exclusion scheduling problem for permutation and comparability graphs,* Information and Computation, 180 (2003), pp. 71–81.
17. M. JARVIS AND B. ZHOU, *Bounded vertex coloring of trees,* Discrete Math., 232 (2001), pp. 145–151.
18. D. KALLER, A. GUPTA, AND T. SHERMER, *The χ_t-coloring problem,* in Symp. on Theoretical Aspects of Comp. Sc. (STACS'95), vol. 900 of Lecture Notes in Comput. Sci., Springer, Berlin, 1995, pp. 409–420.
19. F. KITAGAWA AND H. IKEDA, *An existential problem of a weight-controlled subset and its application to school timetable construction,* Discrete Math., 72 (1988), pp. 195–211.
20. D. KOBLER AND U. ROTICS, *Edge dominating set and colorings on graphs with fixed clique-width,* Discrete Appl. Math., 126 (2003), pp. 197–221.
21. A. V. KOSTOCHKA, K. NAKPRASIT, AND S. V. PEMMARAJU, *Coloring d-degenarate graphs equitable.* Manuscript submitted in November 2003.
22. J. KRARUP AND D. DE WERRA, *Chromatic optimisation: limitations, objectives, uses, references,* European J. Oper. Res., 11 (1982), pp. 1–19.
23. K.-W. LIH, *The equitable coloring of graphs,* in Handbook of Combinatorial Optimization, Vol. 3, Kluwer Acad. Publ., Boston, MA, 1998, pp. 543–566.
24. Z. LONC, *On complexity of some chain and antichain partition problems,* in Graph-theoretic concepts in computer science (WG'91), vol. 570 of Lecture Notes in Comput. Sci., Springer, Berlin, 1992, pp. 97–104.
25. B. MONIEN, *The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete,* SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 505–512.
26. B. F. SMITH, P. E. BJØRSTAD, AND W. D. GROPP, *Domain Decomposition. Parallel multilevel methods for elliptic partial differential equations,* Cambridge University Press, Cambridge, 1996.
27. J. A. TELLE AND A. PROSKUROWSKI, *Algorithms for vertex partitioning problems on partial k-trees,* SIAM J. Discrete Math., 10 (1997), pp. 529 – 550.

The Bidimensional Theory of Bounded-Genus Graphs*

Erik D. Demaine[1], MohammadTaghi Hajiaghayi[1], and Dimitrios M. Thilikos[2]

[1] MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, Massachusetts 02139, USA,
{edemaine,hajiagha}@mit.edu
[2] Departament de Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya, Campus Nord – Mòdul C5,
c/Jordi Girona Salgado 1-3, E-08034, Barcelona, Spain,
sedthilk@lsi.upc.es

Abstract. Bidimensionality is a powerful tool for developing subexponential fixed-parameter algorithms for combinatorial optimization problems on graph families that exclude a minor. This paper completes the theory of bidimensionality for graphs of bounded genus (which is a minor-excluding family). Specifically we show that, for any problem whose solution value does not increase under contractions and whose solution value is large on a grid graph augmented by a bounded number of handles, the treewidth of any bounded-genus graph is at most a constant factor larger than the square root of the problem's solution value on that graph. Such bidimensional problems include vertex cover, feedback vertex set, minimum maximal matching, dominating set, edge dominating set, r-dominating set, connected dominating set, planar set cover, and diameter. This result has many algorithmic and combinatorial consequences. On the algorithmic side, by showing that an augmented grid is the prototype bounded-genus graph, we generalize and simplify many existing algorithms for such problems in graph classes excluding a minor. On the combinatorial side, our result is a step toward a theory of graph contractions analogous to the seminal theory of graph minors by Robertson and Seymour.

1 Introduction

The recent theory of fixed-parameter algorithms and parameterized complexity [13] has attracted much attention in its less than 10 years of existence. In general the goal is to understand when NP-hard problems have algorithms that are exponential only in a parameter k of the problem instead of the problem size n. Fixed-parameter algorithms whose running time is polynomial for fixed parameter values—or more precisely $f(k) \cdot n^{O(1)}$ for some (superpolynomial)

* The last author was supported by the EU within the 6th Framework Programme under contract 001907 (DELIS) and by the Spanish CICYT project TIC-2002-04498-C05-03 (TRACER)

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 191–203, 2004.
© Springer-Verlag Berlin Heidelberg 2004

function $f(k)$—make these problems efficiently solvable whenever the parameter k is reasonably small.

In the last three years, several researchers have obtained exponential speedups in fixed-parameter algorithms for various problems on several classes of graphs. While most previous fixed-parameter algorithms have a running time of $2^{O(k)}n^{O(1)}$ or worse, the exponential speedups result in subexponential algorithms with typical running times of $2^{O(\sqrt{k})}n^{O(1)}$. For example, the first fixed-parameter algorithm for finding a dominating set of size k in planar graphs [2] has running time $O(8^k n)$; subsequently, a sequence of subexponential algorithms and improvements have been obtained, starting with running time $O(4^{6\sqrt{34k}}n)$ [1], then $O(2^{27\sqrt{k}}n)$ [16], and finally $O(2^{15.13\sqrt{k}}k+n^3+k^4)$ [14]. Other subexponential algorithms for other domination and covering problems on planar graphs have also been obtained [1,3,5,17,15].

All subexponential fixed-parameter algorithms developed so far are based on showing a "treewidth-parameter bound": any graph whose optimal solution has value k has treewidth at most some function $f(k)$. In many cases, $f(k)$ is sublinear in k, often $O(\sqrt{k})$. Combined with algorithms that are singly exponential in treewidth and polynomial in problem size, such a bound immediately leads to subexponential fixed-parameter algorithms.

A series of papers [7,8,6] introduce the notion of *bidimensionality* as a general approach for obtaining treewidth-parameter bounds and therefore subexponential algorithms. This theory captures essentially all subexponential algorithms obtained so far. Roughly speaking, a parameterized problem is *bidimensional* if the parameter is large in a "grid-like graph" (linear in the number of vertices) and either closed under contractions *(contraction-bidimensional)* or closed under minors (*minor-bidimensional*). Examples of bidimensional problems include vertex cover, feedback vertex set, minimum maximal matching, dominating set, edge dominating set, r-dominating set, connected dominating set, planar set cover, and diameter. Diameter is a simple computational problem, but its bidimensionality has important consequences as it forms the basis of locally bounded treewidth for minor-closed graph families [9].

Treewidth-parameter bounds have been established for all minor-bidimensional problems in H-minor-free graphs for any fixed graph H [8,6]. In this case, the notion of "grid-like graph" is precisely the regular $r \times r$ square grid. However, contraction-bidimensional problems (such as dominating set) have proved substantially harder. In particular, the largest class of graphs for which a treewidth-parameter bound can be obtained is apex-minor-free graphs instead of general H-minor-free graphs [6]. ("Apex-minor-free" means "H-minor-free" where H is a graph in which the removal of one vertex leaves a planar graph.) Such a treewidth-parameter bound has been obtained for all contraction-bidimensional problems in apex-minor-free-graphs [6]. In this case, the notion of "grid-like graph" is an $r \times r$ grid augmented to have, for each vertex, $O(1)$ edges from that vertex to nonboundary vertices. (Here $O(1)$ depends on H.) Unfortunately, this treewidth-parameter bound is large: $f(k) = (\sqrt{k})^{O(\sqrt{k})}$. For a subexponential algorithm, we essentially need $f(k) = o(k)$. For apex-minor-free graphs, such a

bound is known only for the special cases of dominating set and vertex cover [10,8].

The biggest graph classes for which we know a sublinear (indeed, $O(\sqrt{k})$) treewidth-parameter bound for many contraction-bidimensional problems are single-crossing-minor-free graphs and bounded-genus graphs. ("Single-crossing-minor-free" means "H-minor-free" where H can be drawn in the plane with one crossing.) For single-crossing-minor-free graphs [12,11] (in particular, planar graphs [7]), all contraction-bidimensional problems have a bound of $f(k) = O(\sqrt{k})$. In this case, the notion of "grid-like graph" is an $r \times r$ grid partially triangulated by additional edges that preserve planarity. For bounded-genus graphs [8], a bound of $f(k) = O(\sqrt{k})$ has been shown, for the same notion of "grid-like graphs", but only for contraction-bidimensional problems with an additional property called *α-splittability*: upon splitting a vertex, the parameter should increase by at most $\alpha = O(1)$ (or decrease).

In this paper we complete the theory of bidimensionality for bounded-genus graphs by establishing a sublinear ($f(k) = O(\sqrt{k})$) treewidth-parameter bound for general contraction-bidimensional problems in bounded-genus graphs. Our notion of "grid-like graph" is somewhat broader: a partially triangulated $r \times r$ grid (as above) with up to g additional edges ("handles"), where g is the genus of the original graph. This form of contraction-bidimensionality is more general than α-splittability,[1] and thus we generalize the results for α-splittable contraction-bidimensional problems from [8]. It is easy to construct a parameter that is contraction-bidimensional but not α-splittable, although these parameters are not "natural". So far all "natural" contraction-bidimensional parameters we have encountered are α-splittable, though we expect other interesting problems to arise that violate α-splittability.

Our results show that a partially triangulated grid with g additional edges is the prototype graph of genus g, as observed by Lovász [18]. At a high level, this property means that, to solve an (algorithmic or combinatorial) problem on a general graph of genus g, the "hardest" instance on which we should focus is the prototype graph. This property generalizes the well-known result in graph theory that the grid is the prototype planar graph. This also completes our theory of constructing such prototypes for bidimensional problems.

Further algorithmic applications of our results follow from the graph-minor theory of Robertson and Seymour (e.g., [19]) and its extensions [8,10]. In particular, [23,8] shows how to reduce many problems on general H-minor-free graphs to subproblems on bounded-genus graphs. Essentially, the difference between bounded-genus graphs and H-minor-free graphs are "apices" and "vortices", which are usually not an algorithmic barrier. Applying our new theory for bounded-genus graphs, we generalize the algorithmic extensions of [8,10]. Indeed, we simplify the approaches of both [8] and [10], where it was necessary to "split" bounded-genus graphs into essentially planar graphs because of a lack

[1] This statement is the contrapositive of the following property: if the parameter is k for the partially triangulated grid with g additional edges, then by α-splitting the additional edges, the parameter is at most $k + \alpha g$ on the partially triangulated grid.

of general understanding of bounded-genus graphs. Specifically, we remove the necessity of Lemmas 7.4–7.8 in [10].

Last but not least are the combinatorial aspects of our results. In a series of 23 papers (so far), Robertson and Seymour (e.g., [19]) developed the seminal theory of graphs excluding a minor, which has had many algorithmic and combinatorial applications. Our completed understanding of contraction-bidimensional parameters can be viewed as a step toward generalizing the theory of graph minors to a theory of graph contractions. Specifically, we show that any graph of genus g can be contracted to its core of a partially triangulated grid with at most g additional edges; this result generalizes an analogous result from [23] when permitting arbitrary minor operations (contractions and edge deletions). Avoiding edge deletions in this sense is particularly important for algorithmic applications because many parameters are not closed under edge deletions, while many parameters are closed under contraction.

2 Preliminaries

All the graphs in this paper are undirected without loops or multiple edges. Given a graph G, we denote by $V(G)$ the set of its vertices and by $E(G)$ the set of its edges. For any vertex $v \in V(G)$ we denote by E_v the set of edges incident to v. Moreover, we use the notation $N_G(v)$ (or simply $N(v)$) for the set of neighbors of v in G (i.e., vertices adjacent to v).

Given an edge $e = \{x, y\}$ of a graph G, the graph obtained from G by contracting the edge e *is* the graph we get if we identify the vertices x and y and remove all loops and duplicate edges. A graph H obtained by a sequence of edge-contractions is said to be a *contraction* of G. A graph class $\mathcal{C}$ is a *contraction-closed* class if any contraction of any graph in $\mathcal{C}$ is also a member of $\mathcal{C}$. A contraction-closed graph class $\mathcal{C}$ is *H-contraction-free* if $H \notin \mathcal{C}$. Given any graph class $\mathcal{H}$, we say that a contraction-closed graph class $\mathcal{C}$ is *$\mathcal{H}$-contraction-free* if $\mathcal{C}$ is H-contraction-free for *any* $H \in \mathcal{H}$.

2.1 Treewidth and Branchwidth

A *branch decomposition* of a graph (or a hypergraph) G is a pair (T, τ), where T is a tree with vertices of degree 1 or 3 and τ is a bijection from the set of leaves of T to $E(G)$. The *order* of an edge e in T is the number of vertices $v \in V(G)$ such that there are leaves t_1, t_2 in T in different components of $T(V(T), E(T) - e)$ with $\tau(t_1)$ and $\tau(t_2)$ both containing v as an endpoint. The *width* of (T, τ) is the maximum order over all edges of T, and the *branchwidth* of G, $\mathbf{bw}(G)$, is the minimum width over all branch decompositions of G. (In case where $|E(G)| \leq 1$, we define the branch-width to be 0; if $|E(G)| = 0$, then G has no branch decomposition; if $|E(G)| = 1$, then G has a branch decomposition consisting of a tree with one vertex – the width of this branch decomposition is considered to be 0). The *treewidth* $\mathbf{tw}(G)$ of a graph G is a notion related to branchwidth. We need only the following relation:

Lemma 1 ([22]). *For any connected graph G where $|E(G)| \geq 3$, $\mathbf{bw}(G) \leq \mathbf{tw}(G)+1 \leq \frac{3}{2}\mathbf{bw}(G)$.*

The main combinatorial result of this paper determines, for any k and g, a family of graphs $\mathcal{H}_{k,g}$ such that any $\mathcal{H}$-contraction-free graph G with genus g will have branchwidth $O(gk)$. To describe such a family, we will need some definitions on graph embeddings.

2.2 Graph Embeddings

A *surface* Σ is a compact 2-manifold without boundary. We will always consider connected surfaces. We denote by the $\mathbb{S}_0$ the sphere $(x, y, z \mid x^2 + y^2 + z^2 = 1)$. A *line* in Σ is subset homeomorphic to [0, 1]. An O-arc is a subset of Σ homeomorphic to a circle. Let G be a graph 2-cell embedded in Σ. To simplify notations we do not distinguish between a vertex of G and the point of Σ used in the drawing to represent the vertex or between an edge and the line representing it. We also consider G as the union of the points corresponding to its vertices and edges. That way, a subgraph H of G can be seen as a graph H where $H \subseteq G$. A *face* of G is a connected component of $\Sigma - E(G) - V(G)$. (Every face is an open set.) We use the notation $V(G)$, $E(G)$, and $F(G)$ for the set of the vertices, edges, and faces of G. For $\Delta \subseteq \Sigma$, $\overline{\Delta}$ is the *closure* of Δ. The boundary of Δ is $\mathbf{bd}(\Delta) = \overline{\Delta} \cap \overline{\Sigma - \Delta}$ and the interior is $\mathbf{int}(\Delta) = \overline{\Delta} - \mathbf{bd}(\Delta)$.

A subset of Σ meeting the drawing only in vertices of G is called *G-normal*. If an O-arc is G-normal then we call it *noose*. The length of a noose is the number of its vertices.

Representativity [21] is the measure of how dense is the embedding of a graph in a surface. The *representativity* (or *face-width*) $\mathbf{rep}(G)$ of a graph G embedded in surface $\Sigma \neq \mathbb{S}_0$ is the smallest length of a noncontractible noose in Σ. In other words, $\mathbf{rep}(G)$ is the smallest number k such that Σ contains a noncontractible (non null-homotopic in Σ) closed curve that intersects G in k points.

It is more convenient to work with Euler genus. The *Euler genus* $\mathbf{eg}(\Sigma)$ of a surface Σ is equal to the non-orientable genus $\tilde{g}(\Sigma)$ (or the crosscap number) if Σ is a non-orientable surface. If Σ is an orientable surface, $\mathbf{eg}(\Sigma)$ is $2g(\Sigma)$, where $g(\Sigma)$ is the orientable genus of Σ. Given a graph G its Euler genus $\mathbf{eg}(G)$ is the minimum $\mathbf{eg}(\Sigma)$ where Σ is a surface where G can be embedded.

2.3 Splitting Graphs and Surfaces

In this section we describe precisely how to cut along a noncontractible noose in order to decrease the genus of the graph until we obtain a planar graph.

Let G be a graph and let $v \in V(G)$. Also suppose we have a partition $\mathcal{P}_v = (N_1, N_2)$ of the set of the neighbors of v. Define the *splitting* of G with respect to v and P_v to be the graph obtained from G by (i) removing v and its incident edges; (ii) introducing two new vertices v^1, v^2; and (iii) connecting v^i with the vertices in $N_i, i = 1, 2$. If H is the result of the consecutive application of the above operation on some graph G then we say that H is a *splitting* of G.

If additionally in such a splitting process we do not split vertices that are results of previous splittings then we say that H is a *fair splitting* of G.

The following lemma defines how to find a fair splitting for a given noncontractible noose. It will serve as a link between Lemma 8 and Lemma 10 in the proof of the main result of this paper.

Lemma 2. *Let G be a connected graph 2-cell embedded in a nonplanar surface Σ, and let N be a noncontractible noose of Σ. Then there is a fair splitting G' of G affecting the set $S = (v_1, \ldots, v_\rho)$ of the vertices of G met by N, such that (i) G' has at most two connected components, (ii) each connected component of G' can be 2-cell embedded in a surface with Euler genus strictly smaller than the Euler genus of Σ, and (iii) there are two faces f_1 and f_2, each in the 2-cell embedding of a connected component of G' (and the connected components are different for the two faces if G' is disconnected), such that the boundary of f_i, for $i \in \{1, 2\}$, contains $S_i = (v_1^i, \ldots, v_\rho^i)$ where v_j^1 and v_j^2 are the vertices created after the splitting of the vertex v_j, for $j = 1, \ldots, \rho$.*

3 Incomplete Embeddings and Their Properties

In this section we give a series of definitions and results that support the proof of the main theorem of the next section. In particular, we will need special embeddings of graphs that are incomplete, i.e., only some of the edges and vertices of the graph are embedded in a surface. Moreover, we will extend the definition of a contraction so that it will also consider contractions of faces for the part of the graph that is embedded.

Let Σ be a surface (orientable or not). Given a graph G, a vertex set $V \subseteq V(G)$ and an edge set $E \subseteq E(G)$ such that $\cup_{v \in V} E_v \subseteq E$, we say that G is (V, E)-embeddable in Σ if the graph G^- obtained by G if we remove from it all the vertices in V and all the edges in E, i.e., the graph $G^- = (V(G) - V, E(G) - E)$ has a 2-cell embedding in Σ. We call the graph G^- *ground* of G and we call the edges and vertices of G^- *landed*. In contrary, we call the vertices in V and E *flying*. Notice that the flying edges are partitioned into three categories: those that have both endpoints in $V(G) - V$ (we call them *bridges*), those with one endpoint in $V(G) - V$ and one endpoint in V (we call them *pillars*), and those with both endpoints in V (we call them *clouds*). From now on, whenever we refer to a graph (V, E)-embeddable in Σ we will accompany it with the corresponding 2-cell embedding of G^- in Σ.

The set of *atoms* of G with respect to some (V, E)-embedding of G in Σ is the set $A(G) = V(G) \cup E(G) \cup F(G)$ where $F(G)$ is the set of faces of the 2-cell embedding of G^- in Σ. Notice that a flying atom can only be a vertex or an edge. In this paper, we will consider the faces as open sets whose borders are cyclic sequences of edges and vertices.

3.1 Contraction Mappings

A strengthening of a graph being a contraction of another graph is for there to be a "contraction mapping" which preserves some aspects of the embedding in a

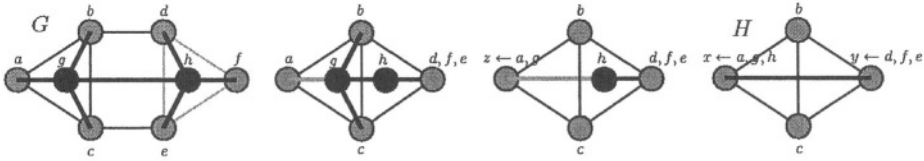


Fig. 1. An example of a contraction of a graph (V, E)-embeddable in $\mathbb{S}_0$. The contraction is shown in a three-step sequence: contracting the edges of the face $\{d, e, f\}$, then the edge $\{a, g\}$, and then edge $\{z, h\}$. A contraction mapping from G to H is defined as follows: $\phi(a)=\phi(g)=\phi(h)=\phi(\{a,g\})=\phi(\{g,h\})=x$, $\phi(b)=b$, $\phi(c)=c$, $\phi(d)=\phi(f)=\phi(e)=\phi(\{f,d\})=\phi(\{d,e\})=\phi(\{e,f\})=\phi(\{d,e,f\})=y$, $\phi(\{a,b\})=\phi(\{g,b\})=\{x,b\}$, $\phi(\{a,c\})=\phi(\{g,c\})=\{x,c\}$, $\phi(\{b,c\})=\{b,c\}$, $\phi(\{b,d\})=\{b,y\}$, $\phi(c,e)=\{c,y\}$, $\phi(\{a,b,c\})=\{x,b,c\}$, $\phi(\{b,d,e,c\})=\{b,c,y\}$, $\phi(\{h,d\})=\phi(\{h,e\})=\phi(\{h,f\})=\{x,y\}$, $\phi(\{a,b,d,f,e,c\})=\{x,b,y,c\}$.

surface during the contractions. See Fig. 1 for an example. Given two graphs G and H that are (V_G, E_G)- and (V_H, E_H)-embeddable in Σ and Σ', respectively, we say that $\phi : A(G) \to A(H)$ is a *contraction mapping* from G to H with respect to their corresponding embeddings if the following conditions are satisfied:

1. For any $v \in V(G)$, $\phi(v) \in V(H)$.
2. For any $e \in E(G)$, $\phi(e) \in E(H) \cup V(H)$.
3. For any $f \in F(G)$, $\phi(f) \in F(H) \cup E(H) \cup V(H)$.
4. For any $v \in V(H)$, $G[\phi^{-1}(v)]$ is a connected subgraph of G.
5. $\{\phi^{-1}(v) \mid v \in V(H)\}$ is a partition of $V(G)$.
6. If $\phi(\{x, y\}) = v \in V(H)$ then $\phi(x) = \phi(y) = v$.
7. If $\phi(\{x, y\}) = e \in E(H)$ then $\{\phi(x), \phi(y)\} \in E(H)$.
8. If $f \in F(G)$ and $\phi(f) = v \in V(G)$ and $f = (x_0, \ldots, x_{r-1})$ then $\phi(\{x_i, x_{i+1}\} = \phi(x_i) = v$ for any $i = 0, \ldots, r-1$ (where indices are taken modulo r).
9. If $f \in F(G)$ and if $\phi(f) = e$ (an edge of H) then there are two edges of f contained in $\phi^{-1}(e)$.
10. If $f \in F(G)$ and if $\phi(f) = g$ (a face of H) then each edge of g is *landed* and is the image of some edge in f.

Notice that, from Conditions 1, 2, and 3, the preimages of the faces of H are faces of G. The following lemma is easy.

Lemma 3. *If there exists some contraction mapping from a graph G to a graph H with respect to some embedding of G and H, then H is a contraction of G.*

3.2 Properties of Contraction Mappings

It is important that the two notions (contraction and existence of a contraction mapping) are identical in the case where G and H have no flying atoms, i.e., $V_G = V_H = E_G = E_H = \emptyset$. We choose to work with contraction mappings instead of simple contractions because they include stronger information enough to build the induction argument of Lemma 10.

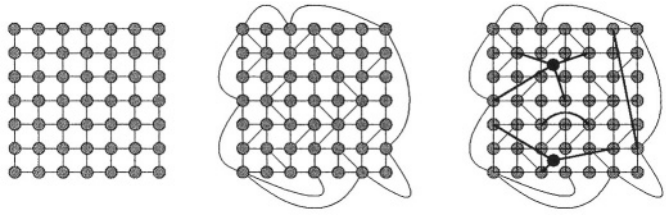


Fig. 2. A (7 × 7)-grid, a partially triangulated (7 × 7)-grid, and a (7, 9)-gridoid (the flying edges and vertices are the distinguished ones).

Lemma 4. *Let G be a graph (V, E)-embeddable on some surface Σ and let H be the graph occurring from G after contracting edges in $E(G^-)$. Then G[V] = H[V], H is also (V, E)-embeddable in Σ, and there exists a contraction mapping ϕ from G to H with respect to their corresponding embeddings.*

We omit the proof of this and other lemmas from this abstract.

3.3 Gridoids

A *partially triangulated* $(r \times r)$*-grid* is any graph that contains a $(r \times r)$-grid as a subgraph and is a subgraph of some triangulation of the same $(r \times r)$-grid.

We call a graph G a (r, k)*-gridoid* if it is (V, E)-embeddable in $\mathbb{S}_0$ for some pair V, E where $|E| \leq k$, $E(G[V]) = \emptyset$ (i.e., G does not have clouds), and so that G^- is a partial triangulated $(r' \times r')$-grid embedded on the $\mathbb{S}_0$ for some $r' \geq r$. For an example of a (7, 9)-gridoid and its construction; see Fig. 2.

4 Main Result

In this section we will prove that if a graph G has branchwidth more that $4k(\mathbf{eg}(G)+1)$ then G will contain as a contraction some $(k - 12\mathbf{eg}(G), \mathbf{eg}(\mathrm{G}))$-gridoid where $k \geq 12\mathbf{eg}(G)$.

4.1 Transformations of Gridoids

Lemma 5. *Let G be a (r, k)-gridoid $(\emptyset, E)$-embeddable in $\mathbb{S}_0$ and let $v \in V(G^-)$. Then there exists some contraction mapping ϕ from G to some $(r-4, k+1)$-gridoid $(\{v\}, E \cup \{\{v, y\}\})$-embeddable in $\mathbb{S}_0$ such that $\phi(v) = v$.*

Fig. 3 illustrates the contractions in one case, where v has degree 4.

Lemma 6. *Let G be a (r, k)-gridoid $(\emptyset, E)$-embeddable in $\mathbb{S}_0$ and let $\mathbf{e}$ be some of its flying edges. Then there exists some $(r-4, k)$-gridoid H $(\emptyset, E')$-embeddable in $\mathbb{S}_0$ for some E' and a contraction mapping ϕ of G to H such that $\phi(e) \in V(H)$.*

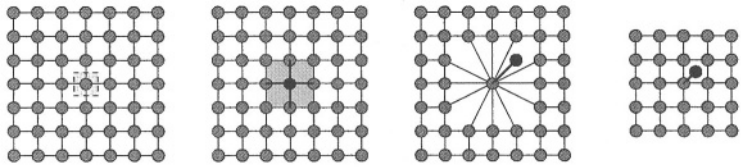


Fig. 3. An example of the first case in the proof of Lemma 5.

Lemma 7. *Let G be a (r,k)-gridoid $(\emptyset,E)$-embeddable in $\mathbb{S}_0$ and let a be some of its atoms. Then there exists some $(r-4,k)$-gridoid $(\emptyset,E)$-embeddable in $\mathbb{S}_0$ and a contraction mapping ϕ from G to H with respect to their corresponding embeddings such that $\phi(a) \in V(H)$.*

4.2 Excluding Gridoids as Contractions

Lemma 8. *Let G be a graph $(\emptyset,\emptyset)$-embeddable on some surface Σ. Let also H be a (r,k)-gridoid $(\emptyset,E)$-embeddable on the sphere and assume that ϕ is a contraction mapping from G to H with respect to their corresponding embeddings.*

Let now $\{v_1^i,\ldots,v_\rho^i\}, i=1,2$ be subsets of the vertices of two faces $f_i, i=1,2$ of the embedding of G where $f_1 \cap f_2 = \emptyset$ (we assume that the orderings of the indices in each subset respect the cyclic orderings of the vertices in $f_i, i=1,2$). Let G' be the graph obtained if we identify in G the vertex v_i^1 with the vertex v_i^2. Then, the following hold:

a. G' has some 2-cell embedding on a surface of bigger Euler genus.
b. There exist some $(r-12,k+1)$-gridoid H, $(\emptyset,E\cup\{\{e\}\})$-embeddable on the sphere such that there exists some contraction mapping from G' to H with respect their corresponding embeddings.

Proof. a. Let Σ be the surface where G is embedded. We define a surface Σ^- from Σ by removing the two "patches" defined by the (internal) points of the faces f_1 and f_2. Notice that G is still embeddable on Σ^- and that Σ^- is a surface with border whose connected components are the borders B_1, B_2 of the faces f_1 and f_2. We now construct a new surface from Σ^- by identifying the borders B_1 and B_2 in a way that v_i^1 is identified with v_i^2. Notice that the embedding that follows is a 2-cell embedding and that the new surface has bigger Euler genus.

b. From conditions 1, 2, and 3, $\phi(f_1)$ is either a vertex or an edge or a face of H. We apply Lemma 7 to construct a contraction mapping σ_1 from H to some $(r-4,k)$-gridoid H_1 where $\sigma_1(\phi(f_1)) \in V(H_1)$. Notice again that $\sigma_1(\phi(f_2))$ is either a vertex or an edge or a face of H_1. We again use Lemma 7 to construct a contraction mapping σ_2 from H_1 to some $(r-8,k)$-gridoid H_2 where $\sigma_2(\sigma_1(\phi(f_i))) = v_i \in V(H_2), i=1,2$. We now apply Lemma 5 for v_1 and construct some contraction mapping σ_3 from H_2 to some $(r-12,k+1)$-gridoid H_3, $(\{v_1\},E\cup\{\{v_1,y\}\})$-embeddable in $\mathbb{S}_0$ such that $\sigma_3(v_1)=v_1$. Summing up we have that $\phi' = \phi\circ\sigma_1\circ\sigma_2\circ\sigma_3$ is a map from G to H_3 with respect to the $(\emptyset,\emptyset)$-embedding of G on Σ and the $(\{v_1\},E\cup\{\{v_1,y\}\})$-embeddable of H_3 in

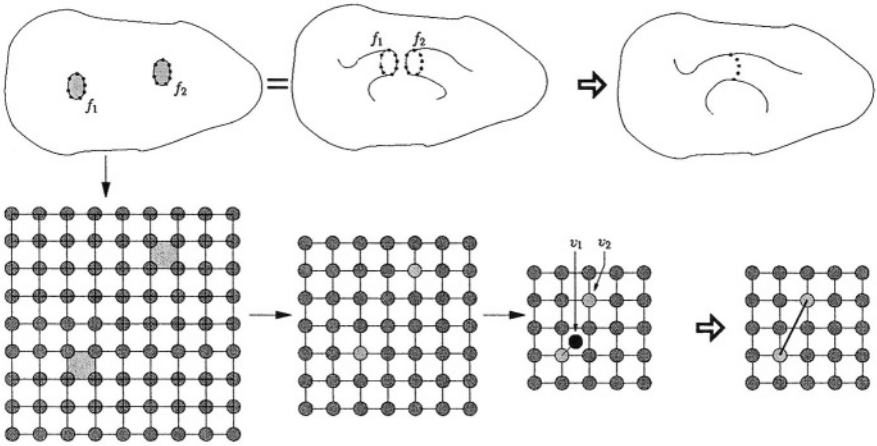


Fig. 4. An example of the transformations in the proof of Lemma 10.

$\mathbb{S}_0$. Moreover, we have that $\phi'(f_1) = v_1$ and $\phi'(f_2) = v_2 \in V(H_3)$ (to facilitate the notation we assume that $\sigma_3(v_2) = v_2$).

Notice now that if v is the result of the identification in H_3 of the vertex v_1 with the vertex v_2 we take a new graph H $(\emptyset, E \cup \{\{v, y\}\})$-embeddable in $\mathbb{S}_0$. Let A' be all the atoms of G that are not included in the faces f_1 and f_2. Notice that these atoms are not harmed while constructing G' from G and we set $\mu(a) = \phi'(a)$ for each $a \in A'$. Finally, for each atom $a \in A(G') - A$ we set $\mu(a) = v$. It now is easy to check that μ is a contraction mapping from G' to H with respect to their corresponding ernbeddings. As H is a $(r-12, k+1)$-gridoid we are done. □

The following is one of the main results in [8].

Theorem 1. *Let G be a graph 2-cell embedded in a non-planar surface Σ of representativity at least θ. Then one can contract edges in G to obtain partially triangulated $(\theta/4 \times \theta/4)$-grid.*

We also need the following easy lemma.

Lemma 9. *Let G be a graph and let H be the graph occurring from G after splitting some vertex $v \in V(G)$. Then* $\mathbf{bw}(H) \leq \mathbf{bw}(G) + 1$.

We are now ready to prove the central result of this section.

Lemma 10. *Let G be a graph $(\emptyset, \emptyset)$-embeddable on a surface Σ of Euler genus g and assume that* $\mathbf{bw}(G) \geq 4(r-12g)(g+1)$. *Then there exists some $(r-12g, g)$-gridoid H, $(\emptyset, E)$-embeddable in $\mathbb{S}_0$ such that there exists some contraction mapping from G to H with respect to their corresponding embeddings.*

Proof. First, if the graph G is disconnected, we discard all but one connected component C such that $\mathbf{bw}(C) = \mathbf{bw}(G)$.

We use induction on g. Clearly, if $g = 0$, G is a planar graph and after applying Lemma 3, the result follows from the planar exclusion theorem of RS. (The induction base use strongly the fact that for conventional embeddings the contraction relation is identical to our mapping.)

Suppose now that $g \geq 1$ and the theorem holds for any graph embeddable in a surface with Euler genus less than g. Refer to Fig. 4. If the representativity of G is at least $4(r-12g)$, then by Theorem 1 we can contract edges in G to obtain a partially triangulated $((r-12g) \times (r-12g))$-grid (with no additional edges), and we are done. Otherwise, the representativity of G is less than $4(r-12g)$. In this case, the smallest noncontractible noose has vertex set S of size less than $4(r-12g)$. Let G' be a splitting of G with respect to S as in Lemma 2. Recall that G' is now $(\emptyset,\emptyset)$-embeddable on a surface of Euler genus $g' \leq g-1$.

By Lemma 9, the branchwidth of G' is at least the branchwidth of G minus $|S|$. As $|S| \leq 4(r-12g)$ we have that $\mathbf{bw}(G') \geq 4(r-12g)(g+1) - 4(r-12g) = 4(r-12g)g \geq 4(r-12g)(g'+1)$. By the induction hypothesis there exist some $(r-12g', g')$-gridoid H', $(\emptyset, E)$-embeddable in $\mathbb{S}_0$ such that there exists some contraction mapping from G' to H with respect to their corresponding embeddings. From Lemma 8, there exist some $(r-12g'-12, g'+1)$-gridoid H, $(\emptyset, E \cup \{\{e\}\})$-embeddable on the sphere such that there exists some contraction mapping from G to H with respect to their corresponding embeddings. As $r-12g'-12 \geq r-12g$ and $g'+1 \leq g$, we are done. □

And we have the conclusion of this section.

Theorem 2. *If a graph G excludes all $(k-12\mathbf{eg}(G), \mathbf{eg}(G))$-gridoids as contractions, for some $k \geq 12\mathbf{eg}(G)$, then G has branchwidth at most $4k(\mathbf{eg}(G)+1)$.*

By Lemma 1 we can obtain a treewidth-parameter bound as desired.

5 Algorithmic Consequences

Define the *parameter* corresponding to an optimization problem to be the function mapping graphs to the solution value of the optimization problem. In particular, *deciding* a parameter corresponds to computing whether the solution value is at most a specified value k. A parameter is *contraction-bidimensional* if (1) its value does not increase under taking of contractions and (2) its value on a $(r, O(1))$-gridoid is $\Omega(r^2)$.[2]

Theorem 3. *Consider a contraction-bidimensional parameter P such that, given a tree decomposition of width at most w for a graph G, the parameter can be decided in $h(w) \cdot n^{O(1)}$ time. Then we can decide parameter P on a bounded-genus graph G in $h(O(\sqrt{k})) \cdot n^{O(1)} + 2^{O(\sqrt{k})} n^{3+\epsilon}$ time.*

[2] The requirement of $\Omega(r^2)$ can be weakened to allow any function $g(r)$, as in [8,6]; the only consequence is that $\sqrt{k}$ gets replaced by $g^{-1}(r)$.

Corollary 1. *Vertex cover, minimum maximal matching, dominating set, edge dominating set, r-dominating set (for fixed r), and clique-transversal set can be solved on bounded-genus graphs in $2^{O(\sqrt{k})}n^{3+\epsilon}$ time, where k is the size of the optimal solution. Feedback vertex set and connected dominating set can be solved on bounded-genus graphs in $2^{O(\sqrt{k}\log k)}n^{3+\epsilon}$ time.*

Acknowledgments. Thanks go to Fedor Fomin for early collaboration on this project, and to the anonymous referees for their helpful feedback on the paper.

References

1. J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica,* 33(4):461–493, 2002.
2. J. Alber, H. Fan, M. R. Fellows, H. Fernau, R. Niedermeier, F. A. Rosamond, and U. Stege. Refined search tree technique for DOMINATING SET on planar graphs. In *MFCS 2001,* LNCS 2136, pp. 111–122.
3. J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: Exponential speed-up for planar graph problems. *ECCC,* 8(23), 2001.
4. E. Amir. Efficient approximation for triangulation of minimum treewidth. In *UAI 2001,* pp. 7–15.
5. M.-S. Chang, T. Kloks, and C.-M. Lee. Maximum clique transversals. In *WG 2001,* pp. 300–310.
6. E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discr. Math.* To appear. A preliminary version appears in *LATIN 2004,* LNCS 2976, 2004, pp. 109–118.
7. E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos. Fixed-parameter algorithms for the (k,r)-center in planar graphs and map graphs. In *ICALP 2003,* pp. 829–844.
8. E. D. Demaine, F. V. Fomin, M. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. In *SODA 2004,* pp. 823–832.
9. E. D. Demaine and M. Hajiaghayi. Diameter and treewidth in minor-closed graph families, revisited. *Algorithmica.* To appear.
10. E. D. Demaine and M. Hajiaghayi. Equivalence of local treewidth and linear local treewidth and its algorithmic applications. In *SODA 2004,* PP. 833–842.
11. E. D. Demaine, M. Hajiaghayi, N. Nishimura, P. Ragde, and D. M. Thilikos. Approximation algorithms for classes of graphs excluding single-crossing graphs as minors. *J. Comput. System Sci.*. To appear.
12. E. D. Demaine, M. Hajiaghayi, and D. M. Thilikos. Exponential speedup of fixed parameter algorithms on $K_{3,3}$-minor-free or K_5-minor-free graphs. In *ISAAC 2002,* LNCS 2518, pp. 262–273.
13. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.
14. F. V. Fomin and D. M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. In *SODA 2003,* pp. 168–177.
15. G. Gutin, T. Kloks, and C. Lee. Kernels in planar digraphs. In *Optimization Online.* Mathematical Programming Society, 2001.

16. I. Kanj and L. Perković. Improved parameterized algorithms for planar dominating set. In *MFCS 2002,* LNCS 2420, pp. 399–410.
17. T. Kloks, C. M. Lee, and J. Liu. New algorithms for k-face cover, k-feedback vertex set, and k-disjoint set on plane and planar graphs. In *WG 2002,* LNCS 2573, pp. 282–296.
18. L. Lovász. Personal communication, Dec. 2003.
19. N. Robertson and P. D. Seymour. Graph minors — a survey. In *Surveys in Combinatorics,* pp. 153–171. Cambridge Univ. Press, 1985.
20. N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms,* 7(3):309–322, 1986.
21. N. Robertson and P. D. Seymour. Graph minors. VII. Disjoint paths on a surface. *J. Combin. Theory Ser. B,* 45(2):212–254, 1988.
22. N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory Ser. B,* 52(2):153–190, 1991.
23. N. Robertson and P. D. Seymour. Graph minors. XVI. Excluding a non-planar graph. *J. Combin. Theory Ser. B,* 89(1):43–76, 2003.

Parallel Knock-Out Schemes in Networks*

Hajo Broersma[1], Fedor V. Fomin[2], and Gerhard J. Woeginger[1]

[1] University of Twente, The Netherlands
{broersma,g.j.woeginger}@math.utwente.nl
[2] University of Bergen, Norway
fomin@ii.uib.no

Abstract. We consider parallel knock-out schemes, a procedure on graphs introduced by Lampert and Slater in 1997 in which each vertex eliminates exactly one of its neighbors in each round. We are considering cases in which after a finite number of rounds, where the minimimum number is called the parallel knock-out number, no vertices of the graph are left. We derive a number of combinatorial and algorithmical results on parallel knock-out numbers.

We observe that for families of sparse graphs (like planar graphs, or graphs with bounded tree-width), the parallel knock-out number grows at most logarithmically with the number n of vertices, which is basically tight for trees. Furthermore, we construct a family of bipartite graphs for which the parallel knock-out number grows proportionally to the square root of n. We characterize trees with parallel knock-out number at most 2, and show that the parallel knock-out number for trees can be computed in polynomial time via a dynamic programming approach, whereas the general problem is known to be NP-hard. Finally we show that claw-free graphs with minimum degree at least 2 have parallel knock-out number at most 2, and that the lower bound on the minimum degree is best possible.

1 Introduction

Lampert & Slater [2] introduced the following *parallel knock-out* procedure for graphs: On every vertex v of an undirected graph, there is a person standing. Every person selects one other person that stands on an adjacent vertex. Then all the selected persons are knocked out simultaneously, and the whole procedure is repeated with the surviving vertices. The game terminates, as soon as there are survivors that do not have any neighbor left to knock out. For instance, on the path $v_1 - v_2 - v_3 - v_4 - v_5$ it may happen that the persons on v_1 and v_3 both decide to knock out the person on v_2, that v_2 and v_4 both decide to knock out v_3, and that v_5 knocks out v_4. Then v_1 and v_5 are the survivors of this round, and the game terminates.

* Fedor Fomin is supported by Norges forskningsråd projects 160233/V30 and 160778/V30. Part of the work was done while Fedor Fomin was visiting the University of Twente.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 204–214, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Formally, let $G = (V, E)$ be an undirected, simple, loopless graph. We denote by $N(v)$ the set of all neighbors of vertex v (not including the vertex v itself). A *KO-selection* is a function $f : V \to V$ with $f(v) \in N(v)$ for all $v \in V$. If $f(v) = u$, we will sometimes say that vertex v *knocks out* or *eliminates* vertex u, or that (in the language of chip firing games) vertex v *fires at* vertex u. For a KO-selection f, we define the corresponding *KO-successor* G^f of G as the subgraph of G that is induced by the vertices in $V - f(V)$; this situation will be denoted by $G \rightsquigarrow G^f$. Note that every graph G without isolated vertices has at least one KO-successor.

In this paper, we are mainly interested in the question whether for a given graph G, there exists a sequence of KO-selections and KO-successors such that

$$G \rightsquigarrow G_1 \rightsquigarrow G_2 \rightsquigarrow \cdots \rightsquigarrow G_r = (\emptyset, \emptyset).$$

If no such sequence exists, then the *parallel knock-out number* of the graph G is infinite, and we write $\mathrm{PKO}(G) = \infty$. Otherwise, we define the parallel knock-out number $\mathrm{PKO}(G)$ of G as the smallest number r for which such a sequence exists. A graph G is called *KO-reducible* if and only if $\mathrm{PKO}(G)$ is finite.

A sequence of KO-selections that transform a KO-reducible graph into the empty graph is called a *KO-reduction scheme.* A single step in this sequence is called a *round* of the KO-reduction scheme.

It has been proved by Lampert & Slater [3] that it is an NP-complete problem to decide whether a given graph is KO-reducible. It is not difficult to check that a graph G has $\mathrm{PKO}(G) = 1$ if and only if G contains a spanning subgraph consisting of a number of mutually disjoint cycles and matching edges. This implies that the problem of deciding whether $\mathrm{PKO}(G) = 1$ is polynomially solvable; the equivalent formulation yields a folklore problem appearing in many standard books in combinatorial optimization. The equivalence also shows that the concept of the parallel knock-out number is related to well-known concepts like perfect matchings, 2-factors, and Hamiltonian cycles.

2 Results of This Paper

We derive a number of combinatorial and algorithmical results around parallel knock-out numbers. In Section 3 we observe that for families of sparse graphs (like planar graphs, or graphs with bounded tree-width), the parallel knock-out number grows at most logarithmically with the number n of vertices. Furthermore, we construct a family of bipartite graphs for which the parallel knock-out number grows proportionally to the square root of n. Our upper bound result on sparse graphs is basically tight for trees (up to a constant factor): Section 4 presents a corresponding lower bound construction. In Section 5 we characterize trees with parallel knock-out number at most 2. This involves a condition analogous to Hall's condition for the existence of matchings in bipartite graphs.

Section 6 investigates the algorithmical behavior of the parallel knock-out number for trees: It can be computed in polynomial time via a dynamic programming approach. This seems to be one of the rare cases where a dynamic

program for trees does not immediately carry over to the bounded tree-width classes: At the heart of our dynamic program for trees, there sits a certain bipartite matching problem; for higher tree-widths, this bipartite matching problem translates into something ugly. Section 7 turns to claw-free graphs: Under mild degree conditions ($\delta(G) \geq 2$), a claw-free graph G has parallel knock-out number at most 2, and the lower bound on the minimum degree is best possible.

3 Upper and Lower Bounds

Lemma 1. *Let $\tau > 0$ be a fixed real number. Let $\mathcal{G}$ be a class of graphs that satisfies the following two properties.*

- *$\mathcal{G}$ is closed under taking vertex-induced subgraphs.*
- *Every graph $G = (V, E)$ in $\mathcal{G}$ satisfies $|E| \leq \tau |V|$.*

Then any KO-reducible graph $G = (V, E)$ in $\mathcal{G}$ satisfies $\text{PKO}(G) \leq \lceil 4\tau \rceil \cdot \log_2 |V|$.

Proof. Consider a KO-reduction scheme for the n-vertex graph G. We claim that after $\lceil 4\tau \rceil$ rounds, the number of surviving vertices goes down by at least a factor of two. Suppose otherwise. Then for $\lceil 4\tau \rceil$ rounds, the number of vertices is at least $n/2$. In every round, each of these $n/2$ vertices fires along some edge, and every edge is used by at most two vertices. Hence, in every round at least $n/4$ edges are removed from the graph, and so after $\lceil 4\tau \rceil$ rounds the graph would be without edges. This proves our claim. The statement in the lemma now follows by induction. □

Lemma 1 can be used to get logarithmic upper bounds e.g. for planar graphs and for graphs of bounded tree-width.

Lemma 2. *Let $1 \leq a \leq b$. Then the complete bipartite graph $K_{a,b}$ is KO-reducible if and only if $b \leq \frac{1}{2}a(a+1)$. If this inequality is satisfied then*

$$\text{PKO}(K_{a,b}) = \left\lceil a + \frac{1}{2} - \sqrt{(a+\frac{1}{2})^2 - 2b} \right\rceil \tag{1}$$

This implies $\text{PKO}(K_{a,b}) < \sqrt{2(a+b)}$.

Proof. In $K_{a,b}$ the a vertices on one side of the bipartition will be called the *left* vertices, and the remaining vertices will be called the *right* vertices. The situation after r rounds of some KO-reduction scheme is fully specified by the number a_r of surviving left vertices and by the number b_r of surviving right vertices. The initial situation is described by $a_0 = a$ and $b_0 = b$. We will denote the expression in the right hand side of (1) by $F(a, b)$. We will first show that $\text{PKO}(K_{a,b}) \leq F(a, b)$ and then that $\text{PKO}(K_{a,b}) \geq F(a, b)$.

It can be verified that $F(a, b) = 1$ implies $a = b$. Hence, these cases indeed satisfy $\text{PKO}(K_{a,b}) \leq F(a, b)$, and from now on we will assume $F(a, b) \geq 2$. Let $k = F(a, b) - 2$, and consider the following KO-reduction scheme: In the first k

rounds, all right vertices fire at the same left vertex, and all left vertices fire at pairwise distinct right vertices. For $r = 1, \ldots, k$ this yields $a_r = a_{r-1} - 1$ and $b_r = b_{r-1} - a_{r-1}$, which is equivalent to $a_r = a - r$ and $b_r = b - r\,a + \frac{1}{2}r(r-1)$. It can be shown that $a_k < b_k \leq 2a_k - 1$ holds. In the $(k+1)$th round, all right vertices fire at the same left vertex, whereas the left vertices fire at $b_k - a_k + 1$ distinct right vertices. This yields $a_{k+1} = b_{k+1} = a_k - 1$. In the $(k+2)$th round, the left and right vertices fire at each other in pairs. This shows $\text{PKO}(K_{a,b}) \leq F(a,b)$.

Next, consider $K_{a,b}$ with $1 \leq a < b \leq \frac{1}{2}a(a+1)$. The first round of some KO-reduction scheme, we have $a_1 = a - x$ with $1 \leq x \leq a-1$ and $b_1 = b - y$ with $1 \leq y \leq a$. It can be shown that for all these cases either $F(a,b) \leq F(a_1, b_1) + 1$ holds, or that otherwise the resulting graph is not KO-reducible. □

We omit the proof of the following lemma.

Lemma 3. *If an n-vertex tree T is KO-reducible, then* $\text{PKO}(T) \leq \lceil \log_3 n \rceil$.

4 Trees with High Parallel Knock-Out Numbers

In this section we will construct trees with high parallel knock-out numbers. The construction is done inductively via the following two sequences $\langle Y_1, Y_2, \ldots \rangle$ and $\langle Z_1, Z_2, \ldots \rangle$ of rooted trees:

- The tree Y_1 consists of a root with one child.
- The tree Z_1 consists of a root with one child and one grandchild.
- For $\ell \geq 2$, the tree Y_ℓ consists of a root with ℓ children. The first $\ell - 2$ of these children are the roots of copies of the trees $Z_1, \ldots, Z_{\ell-2}$. The last two children are roots of copies of $Z_{\ell-1}$.
- For $\ell \geq 2$, the tree Z_ℓ consists of a root with ℓ children. These children are the roots of copies of the trees $Y_1, \ldots, Y_\ell$.

Lemma 4. *For every $\ell \geq 1$, there exists a sequence of ℓ KO-selections that transforms Y_ℓ into the empty graph. For every $\ell \geq 1$, there exists a sequence of ℓ KO-selections that transforms Z_ℓ into a one-vertex tree that consists of the root of Z_ℓ.*

Proof. We prove both statements simultaneously by induction on ℓ. For $\ell = 1$, we let the two vertices in Y_1 eliminate each other simultaneously, and we let the root of Z_1 fire at its child, and let the child and the grandchild fire at each other.

We use the following scheme for Y_ℓ with $\ell \geq 2$. The ℓ subtrees $Z_1, \ldots, Z_{\ell-2}$, $Z_{\ell-1}, Z_{\ell-1}$ that are attached at the children of the root use the KO-schemes that exist by the inductive claim. In round k with $1 \leq k \leq \ell - 1$, the root of Y_ℓ fires at the root of subtree Z_k (and makes this subtree disappear from the game). In the final round ℓ, the root of Y_ℓ and the root of the last subtree $Z_{\ell-1}$ eliminate each other simultaneously.

We use the following scheme for Z_ℓ with $\ell \geq 2$. The ℓ subtrees $Y_1, \ldots, Y_\ell$ that are attached at the children of the root use the KO-schemes that exist by

the inductive claim. In round k with $1 \leq k \leq \ell$, the root of Z_ℓ fires at the root of subtree Y_k and makes this subtree disappear from the game. After round ℓ, the root of Z_ℓ is the only survivor. □

Lemma 5. *Let T be a rooted tree that contains Y_ℓ (respectively Z_ℓ) as a rooted maximal subtree. Let $r(y)$ (respectively $r(z)$) denote the root of Y_ℓ (respectively, of Z_ℓ). Assume furthermore that T is KO-reducible. Then in every KO-reduction scheme for T the following statements hold:*

(a) In the first ℓ rounds, the root $r(y)$ of Y_ℓ fires at its ℓ children.
(b) In the ℓth round, the root $r(y)$ of Y_ℓ is eliminated by one of its ℓ children.
(c) The root $r(z)$ of Z_ℓ can not be eliminated by any of its children.
(d) If the root $r(z)$ of Z_ℓ is still alive in round $\ell+1$, then it will fire at its father.

Proof. We prove all four statements (a)–(d) simultaneously by induction on ℓ. In Y_1, the leaf fires at the root in the first round. The leaf can only be eliminated, if the root fires at it in the first round. This proves (a) and (b) for $\ell = 1$. In Z_1, the child and the grandchild must fire at each other in the first round. If the root survives the first round, it can only fire at its father in the second round. This proves (c) and (d) for $\ell = 1$.

Proofs of statements (a) and (b) for $\ell \geq 2$: The children of $r(y)$ form the roots of certain subtrees Z_k. By the inductive assumption of statement (c), these children can not be eliminated by their own children. Hence, they all must be eliminated by $r(y)$, and $r(y)$ must stay alive for the first ℓ rounds. By the inductive assumption of statement (d), the roots of Z_k $(1 \leq k \leq \ell - 2)$ must be eliminated in the first $\ell - 2$ rounds. In rounds $\ell - 1$ and ℓ, the roots of the two subtrees $Z_{\ell-1}$ must be eliminated; this proves (a). Moreover, in round ℓ the last surviving child of $r(y)$ eliminates $r(y)$; this proves (b).

Proofs of statements (c) and (d) for $\ell \geq 2$: The children of $r(z)$ form the roots of certain subtrees Y_k. By the inductive assumption of statements (a) and (b), these children only eliminate their own children, and eventually are eliminated by their own children. This proves (c). Since the root $r(z)$ is not eliminated by any of its children, it must be eliminated by its father. If $r(z)$ is still alive in round $\ell + 1$, it will fire at its last surviving neighbor, and this neighbor must be the father. This proves (d). □

According to Lemma 4 $\text{PKO}(Y_\ell) \leq \ell$, and according to Lemma 5.(b) $\text{PKO}(Y_\ell) \geq \ell$. This yields $\text{PKO}(Y_\ell) = \ell$. It can be shown by induction that Y_ℓ has at most $(2+\sqrt{2})^\ell/\sqrt{2}$ vertices, and that Z_ℓ has at most $(2+\sqrt{2})^\ell$ vertices.

Theorem 1. *For arbitrarily large n, there exist KO-reducible n-vertex trees T with $\text{PKO}(T) = \Omega(\log n)$.* □

5 Trees with Low Parallel Knock-Out Numbers

In this section we will characterize trees with parallel knock-out number at most 2. This involves a condition analogous to Hall's condition for the existence of matchings in bipartite graphs.

We will start with an easy but useful observation (we omit its proof here).

Lemma 6. *Let T be a tree that has no matching saturating all leaves of T. Then* $\text{PKO}(T) = \infty$.

Now suppose T is a KO-reducible tree, and choose a matching M of maximum cardinality subject to the condition that it saturates all leaves. It is obvious that $\text{PKO}(T) = 1$ if and only if M is a perfect matching. Assuming that M is not a perfect matching we consider the set U of unsaturated vertices. Clearly U in an independent set and by standard arguments from matching theory there are no M-alternating paths between pairs of vertices of U. For a vertex $u \in U$, a *u-triplet* is a P_3 in $T - U$ with the property that one of the end vertices of the P_3 is adjacent to u and the other end vertex has degree at least 2 in T. Thus this P_3 is an M-alternating path with two saturated end vertices in T. Let $T(u)$ denote the set of u-triplets of $u \in U$ in $T - U$. Note that a P_3 can be a u-triplet and a v-triplet for two distinct vertices u and v of U, but that in such cases u and v are adjacent to the same end vertex of the P_3; otherwise there is an M-alternating path between u and v, a contradiction. For a subset $S \subseteq U$ the set $T(S)$ of *S-triplets* is the union of all u-triplets for $u \in S$ in $T - U$. A tree T is called *Hall-perfect* if it has a matching saturating all leaves and for some maximum matching M with this property we have that either M is a perfect matching or

$$|T(S)| \geq |S| \quad \text{for all} \quad S \subseteq U, \tag{2}$$

where U is the set of unsaturated vertices of T.

Theorem 2. *A tree T is Hall-perfect if and only if* $\text{PKO}(T) \leq 2$.

Proof. If T is a Hall-perfect tree, we can give a KO-reduction scheme with one or two rounds. One round suffices if and only if M is a perfect matching; this is clear. Suppose M is a matching satisfying (2). By Hall's Theorem on matchings in bipartite graphs, this implies we can assign $p = |U|$ distinct U-triplets to the vertices of U, one v_i-triplet for each $v_i \in U = \{v_1, \ldots, v_p\}$. We can use the following KO-reduction scheme for each v_i and its associated v_i-triplet given by the vertices u_i, w_i, x_i, y_i of the P_3, where u_i is a neighbor of v_i. In the first round, for each i, v_i fires at one of the other neighbors (v_i is not an end vertex), u_i fires at w_i, and w_i fires at x_i, and the other matching edges of M are used to eliminate all (other) saturated vertices (except for all u_i). In the second round the edges $v_i u_i$ are used to eliminate all remaining vertices.

For the converse, suppose T can be eliminated in at most two rounds. If T needs only one round, we are done since this implies T has a perfect matching. Now suppose T needs exactly two rounds. Then in the second round the edges of a matching N between the remaining vertices after the first round are used to mutually eliminate their end vertices. Let us consider the edges $u_i v_i$ of this matching N. Each u_i has fired at a vertex x_i of T in the first round; at x_i starts a path $Q_i = x_{i_1} \ldots x_{i_{t_i}}$ of length at least one with the property that x_{i_s} fires at $x_{i_{s+1}}$ in the first round ($s = 1, \ldots, t_i - 1$), and $x_{i_{t_i}}$ fires at $x_{i_{t_i - 1}}$. Similar paths

R_i exist for the vertices v_i. Clearly, Q_i and R_i do not intersect, but for each edge u_iv_i of N one of the paths Q_i and R_i will have its tail in common with at least one of the paths associated with the other edges of N if $|N| \geq 2$ (since T is connected). Suppose that the two rounds in the KO-reduction scheme are chosen in such a way that $|N|$ is as small as possible. If there is only one edge in N, then $T = P_n$ with $n \geq 7$ and n odd, and it is easy to find a matching M showing that T is Hall-perfect. Suppose that $|N| \geq 2$. Consider an edge $u_iv_i \in N$ with the property that one of the associated paths Q_i and R_i, say Q_i, does not intersect with another path. Such an edge exists since T is a tree. Then R_i intersects with another path in a vertex w. We may assume that u_iv_i and Q_i are chosen such that $v_iw \in E(T)$. From the choice of N we also get that Q_i has even length ≥ 2. Then we can define a matching M_i saturating all vertices (including the leaf) of $V(Q_i) \cup \{u_i\}$, and a v_i-triplet on the path induced by $V(Q_i) \cup \{u_i\}$ for the unsaturated vertex v_i. Removing the vertices of $V(Q_i) \cup \{u_i, v_i\}$, and repeating this procedure as long as there are intersecting paths, and completing the procedure by a suitable choice for the remaining path, we obtain a matching M which shows that T is Hall-perfect. □

6 A Dynamic Program for Trees

In this section, we describe a polynomial time algorithm for computing the parallel knock-out number of a tree T. By Lemma 3 PKO(T) is either infinite, or it is bounded from above by $\lceil \log_3 n \rceil$, where n denotes the number of vertices in T. Without loss of generality we assume that $n \geq 3$.

We root the tree T in an arbitrary vertex called ROOT. We denote by $T(v)$ the maximal subtree of T that is rooted at vertex v. If $v \neq$ ROOT, there is some edge e_v that connects v to its father f_v. We are interested in the behavior of KO-reduction schemes inside of the subtrees $T(v)$: For $v \neq$ ROOT, the only interaction between $T(v)$ and $T - T(v)$ occurs along the edge e_v, and there is at most one round during which this edge e_v can be used. If edge e_v is used, then it is either fired upwards (the child v fires at the father f_v), or downwards (the father fires at the child), or both ways (father and child simultaneously fire at each other).

For every edge $v \neq$ ROOT and for every $r = 1, \ldots, \lceil \log_3 n \rceil$, we define three Boolean predicates UP$[v;r]$, DOWN$[v;r]$, and BOTH$[v;r]$: The predicate UP$[v;r]$ (respectively DOWN$[v;r]$, respectively BOTH$[v;r]$) is true, if there exists a KO-reduction scheme for $T(v)$, in which in round r the edge e_v is fired upwards (respectively downwards, respectively both ways). Moreover, for every vertex $v \neq$ ROOT (including the root), we introduce a Boolean predicate NONE$[v]$ which is true, if there exists a KO-reduction scheme for $T(v)$ which does not interact with vertices outside of $T(v)$; for $v \neq$ ROOT this means that the edge e_v is not used at all.

We compute the values of all these predicates by working upwards through the tree, starting in the leaves and ending in the root. For every leaf v_L, we have BOTH$[v_L;1]$ = DOWN$[v_L;1]$ = true and UP$[v_L;1]$ = NONE$[v_L]$ =

false. Moreover, for all $r \geq 2$ the three predicates $\mathrm{UP}[v_L; r]$, $\mathrm{DOWN}[v_L; r]$, and $\mathrm{BOTH}[v_L; r]$ are false. For non-leaf vertices v, the computation of the predicates is described in the following four lemmas.

Lemma 7. *For every non-leaf $v \in V(T)$ and for every $r = 1, \ldots, \lceil \log_3 n \rceil$, the value of $\mathrm{DOWN}[v; r]$ can be determined in polynomial time.*

Proof. Let v be a non-leaf vertex with children $v_1, \ldots, v_d$ and father f_v. What does it mean that $\mathrm{DOWN}[v; r]$ is true? Since the father f_v fires in round r along the edge e_v, vertex v is eliminated in round r. In the first r rounds, vertex v must have fired at r of its children. In the first $r-1$ rounds, none of the children has fired at vertex v. In round r, some of the surviving children of v may fire at v. In later rounds, none of the children can fire at v.

We model this situation via a bipartite auxiliary graph: The left vertex class in this bipartite graph has d vertices that correspond to the children $v_1, \ldots, v_d$. The right vertex class of the bipartite graph has d vertices that correspond to the possible firings along the d edges between vertex v and its children.

- For $k = 1, \ldots, r-1$ there is one edge that is used downwards during round k. We label a corresponding vertex in the bipartite graph by (DOWN, k).
- There is one edge along which v fires in round r. We label a corresponding vertex in the bipartite graph by the two labels (DOWN, r) and (BOTH, r).
- The remaining $d - r$ edges may be fired upwards in round r, or they are not being used at all. We label $d - r$ corresponding vertices in the bipartite graph by the two labels (UP, r) and (NONE).

The edges in the bipartite graph are defined as follows:

- If a vertex x in the right class has one label (DOWN, k) (respectively (UP, k), respectively (BOTH, k)), and if $\mathrm{DOWN}[v_i; k] = \mathrm{true}$ (respectively if $\mathrm{UP}[v_i; k] = \mathrm{true}$, respectively if $\mathrm{BOTH}[v_i; k] = \mathrm{true}$), then the bipartite graph has an edge between x and the vertex corresponding to v_i in the left class.
- Analogously, if a vertex x in the right class has a label (NONE) and if $\mathrm{NONE}[v_i; k] = \mathrm{true}$, then the bipartite graph has an edge between x and the vertex corresponding to v_i in the left class.

There are no other edges in the auxiliary graph. It can be seen that $\mathrm{DOWN}[v; r]$ is true if and only if the auxiliary graph contains a perfect matching. The existence of a perfect matching can be decided in polynomial time by standard methods. □

Lemma 8. *For every non-leaf $v \in V(T)$ and for every $r = 1, \ldots, \lceil \log_3 n \rceil$, the value of $\mathrm{UP}[v; r]$ can be determined in polynomial time.*

Proof. Let v be a non-leaf vertex with children $v_1, \ldots, v_d$ and father f_v. If $\mathrm{UP}[v; r]$ is true, then vertex v fires in round r upwards along the edge e_v. Therefore, vertex v must stay alive until it is eliminated in some round $s \geq r$. In the rounds $1, 2, \ldots, r-1$ and $r+1, \ldots, s$, vertex v must have fired at its children.

In the first $s-1$ rounds, none of the children has fired at vertex v. In round s, some of the surviving children of v may fire at v.

Hence, if we are given the value of s, then we can model this situation as a bipartite matching problem pretty much the same way as we did in the proof of Lemma 7. To find the value of $\mathrm{UP}[v;r]$, we simply test all possible values for $s = r, r+1, \ldots, \lceil \log_3 n \rceil$. $\mathrm{UP}[v;r]$ is true, if and only if at least one of these bipartite auxiliary graphs has a perfect matching. □

Lemma 9. *For every non-leaf $v \in V(T)$ and for every $r = 1, \ldots, \lceil \log_3 n \rceil$, the value of $\mathrm{BOTH}[v;r]$ can be determined in polynomial time.*

Proof. Once again, let v be a non-leaf vertex with children $v_1, \ldots, v_d$ and father f_v. If $\mathrm{UP}[v;r]$ is true, then vertex v and its father f_v eliminate each other in round r. In the rounds $1, 2, \ldots, r-1$ vertex v must have fired at its children, whereas none of the children has fired back at v. In round r, vertex v does not fire at its children, whereas some of the surviving children of v may fire at v. This problem can be modelled and solved as a bipartite matching problem too. □

Lemma 10. *For every non-leaf $v \in V(T)$, the value of $\mathrm{NONE}[v]$ can be determined in polynomial time.*

Proof. If $\mathrm{NONE}[v]$ is true, then vertex v is killed by one of its children in some round s with $1 \leq s \leq \lceil \log_3 n \rceil$. In the first $s-1$ rounds, vertex v must have fired at its children, whereas none of the children has fired back at v. In round s, vertex v fires at a child, and some of the surviving children of v may fire at v. We test all possible values for s, and solve the corresponding bipartite matching problems. □

If NONE[ROOT] is true in the end, then T is KO-reducible. To find the exact value of $\mathrm{PKO}(T)$, we remember the smallest number s in the proof of Lemma 10 for which a perfect matching exists. A perfect matching in a bipartite graph with α vertices and β edges can be found in $O(\beta\sqrt{\alpha})$ time. Our algorithm faces matching problems with $O(n)$ vertices and $O(n^2)$ edges, and altogether there are $O(n \log^2 n)$ matching problems to be solved. This yields the following theorem.

Theorem 3. *The parallel knock-out number of an n-vertex tree T can be computed in $O(n^{3.5} \log^2 n)$ time.* □

7 Claw-Free Graphs

We now turn to *claw-free* graphs, i.e. graphs that contain no $K_{1,3}$ as an induced subgraph. This is a well-studied class of graphs, especially with respect to algorithmical and structural properties. We refer to [1] for an excellent survey paper on claw-free graphs.

Since claw-free graphs admit perfect matchings and 2-factors under rather mild conditions, it is natural to consider conditions that guarantee a low parallel knock-out number in a claw-free graph. Here we prove a result involving the minimum degree $\delta(G)$ of the vertices of G.

Theorem 4. *Let G be a claw-free graph with $\delta(G) \geq 2$. Then* $\mathrm{PKO}(G) \leq 2$.

In fact, we will prove the following slightly stronger result from which the previous result is an easy consequence. For convenience we define the notion of a *2-KO-factor* of a graph G as a spanning subgraph of G consisting of a number of mutually disjoint copies of cycles, paths on 2 vertices, paths on 4 vertices, and paths on at least 6 vertices. Note that all the components of a 2-KO-factor have parallel knock-out number at most 2.

Theorem 5. *Every claw-free graph G with $\delta(G) \geq 2$ has a 2-KO-factor.*

Proof. The proof is by contradiction. Let G be a claw-free graph with $\delta(G) \geq 2$ and suppose G has no 2-KO-factor. Let $S \subseteq V(G)$ denote a smallest set such that the graph $G - S$ contains a 2-KO-factor F, and assume that subject to this S and F are chosen in such a way that the number of vertices of F contained in cycles of F is maximum, and subject to this the remaining vertices are on as few paths of F as possible. Clearly S is an independent set. Let $v \in S$. We first prove four claims in order to obtain restrictions on the neighborhood of v in F.

Claim. v is not adjacent to a vertex on a cycle of F.

Proof. If v is adjacent to a vertex u on a cycle C of F, then by the choice of S and F, vertex v is not adjacent to the two neighbors x and y of u on C. Since G is claw-free this implies that $xy \in E(G)$. We obtain a contradiction to the choice of S and F by replacing xuy in F by xy and adding the path corresponding to the edge vu. □

Claim. v is not adjacent to a vertex on a path component with more than 6 vertices of F.

Proof. If v is adjacent to a vertex u on a path component P_k of F with $k \geq 7$, then by the choice of S and F, vertex u is not an end vertex of P, and v is not adjacent to the two neighbors x and y of u on P. Since G is claw-free this implies that $xy \in E(G)$. We obtain a contradiction to the choice of S and F by replacing xuy in F by xy and adding the path corresponding to the edge vu. □

Claim. v is not adjacent to two vertices on one path component $P \neq P_4$ of F.

Proof. If v is adjacent to two vertices x and y on one path component $P \neq P_4$ of F, then by the choice of S and F, $xy \notin E(P)$. Hence $P \neq P_2$. Let $P = P_k$ with $k \geq 6$. By Claim 7, $k = 6$ must hold. Clearly none of x and y is an end vertex of P, and because of the claw-freeness there are edges in G between the two neighbors of x on P and between the two neighbors of y on P. If x or y is next to an end vertex of P, we easily find a contradiction to the choice of S and F. If both x and y are not next to an end vertex of P, we obtain a C_3 and two matching edges containing all vertices of $V(P) \cup \{v\}$, also contradicting the choice of S and F. □

Claim. v is not adjacent to two vertices on one path component $P = P_4$ of F.

Proof. If v is adjacent to two vertices x and y on one path component $P = P_4$ of F, then by the choice of S and F, x and y are neighbors on P; otherwise we obtain a C_5 or a C_4 containing at least the same number of vertices as P. None of x and y is an end vertex of P; otherwise we obtain a C_3 and a matching edge containing all vertices of $V(P) \cup \{v\}$. Next we consider an end vertex v' of P. Clearly the choice of S and F implies that v' is neither adjacent to a second vertex of P, nor to a vertex of a cycle of F or a P_2 or P_4 of F. By the same arguments as above, v' is not adjacent to a vertex of a P_k of F with $k \geq 7$. Hence v' is adjacent to a vertex w of a P_6 of F, and by similar arguments as before, w is neither an end vertex of the P_6 nor next to an end vertex of the P_6. We can obtain a P_9 and a P_2 containing all the vertices of $V(P) \cup \{v\}$ and of the P_6. □

Now let us complete the proof of Theorem 5. Since S is an independent set and $\delta(G) \geq 2$, Claims 7 to 7 imply that v is adjacent to (at least) two vertices on different path components on 2, 4 or 6 vertices of F, and to no other vertices on the same path component. In the case where such a neighbor u is not an end vertex, claw-freeness implies the existence of an edge between the two neighbors of u on the path component. Below we indicate how to obtain a contradiction with the choice of S and F in all possible cases, without giving all the details. Here $v, P_i, P_j \rightarrow P_{i+j+1}$ and $v, P_i, P_j \rightarrow v', P_{i+j}$ indicate that we can obtain a longer path containing all vertices or all vertices but one, respectively.

$v, P_2, P_2 \rightarrow v', P_4$

$v, P_2, P_4 \rightarrow P_7$

$v, P_2, P_6 \rightarrow P_9$ or $v, P_2, P_6 \rightarrow v', P_8$

$v, P_4, P_4 \rightarrow P_9$

$v, P_4, P_6 \rightarrow P_{11}$ or $v, P_4, P_6 \rightarrow v', P_{10}$

$v, P_6, P_6 \rightarrow P_{13}$ or $v, P_6, P_6 \rightarrow v', P_{12}$ or we can obtain a P_9 and two matching edges containing all vertices. This completes the proof of Theorem 5. □

It is easy to give examples showing that we cannot omit the degree condition in the above results. One could try to replace the minimum degree condition by the weaker condition that every vertex with degree 1 has a neighbor with a high degree, but this does not work either. Consider e.g. the claw-free graph G obtained from a complete graph K_k $(k \geq 2)$ by adding $k-1$ new vertices and matching edges saturating all new vertices. One easily checks that $\text{PKO}(G) = \infty$.

References

1. R. Faudree, E. Flandrin and Z. Ryjáček (1997). Claw-free graphs—a survey. *Discrete Mathematics 164,* 87–147.
2. D.E. Lampert and P.J. Slater (1998). Parallel knockouts in the complete graph. *American Mathematical Monthly 105,* 556–558.
3. D.E. Lampert and P.J. Slater (1997). The expected number of surviving vertices under parallel knockouts. Unpublished manuscript.

Online Algorithms for Disk Graphs*

Ioannis Caragiannis[1], Aleksei Fishkin[2], and Christos Kaklamanis[1], and Evi Papaioannou[1]

[1] Research Academic Computer Technology Institute and Department of Computer Engineering and Informatics University of Patras, 26500 Rio, Greece
[2] Institut für Informatik und Praktische Mathematik Universität zu Kiel, Christian-Albrechts Platz 4, 24098, Germany

Abstract. We study the on-line versions of two fundamental graph problems, maximum independent set and minimum coloring, for the case of *disk graphs* which are graphs resulting from intersections of disks on the plane. In particular, we investigate whether randomization can be used to break known lower bounds for deterministic on-line independent set algorithms and present new upper and lower bounds; we also present an improved upper bound for on-line coloring.

1 Introduction

We study two fundamental graph problems, maximum independent set and minimum coloring. Given a graph G, the maximum independent set problem is to find an independent set (i.e., a set of nodes without edges between them) of maximum size, while the minimum coloring problem is to find an assignment of colors (i.e, positive integers) to the nodes of the graph so that no two nodes connected by an edge are assigned the same color and the number of colors used is minimized. We consider graphs modelling intersections of disks in the plane.

The intersection graph of a set of disks in the Euclidean plane is the graph having a node for each disk and an edge between two nodes if and only if the corresponding disks overlap. Each disk is defined by its radius and the coordinates of its center. Two disks overlap if the distance between their centers is strictly smaller than the sum of their radii. A graph G is called a *disk graph* if there exists a set of disks in the Euclidean plane whose intersection graph is G. The set of disks is called the disk representation of G. A disk graph is called unit disk graph if all disks in its disk representation have the same radius. A disk graph is σ-bounded if the ratio between the maximum and the minimum radius among all the disks in its disk representation is at most σ.

In disk graphs, maximum independent set and minimum coloring are important since they can model resource allocation problems in radio communication networks [11]. Consider a set of transmitters located in fixed positions within a

* This work was partially supported by the European Union under IST FET Project CRESCCO, IST FET Project DELIS and RTN ARACNE.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 215–226, 2004.
© Springer-Verlag Berlin Heidelberg 2004

geographical region. Each transmitter uses a specific frequency to transmit its messages. Two transmitters can successfully (i.e., without signal interference) transmit messages simultaneously either if they use different frequencies or if they use the same frequency and their ranges do not overlap. Given a set of transmitters in a radio network, in order to guarantee successful transmissions simultaneously, important engineering problems that have to be solved are the frequency assignment problem where the objective is to minimize the number of frequencies used all over the network, and the call admission problem where the objective is to find a maximum-sized set of transmitters which can use the same frequency. Assuming that all transmitters have circular range, the graph reflecting possible interference between pairs of transmitters is a disk graph. The frequency assignment and call admission problems are equivalent to minimum coloring and maximum independent set problems, respectively.

An instance of the maximum independent set or the minimum coloring problem may or may not include the disk representation (i.e., disk center coordinates and/or radii) of the disk graph as part of the input. Clearly, the latter case is more difficult. Information about the disk representation of a disk graph is not easy to extract. Actually, determining whether a graph is a disk graph is an NP-complete problem [13].

The maximum independent set in disk graphs has been proved to be NP-hard even for unit disk graphs and even if the disk representation is given as part of the input [3]. A naive independent set algorithm is the algorithm First-Fit: starting from an empty set, it incrementally constructs an independent set by examining the nodes of the graph in an arbitrary order and including a node in the independent set only if none of its neighbors has been previously included. When applied to unit disk graphs, First-Fit has approximation ratio at most 5 and does not use the disk representation [18] (also implicit in [14]). In [18], a 3-approximation algorithm is obtained by computing a specific ordering of the nodes of a unit disk graph and running First-Fit according to this ordering. A similar idea leads to a 5-approximation algorithm in general disk graphs [18]. Furthermore, as it has been observed in [5], a $(2.5+\epsilon)$-approximation algorithm for unit disk graphs follows by a more general result presented in [12]. None of the algorithms above use the disk representation. Polynomial-time approximation schemes have been presented for both unit disk graphs [15,19] and general disk graphs [6,2] when the disk representation is given.

The minimum coloring problem has also been proved to be NP-hard in [3,9] even for unit disk graphs. Again, First-Fit algorithm can be used. It examines the nodes of the graph in an arbitrary order and assigns to each node the smallest color not assigned to its already examined neighbors. Algorithm First-Fit computes 5-approximate solutions in unit disk graphs [8,18]. By processing the nodes of the graph in a specific order, First-Fit computes 3-approximate solutions in unit disk graphs [9,18,20]. In general disk graphs, a smallest-degree-last version of First-Fit achieves an approximation ratio of 5 [8,17,18].

In the on-line versions of the problems, the disk graph is not given in advance but is revealed in steps. In each step, a node of the graph appears together with

its edges incident to nodes appeared in previous steps (and possibly, together with the center coordinates and/or the radius of the corresponding disk). When a node appears, an on-line independent set algorithm decides either to accept the node by including it in the independent set or to reject it, while an on-line coloring algorithm decides which color to assign to the node. In each case, the decisions of the algorithm cannot change in the future. The performance of an on-line algorithm is measured in terms of its competitive ratio (or competitiveness). For on-line independent set algorithms, the competitive ratio is defined as the maximum over all possible sequences of disks of the ratio of the size of the maximum independent set over the size of the independent set computed by the algorithm. For on-line coloring algorithms, the competitive ratio is defined as the maximum over all possible sequences of disks of the ratio of the number of colors used by the algorithm over the minimum number of colors sufficient for coloring the graph.

First-Fit is essentially an on-line algorithm. For the independent set problem, it has competitive ratio 5 in unit disk graphs [14,18] and $O(\min\{n, \sigma^2\})$ in σ-bounded disk graphs with n nodes [5]. As it is observed in [5], First-Fit is optimal within the class of deterministic on-line algorithms.

The First-Fit coloring algorithm has been widely studied in a more general context and has been proved to be $\Theta(\log n)$-competitive in inductive graphs with n nodes [16,10]. The lower bound holds also for trees (which are disk graphs) so the $\Theta(\log n)$ bound holds for general disk graphs. In unit disk graphs, First-Fit is at most 5-competitive [8,18] while for σ-bounded disk graphs with n nodes, it is at most $O(\min\{\log n, \sigma^2\})$-competitive [4]. For unit disk graphs, a lower bound of 2 on the competitiveness of any deterministic on-line coloring algorithm is presented in [7]. The best known lower bound on the competitiveness of deterministic coloring algorithms in σ-bounded disk graphs is $\Omega(\min\{\log n, \log\log \sigma\})$ [4]. Better on-line coloring algorithms exist for σ-bounded disk graphs in the case where the disk representation is given. Most of them use First-Fit as a subroutine. The best competitiveness upper bound in this case is $O(\min\{\log n, \log \sigma\})$ [4].

In this paper, we study the on-line version of both problems. For the independent set problem, we investigate whether randomization helps in improving the competitiveness of on-line algorithms. For randomized on-line independent set algorithms, the competitive ratio is defined as the maximum over all possible sequences of disks of the ratio of the size of the maximum independent set over the expected size of the independent set computed by the algorithm. We assume that the sequences of disks are selected by oblivious adversaries, i.e., adversaries that have no knowledge of the random choices of the algorithms (but may know the probability distribution used by the algorithm for making random choices). This is a typical assumption usually made in the study of randomized on-line algorithms [1]. Somewhat surprisingly, we show that, in general, randomization does not help against oblivious adversaries even if the disk representation is given, i.e., we construct sequences of disks for which no (possibly randomized) on-line algorithm can be better than $\Omega(n)$-competitive. In the case that the disk representation is not given, we prove a lower bound of $\Omega(\min\{n, \sigma^2\})$ on the

Table 1. Summary of results for the on-line independent set problem. (*) indicates results in this paper.

		Deterministic alg.	Randomized algorithms	
	Disk repr.	Lower/upper bound	Lower bound	Upper bound
σ-DG	+	$\Theta(\min\{n, \sigma^2\})$	* $\Omega(\min\{n, \log\sigma\})$	* $O(\min\{n, \log\sigma\})$ * $O\left(\min\{n, \frac{1}{\epsilon}\log\sigma\log^{1+\epsilon}\log\sigma\}\right)$
σ-DG	−	$\Theta(\min\{n, \sigma^2\})$	* $\Omega(\min\{n, \sigma^2\})$	$O(\min\{n, \sigma^2\})$
UDG	+	5	* 2.5	* $\frac{8\sqrt{3}}{\pi} \approx 4.41$
UDG	−	5	* 3	5

competitiveness of on-line algorithms on σ-bounded disk graphs with n nodes meaning that algorithm First-Fit is optimal within a small constant factor. For the case of σ-bounded disk graphs with given representation, we present randomized algorithms with competitive ratio almost logarithmic in σ and show that they are optimal. For unit disk graphs, we present a randomized algorithm with competitive ratio 4.41. We also show lower bounds of 2.5 and 3 for randomized algorithms in unit disk graphs. Our results for the on-line independent set problem together with the previously known results on deterministic on-line algorithms are summarized in Table 1. For the coloring problem, we show how to achieve the best known upper bound of $O(\min\{\log n, \log\sigma\})$ for σ-bounded sequences of n disks even if the disk representation is not given.

The rest of the paper is structured as follows. Section 2 is devoted to the on-line independent set problem in σ-bounded disk graphs. Our results for unit disk graphs are presented in Section 3 while our coloring algorithm is presented in Section 4. We conclude with extensions and open problems in Section 5. Due to lack of space, most of the proofs have been omitted. They will appear in the final version of the paper.

2 Independent Sets in σ-Bounded Disk Graphs

2.1 Upper Bounds

In this section we present the randomized on-line algorithm Classify for computing independent sets in disk graphs. It has a competitive ratio $O(\min\{n, \log\sigma\})$ against oblivious adversaries on σ-bounded disk graphs with n nodes. The algorithm uses the value of σ which is supposed to be known in advance and makes its random choices based on the disk representation. Despite these limitations, this is the first algorithm achieving a competitive ratio logarithmic in σ and (as we will prove in Section 2.2) is optimal among the on-line algorithms that use the disk representation.

Algorithm Classify works as follows. When the first disk is presented, the algorithm flips a coin. On heads, it accepts the disk and executes algorithm First-Fit for disks having radii in the interval $[R, 2R)$, where R is the radius of

the first disk presented, ignoring (i.e., rejecting) all other disks. On tails, the algorithm selects equiprobably a number i from the set $\{-\lceil \log \sigma \rceil, -\lceil \log \sigma \rceil + 1, ..., -1, 1, ..., \lceil \log \sigma \rceil\}$ and executes algorithm First-Fit for disks of radius in the interval $[R2^i, R2^{i+1})$, ignoring (i.e., rejecting) all other disks.

We prove the following theorem.

Theorem 1. *Algorithm* Classify *is* $O(\min\{n, \log \sigma\})$*-competitive against oblivious adversaries on* σ*-bounded disk graphs with* n *nodes.*

Proof. Since the first disk is accepted with probability 1/2, the algorithm has competitive ratio $O(n)$. In what follows, we show that the algorithm is $O(\log \sigma)$-competitive as well. Denote by OPT the optimal independent set of the sequence. For $i = -\lceil \log \sigma \rceil, -\lceil \log \sigma \rceil - 1, ..., \lceil \log \sigma \rceil$, denote by S_i the set of disks with radius in the interval $[R2^i, R2^{i+1})$ and by OPT_i the maximum independent set among the disks belonging to set S_i. Clearly, $|OPT_i| \geq |OPT \cap S_i|$ since $OPT \cap S_i$ is an independent set for S_i. Assume that the algorithm selects set S_i and executes algorithm First-Fit on the disks of that set. Observe that disks in S_i form a 2-bounded disk graph. In such graphs, the following lemma gives a guarantee on the performance of algorithm First-Fit for computing independent sets.

Lemma 1. *Algorithm* First-Fit *is at most* 15*-competitive on* 2*-bounded disk graphs.*

Using the lemma, we obtain that the algorithm accepts at least

$$B_i \geq \frac{1}{15}|OPT_i| \geq \frac{1}{15}|OPT \cap S_i|$$

disks of S_i. Now, the expected size of the independent set computed by algorithm Classify is

$$\begin{aligned} E[B] &= \sum_{i=-\lceil \log \sigma \rceil}^{\lceil \log \sigma \rceil} (\Pr[S_i \text{ is selected}] \cdot B_i) \\ &\geq \frac{1}{15} \sum_{i=-\lceil \log \sigma \rceil}^{\lceil \log \sigma \rceil} (\Pr[S_i \text{ is selected}] \cdot |OPT \cap S_i|) \\ &\geq \frac{1}{15} \min_i \{\Pr[S_i \text{ is selected}]\} \cdot |OPT| \\ &\geq \frac{1}{60\lceil \log \sigma \rceil} \cdot |OPT|. \end{aligned}$$

Hence, the competitive ratio of the algorithm is $O(\log \sigma)$. □

We now present algorithm Guess which achieves a slightly weaker competitive ratio but does not need to known neither n nor σ in advance. Consider a sequence of n disks and let R be the radius of the first disk of the sequence. Then, for any $i = 0, 1, ..., 2\lceil \log \sigma \rceil - 1$, define the set of disks S_i with radii at least $R/2^{\lceil \log \sigma \rceil - i}$ and smaller than $R/2^{\lceil \log \sigma \rceil - i - 1}$. When the first disk of each set is presented,

the algorithm probabilistically determines whether it will consider disks from that specific set and ignore all disks from all other sets. Sets are divided into epochs. Epoch 0 consists of the set containing the first disk of the sequence. For $j = 1, 2, ...$, epoch j consists of the 2^j sets presented after the sets of epoch $j-1$.

When the first disk appears, the algorithm tosses a coin with $\Pr[HEADS] = 1/2$. On heads, the algorithm decides to run First-Fit on the disks belonging to the first set of epoch 0 and ignores (i.e., rejects) all other disks; on tails, it decides to reject all disks belonging to the first set of epoch 0. When the first disk of the first set of the j-th epoch (for $j > 0$) is presented, if the algorithm has rejected all sets in epochs $0, ..., j-1$, it tosses a coin with $\Pr[HEADS] = \frac{\epsilon}{i+2}$. On HEADS, it equiprobably selects one of the 2^j sets of the epoch and decides to run First-Fit on the disks belonging to that set and ignores (i.e., rejects) disks from all other sets; on tails, it rejects all disks from all sets of the epoch.

Theorem 2. *Algorithm* Guess *is at most* $O\left(\min\left\{n, \frac{1}{\epsilon}\log\sigma\log^{1+\epsilon}\log\sigma\right\}\right)$*-competitive, for any* $\epsilon \in (0, 1]$*, against oblivious adversaries in* σ*-bounded disk graphs with* n *nodes.*

2.2 Lower Bounds

The lower bounds presented in this section show that, in general, randomization does not help, i.e., there are sequences of n disks for which any on-line algorithm is $\Omega(n)$-competitive even if the disk representation is given. For σ-bounded disk graphs, the next lower bound states that when the disk representation is not given, on-line algorithms with competitive ratio logarithmic in σ do not exist.

Theorem 3. *Any randomized on-line algorithm for computing independent sets in* σ*-bounded disk graphs with* n *nodes is* $\Omega(\min\{n, \sigma^2\})$*-competitive against oblivious adversaries, if the disk representation is not given.*

Proof. Let κ be a positive integer. We will construct an adversary which generates a graph G_κ with an independent set of size $\kappa+1$ such that the expectation of the size of the independent set of G_κ that any randomized on-line algorithm can find is at most 2.

The graph G_κ generated by the adversary is defined as follows. The nodes of G_κ are partitioned into κ levels $0, 1, ..., \kappa-1$. Each level i has two nodes: a left node v_l^i and a right node v_r^i. The two nodes of a level are non-adjacent. First, the adversary generates the two nodes of level 0. For $i = 1, ..., \kappa-1$, the nodes of level i are generated after the nodes of level $i-1$. The adversary tosses a coin in order to connect the nodes of level i with nodes of smaller levels. On heads, it connects both nodes of level i to node v_l^{i-1} and to all nodes of levels $i-2, i-3, ..., 0$ to which node v_l^{i-1} is connected; on tails, it connects both nodes of level i to node v_r^{i-1} and to all nodes of levels $i-2, ..., 0$ to which node v_r^{i-1} is connected.

Consider the set of nodes consisting of the two nodes of level $\kappa-1$ and, for $i = 0, ..., \kappa-2$, of the node of level i which is not connected to nodes of higher

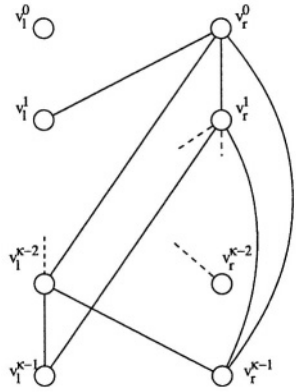


Fig. 1. An example of graph G_κ.

levels. This is an independent set of G_κ. Hence, the optimal independent set of G_κ has size at least $\kappa + 1$.

In what follows we will show that the size of the independent set of G_κ any (possibly randomized) on-line algorithm can compute is at most 2. Consider the application of an algorithm A on t sequences of disks produced by the adversary. Denote by l_i the number of executions in which the algorithm accepts the left node of level i, by r_i the number of executions in which the algorithm accepts the right node of level i, and by b_i the number of executions in which the algorithm accepts both nodes of level i.

For $i = 0, 1, ..., \kappa - 1$, let X_i be the random variable denoting the number of executions in which the nodes presented at level i are unconstrained by nodes of smaller levels (i.e., they are not connected to nodes of smaller levels that have been accepted by the algorithm). Then, nodes of level $i+1$ are constrained only if: (i) the left node of level i is rejected and the nodes of level $i+1$ are connected to the left node of level i or (ii) the right node of level i is rejected and the nodes of level $i+1$ are connected to the right node of level i. Hence,

$$E[X_{i+1}|X_i] \leq X_i - \frac{r_i + l_i}{2} - b_i$$

and

$$E[X_{\kappa-1}] \leq t - \sum_{i=0}^{\kappa-2} \frac{r_i + l_i + 2b_i}{2} \Rightarrow$$

$$\sum_{i=0}^{\kappa-2} (r_i + l_i + 2b_i) \leq 2t - 2E[X_{\kappa-1}] \tag{1}$$

Since the number of executions in which the algorithm accepts at least one node from level $\kappa - 1$ is at most the number of executions in which the nodes of level $\kappa - 1$ are unconstrained, it is $r_{\kappa-1} + l_{\kappa-1} + b_{\kappa-1} \leq E[X_{\kappa-1}]$. Now, using (1), we

obtain that the expectation of the size of the independent set $B(G_\kappa)$ computed by the algorithm is

$$\begin{aligned} E[|B(G_\kappa)|] &= \frac{1}{t}\sum_{i=0}^{\kappa-1}(r_i + l_i + 2b_i) \\ &= \frac{1}{t}\sum_{i=0}^{\kappa-2}(r_i + l_i + 2b_i) + \frac{1}{t}(r_{\kappa-1} + l_{\kappa-1} + 2b_{\kappa-1}) \\ &\leq 2 - \frac{1}{t}(2E[X_{\kappa-1}] - (r_{\kappa-1} + l_{\kappa-1} + 2b_{\kappa-1})) \\ &\leq 2. \end{aligned}$$

We conclude that the competitive ratio of the algorithm is at least $\frac{\kappa+1}{2}$.

It remains to show that graph G_κ for $\kappa = \Omega\left(\min\{n, \sigma^2\}\right)$ is a σ-bounded disk graph. This is stated in the following lemma.

Lemma 2. *For any $\sigma \geq 2$, graph G_{4d^2} for $d = \lfloor \frac{\sigma+2}{4} \rfloor$ is a σ-bounded disk graph.*

The proof of the lemma will appear in the final version of the paper. This completes the proof of the theorem. □

The next lower bound states that algorithm Classify is optimal.

Theorem 4. *Any randomized on-line algorithm for computing independent sets in σ-bounded disk graphs with n nodes is $\Omega(\min\{n, \log\sigma\})$-competitive against oblivious adversaries.*

3 Independent Sets in Unit Disk Graphs

In this section, we present new upper and lower bounds on the competitiveness of on-line randomized independent set algorithms for unit disk graphs.

We first present algorithm Filter, an on-line randomized algorithm for computing independent sets in unit disk graphs. We show that the algorithm is $\frac{8\sqrt{3}}{\pi} \approx 4.41$-competitive against oblivious adversaries.

At the beginning, algorithm Filter selects α and β uniformly at random from the intervals $[0, 4)$ and $[0, 2\sqrt{3})$, respectively. When a new disk centered at point (x, y) appears, the algorithm does the following: If there are integers κ, λ such that the point $(x + \alpha, y + \beta)$ has distance less than 1 from the point with coordinates $(4\kappa + 2(\lambda \bmod 2), 2\lambda\sqrt{3})$, then Filter executes algorithm First-Fit, else it ignores the disk.

Theorem 5. *Algorithm* Filter *is $\frac{8\sqrt{3}}{\pi}$-competitive against oblivious adversaries.*

Proof. Consider the application of algorithm Filter on a sequence $\mathcal{D}$ of disks of unit radius. Let $\mathcal{D}'$ denote the (random) subsequence of $\mathcal{D}$ consisting of the disks not ignored by the algorithm. We denote by $A(\mathcal{D})$ the maximum independent set of a sequence $\mathcal{D}$ and by $B(\mathcal{D})$ the set of disks accepted by the algorithm.

We first show that the probability that a disk is not ignored by the algorithm is $\frac{\pi}{8\sqrt{3}}$. Consider a disk D with center at point (x, y) and the rectangle defined by the diagonal points (x, y) and $(x + 4, y + 2\sqrt{3})$. Also, consider the unit disks containing the points at distance less than 1 from points with coordinates $(4\kappa + 2(\lambda \bmod 2), 2\lambda\sqrt{3})$ for integer κ and λ, and observe that the total area of the intersection of these disks with the rectangle equals the area of a disk with radius 1 (see Figure 2. Since point $(x + \alpha, y + \beta)$ is uniformly distributed within the rectangle, the probability that the disk D is not ignored by algorithm Filter is equal to the area of a disk of radius 1 over the area of the rectangle, i.e., $\frac{\pi}{8\sqrt{3}}$.

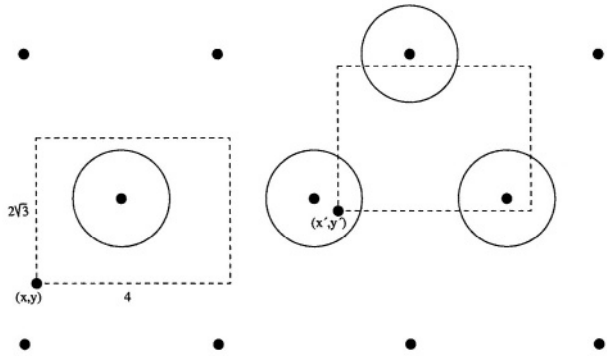


Fig. 2. Disk centers (x, y) and (x', y') and the rectangles where points $(x + \alpha, y + \beta)$ and $(x' + \alpha, y' + \beta)$ are uniformly distributed.

Now consider the maximum independent set $A(\mathcal{D})$ and let $A'(\mathcal{D})$ be the (random) subset of $A(\mathcal{D})$ consisting of the disks of $A(\mathcal{D})$ not ignored by algorithm Filter. Clearly, $A'(\mathcal{D})$ is an independent set for the set of disks $\mathcal{D}'$, thus, it is $|A(\mathcal{D}')| \geq |A'(\mathcal{D})|$. By linearity of expectation, we obtain that $|A'(\mathcal{D})| = \frac{\pi}{8\sqrt{3}}|A(\mathcal{D})|$ meaning that

$$E[|A(\mathcal{D}')|] \geq \frac{\pi}{8\sqrt{3}}|A(\mathcal{D})| \tag{2}$$

We now observe that each connected component of the intersection graph defined by the disks in $\mathcal{D}'$ is a clique. In particular, consider the two points O_1 with coordinates $(4\kappa_1 + 2(\lambda_1 \bmod 2), 2\lambda_1\sqrt{3})$ and O_2 with coordinates $(4\kappa_2 + 2(\lambda_2 \bmod 2), 2\lambda_2\sqrt{3})$ such that either $\kappa_1 \neq \kappa_2$ or $\lambda_1 \neq \lambda_2$ and three disks D_1, D_2, and D_3 centered at points C_1, C_2, and C_3 with coordinates (x_1, y_1), (x_2, y_2), and (x_3, y_3), respectively. Also, denote by C_1', C_2', and C_3' the points with coordinates $(x_1 + \alpha, y_1 + \beta)$, $(x_2 + \alpha, y_2 + \beta)$, and $(x_3 + \alpha, y_3 + \beta)$, respectively. Assume that points C_1 and C_3 have distance smaller than 1 from point O_1, and point C_2 has distance smaller than 1 from point O_2. We will show that disks D_1 and D_3 overlap while disks D_1 and D_2 are non-overlapping. Clearly, it is $|C_1C_3| = |C_1'C_3'|$ and by triangle inequality, we obtain that $|C_1C_3| \leq |C_1'O_1| + |O_1C_2'| < 2$.

Hence, disks D_1 and D_3 overlap. Now, it can be easily verified that if either $\kappa_1 \neq \kappa_2$ or $\lambda_1 \neq \lambda_2$, it is $|O_1O_2| \geq 4$. By the triangle inequality, we have that $|O_1C_1'| + |C_1'C_2'| + |C_2'O_2| \geq 4$. Clearly, $|C_1'C_2'| = |C_1C_2|$ and since $|O_1C_1'| < 1$ and $|C_2'O_2| < 1$, it is also $|C_1C_2| > 2$ meaning that disks D_1 and D_2 do not overlap.

Now, since each connected component of the intersection graph of $\mathcal{D}'$ is a clique, the maximum independent set in the neighborhood of a disk has size at most 1. So, any disk accepted by algorithm Filter may block at most one disk in $A(\mathcal{D}')$. Hence, for the subsequence $\mathcal{D}'$ of the disks not ignored by algorithm Filter, it is $B(\mathcal{D}') \geq |A(\mathcal{D}')|$ implying that $E[B(\mathcal{D}')] \geq E[|A(\mathcal{D}')|]$. Using (2), we obtain that the competitive ratio of algorithm Filter is

$$\frac{|A(\mathcal{D})|}{E[B(\mathcal{D}')]} \leq \frac{8\sqrt{3}}{\pi}.$$

□

By adapting the lower bound construction of Section 2.2 to the case of unit disk graphs, we obtain the following statement.

Theorem 6. *No on-line (randomized) algorithm for computing independent sets in unit disk graphs can be better than* 3*-competitive against oblivious adversaries if the disk representation is not given. Even if the disk representation is given, then no on-line (randomized) algorithm can be better than* 2.5*-competitive against oblivious adversaries.*

4 An Upper Bound for Online Coloring

In this section we present an on-line coloring algorithm for disk graphs which does not require the disk representation. It achieves competitive ratio $O(\min\{\log n, \log \sigma\})$ for coloring σ-bounded sequences of n disks matching the best known upper bound for the case where the disk representation is given. The algorithm is a combination of algorithm First-Fit and algorithm Layered which is presented in the following.

The algorithm Layered classifies the disks into layers and applies algorithm First-Fit to each layer separately, using a different set of colors in each layer. Layers are numbered with integers 1, 2, ... and a disk is classified into the smallest layer possible under the constraint that it cannot be classified into a layer if it overlaps with at least 16 mutually non-overlapping disks belonging to this layer. We can show the following.

Lemma 3. *For any σ-bounded sequence of disks, the number of layers constructed by algorithm* Layered *is at most* $1 + \log \sigma$.

Theorem 7. *The algorithm* Layered *is $O(\log \sigma)$-competitive when applied to σ-bounded sequences of disks.*

Proof. Consider the application of algorithm Layered on a σ-bounded sequence of disks. Let j be the layer where the maximum number of colors has been used. Let α be the highest color used in layer j and let D_j be the disk colored with color α. Then, this disk overlaps with at least $\alpha-1$ disks of layer j appeared prior to it. By the definition of the algorithm, the number of mutually non-overlapping disks of layer j overlapping with D_j is at most 15. This implies that the optimal algorithm should use at least $\alpha/15$ colors for coloring the disks of the sequence while, by Lemma 3, algorithm Layered uses at most $\alpha(\log\sigma+1)$ colors, hence, it is at most $15(\log\sigma+1)$-competitive. □

We now combine algorithms First-Fit and Layered using a technique proposed in [4,5] to obtain a better result. We use two separate sets of colors for algorithms First-Fit and Layered. When a new disk D_i is presented we run algorithm First-Fit on D_i together with those disks colored by First-Fit. Similarly, we execute Layered. Then, we compare the results of these two algorithms and color D_i with the algorithm that has used fewer colors up to that point (including the color used for disk D_i). The total number of colors used is at most the sum of the number of colors used by both methods. Note that at any time of the execution of the combined algorithm, the number of colors used by First-Fit and the number of colors used by Layered differ by at most one. Assume that $n<\sigma$. The number of colors used by First-Fit is at most $O(\log n)$ times the optimal number of colors. The number of colors used by Layered is at most one more than that of First-Fit. So, the total number of colors is at most $O(\log n)$ times the optimal number of colors. A similar argument holds in the case where $n\geq\sigma$. We obtain the following theorem.

Theorem 8. *There exists an* $O(\min\{\log n,\log\sigma\}$*-competitive algorithm for on-line coloring a* σ*-bounded disk graph with* n *nodes.*

5 Extensions and Open Problems

The results for the independent set extend to the more general problem where we are given $w\geq 1$ colors and the objective is to accept the maximum number of disks which can be properly colored with at most w colors (clearly, for $w=1$, this is the independent set problem). Algorithms Classify, Guess, and Filter can be easily modified to solve this problem with the same competitiveness bounds we proved for the independent set problem.

The most interesting open problem related to the independent set problem is perhaps to close the gap on the competitiveness of (randomized) on-line algorithms in unit disk graphs. It would be very interesting even to find an algorithm with competitive ratio smaller than 5 which does not require the disk representation. For the coloring problem, there is still a large gap (in terms of σ) between the competitiveness of algorithm Layered and the known lower bounds.

References

1. A. Borodin and R. El-Yaniv. Online computation and competitive analysis. *Cambridge University Press,* 1998.
2. T.M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms,* 46, pp. 178-189, 2003.
3. B.N. Clark, C.J. Colbourn, and D.S. Johnson. Unit disk graphs. *Discrete Mathematics,* 86, pp. 165-177, 1990.
4. T. Erlebach and J. Fiala. On-line coloring of geometric intersection graphs. *Computational Geometry: Theory and Applications,* 9(1-2), pp. 3-24, 2002.
5. T. Erlebach and J. Fiala. Independence and coloring problems on intersection graphs of disks. *Manuscript,* 2003.
6. T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proc. of the 12th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA '01),* pp. 671-679, 2001.
7. J. Fiala, A.V. Fishkin, and F.V. Fomin. Off-line and on-line distance constrained labeling of graphs. In *Proc. of the 9th Annual European Symposium on Algorithms (ESA '01),* LNCS 2161, pp. 464-475, 2001.
8. A. Gräf. Coloring and recognizing special graph classes. Musikinformatik und Medientechnik Bericht 20/95, Johannes Gutenberg, Universität Mainz, 1995.
9. A. Gräf, M. Stumpf, and G. Weissenfels. On coloring unit disk graphs. *Algorithmica,* 20(3), pp. 277-293, 1998.
10. A. Gyárfás and J. Lehel. On-line and first fit colorings of graphs. *Journal of Graph Theory,* 12(2), pp. 217-227, 1988.
11. D.K. Hale. Frequency assignment: theory and applications. In *Proc. of the IEEE,* 68(12), pp. 1497-1514, 1980.
12. M.M. Halldórsson. Approximating discrete collections via local improvements. In *Proc. of the 6th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA '95),* pp. 160-169, 1995.
13. P. Hliněný and J. Kratochvíl. Representing graphs by disks and balls. *Discrete Mathematics,* 229(1-3), pp. 101-124, 2001.
14. D. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics,* 6, pp. 243-254, 1983.
15. H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R.E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *Journal of Algorithms,* 26(2), pp. 238-274, 1998.
16. S. Irani. Coloring inductive graphs on-line. *Algorithmica,* 11, pp. 53-72, 1994.
17. E. Malesińska. Graph theoretical models for frequency assignment problems. *PhD Thesis,* Technical University of Berlin, 1997.
18. M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, and D.J. Rosenkrantz. Simple hueristics for unit disk graphs. *Networks,* 25, pp. 59-68, 1995.
19. T. Matsui. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In *Discrete and Computational Geometry,* LNCS 1763, Springer, pp. 194-200, 2000.
20. R. Peeters. On coloring j-unit sphere graphs. Technical report, Dept. of Economics, Tilburg University, 1991.

Protein Folding in the HP Model on Grid Lattices with Diagonals*
(Extended Abstract)

Hans-Joachim Böckenhauer and Dirk Bongartz

Lehrstuhl für Informatik I, RWTH Aachen
Ahornstraße 55, 52074 Aachen, Germany
{hjb,bongartz}@cs.rwth-aachen.de

Abstract. The protein folding problem, i.e., the computational prediction of the three-dimensional structure of a protein from its amino acid sequence, is one of the most important and challenging problems in computational biology. Since a complete simulation of the folding process of a protein is far too complex to handle, one tries to find an approximate solution by using a simplified, abstract model. One of the most popular models is the so-called HP model, where the hydrophobic interactions between the amino acids are considered to be the main force in the folding process, and furthermore the folding space is modelled by a two- or three-dimensional grid lattice.
In this paper, we will present some approximation algorithms for the protein folding problem in the HP model on an extended grid lattice with plane diagonals. The choice of this kind of lattice removes one of the major drawbacks of the original HP model, namely the bipartiteness of the grid which severely restricts the set of possible foldings. Our algorithms achieve an approximation ratio of $\frac{26}{15} \approx 1.733$ for the two-dimensional and of $\frac{8}{5} = 1.6$ for the three-dimensional lattice. This improves significantly over the best previously known approximation ratios for the protein folding problem in the HP model on any lattice.

1 Introduction

Proteins are biological molecules which are responsible for many functions in all living organisms. A protein consists of a chain of amino acid residues. There are twenty different amino acids from which all proteins are built, the linear sequence of these amino acids is called the *primary structure* of the protein. Moreover, under physiological conditions every protein folds into a unique specific three-dimensional structure, called the *native tertiary structure*. This native structure determines the function of the protein.

One of the most important and challenging problems in molecular biology with many applications, e.g. in the area of pharmaceutics and drug design, is

* This work was partly done while the authors visited the group of Prof. Peter Widmayer at ETH Zurich.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 227–238, 2004.
© Springer-Verlag Berlin Heidelberg 2004

to find out this native three-dimensional structure of a protein. This can in principle be done by experimental methods like X-ray cristallography or NMR spectroscopic analysis, but these methods are very slow and expensive. Thus, the goal is to find a computational method for predicting the native tertiary structure of a protein, given the linear sequence of the amino acid residues. This task is called the *protein folding problem.*[1] Any possible folding of a protein in the three-dimensional space is called a *conformation,* and the native structure is believed to be the conformation with the lowest free energy. But there has not yet been developed any complete formal model taking exactly into account all forces that are involved in the folding of the protein, and furthermore such a complete model would most probably be far too complex to handle. Therefore, a lot of different simplified models were considered in the literature.

The discrete model of protein folding we will analyse in this paper is based on the so called *HP model* which was introduced by Dill [9,10]. In the HP model, a protein is treated as its sequence of amino acids, and the molecule is further abstracted by considering only a certain characteristic of each amino acid, namely whether it is hydrophobic or hydrophilic (polar). This abstraction is motivated by the assumption that the hydrophobic interactions between the hydrophobic amino acids are the strongest force in the process of protein folding. This assumption seems to be well motivated by experiments, most proteins whose three-dimensional structure is known tend to fold in such a way that the hydrophobic amino acids form a core of the molecule, surrounded by the polar amino acids.

In this model, the space in which the protein folds is discretized in terms of a two-dimensional or three-dimensional grid lattice. Any conformation of the protein corresponds to an embedding of the amino acid sequence into the grid, where every position in the amino acid sequence is assigned to one point of the lattice such that adjacent positions in the sequence are assigned to adjacent points of the lattice, and no two positions of the sequence are assigned to the same lattice point.

The only force taken into account by this model for calculating the free energy is the interaction between the hydrophobic amino acids of the protein. The assumption made by the model is that two hydrophobic amino acids interact if and only if they are spatially close to each other. For scoring a conformation, i.e., an embedding of the protein into the lattice, the model counts the number of pairs of amino acids that are embedded onto neighboring points in the lattice, but are not neighbors in the amino acid sequence themselves. Such a pair is called a *contact.* The goal is now to find a conformation maximizing this score (and thus minimizing the free energy of the protein).

One might think that this model oversimplifies the biological problem and indeed it does not take into account many aspects of protein folding. On the

[1] This approach of calculating the tertiary structure of a protein from its primary structure relies on experiments by Anfinsen and others [2,3] that have shown that for most proteins all information needed for folding into the native tertiary structure is indeed contained in the amino acid sequence alone.

other hand, already the problem in this very abstract and simple model turns out to be NP-hard for the two-dimensional [8] as well as for the three-dimensional grid lattice [5]. For more complex models many hardness results are known, too [13,4,17,19,7]. For the protein folding problem in the HP model there are many attempts known to attack the problem, including approximation algorithms [12, 18], constraint programming algorithms, local search, genetic algorithms, and other heuristics. An overview of the existing literature is given in the survey articles [7,11].

Nevertheless, the HP model has a serious shortcoming, namely that contacts can only be established between positions of different parity in the amino acid sequence, i.e., it is impossible to have a contact between two hydrophobic amino acids, if they both are on even (resp. both on odd) positions of the sequence. The approximation algorithms in [12,18] all depend on a lower bound which directly results from this parity constraint.

To overcome this drawback of the HP model, one can use an extended lattice as a model for the folding space. There are several lattices considered for protein folding problems in the literature [14,1,15]. The lattice we will consider here is similar to the grid lattice, but the adjacency in the lattice is extended to the diagonals of each basic square in the grid (but not to the steric diagonals in the three-dimensional case). Note that due to the extension of the grid by diagonals the resulting lattice does no longer exhibit the unnatural parity constraint of the original grid lattice. This grid lattice with diagonals was already considered by Heun [15], but only for a variant of the HP model, called the HP side chain model, introduced in [14], where every amino acid is modeled by a backbone part and a side chain which are embedded into two separate (but neighboring) points of the lattice.

The computational complexity of the protein folding problem in the HP model on the grid lattice with diagonals, *HPd model* for short, is unknown, but due to its similarity to the original HP model we conjecture that also this variant of protein folding is computationally hard. In this paper we present some approximation algorithms for protein folding in the HPd model in two as well as in three dimensions. The achieved approximation ratios of our algorithms are significantly better than those of all other known algorithms for protein folding in the different variants of the HP model (without side chains). For the two-dimensional case, our algorithm achieves a worst-case approximation ratio of $\frac{26}{15} \approx 1.733$, for the three-dimensional case of $\frac{8}{5} = 1.6$.

The paper is organized as follows. In the next section we will formally define the model and the protein folding problem. In Section 3 we will present our algorithms for the two-dimensional case, and in Section 4 we will briefly sketch how to adapt them for the three-dimensional case. We will conclude the paper in Section 5 by formulating some open problems. Due to space constraints, most of the proofs, and the detailed description of the algorithms for the three-dimensional case are omitted in this extended abstract.

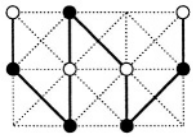


Fig. 1. An embedding of the string 011010110 into the lattice $\mathcal{L}_{2D}$.

2 Preliminaries

In the following we will treat a protein as its sequence of amino acids, thus as a string over a finite alphabet, and we will further abstract the molecule by considering only a certain characteristic of each amino acid, namely whether it is hydrophobic or hydrophilic (polar). We will therefore consider a string over the alphabet {0, 1}, where 0 denotes a polar and 1 denotes a hydrophobic amino acid. The lattice that will serve as a spatial model for the folding of the protein can be formally defined as follows.

Definition 1. *The* two-dimensional grid lattice with diagonals *is the infinite graph* $\mathcal{L}_{2D} = (V, E)$ *with vertex set* $V = \mathbb{Z}^2$ *and edge set* $E = \{\{x, x'\} \mid x, x' \in \mathbb{Z}^2, |x - x'|_2 \leq \sqrt{2}\}$, *where* $|\cdot|_2$ *denotes the Euclidean norm.*

The three-dimensional grid lattice with plane diagonals *is the infinite graph* $\mathcal{L}_{3D} = (V, E)$ *with vertex set* $V = \mathbb{Z}^3$ *and edge set* $E = \{\{x, x'\} \mid x, x' \in \mathbb{Z}^3, |x - x'|_2 \leq \sqrt{2}\}$.

Note that for simplicity we defined the lattice as an infinite graph although we will need only a finite subgraph of it for each embedding of a protein. Note that in the three-dimensional case the steric diagonals are not included in the lattice.

We will now define the allowed embeddings of a protein into this lattice in our HPd model. Intuitively, the embedding maps the symbols of the string onto a subset of the vertices of the lattice in such a way that adjacent symbols in the string are also adjacent in the embedding. In Figure 1, an example of the embedding of a protein string into the two-dimensional lattice $\mathcal{L}_{2D}$ is shown. In this figure, the embedded ones of the string are drawn as black circles, the zeros as white circles, and the fat edges denote the adjacencies in the string. Such an embedding can be formally defined as follows.

Definition 2. *Let* $p = p_1 \dots p_m$ *be a string of length* m *over the alphabet* {0,1}, *and let* $\mathcal{L} = (V, E) \in \{\mathcal{L}_{2D}, \mathcal{L}_{3D}\}$ *be a lattice. An* embedding *of* p *into* $\mathcal{L}$ *is an injective function* $\phi : \{1, \dots, m\} \to V$ *from the positions of the string to the vertices of the lattice that assigns adjacent positions in* p *to adjacent vertices in* $\mathcal{L}$, *i.e.,* $\{\phi(i), \phi(i+1)\} \in E$ *for all* $1 \leq i \leq m-1$. *These edges* $\{\phi(i), \phi(i+1)\} \in E$ *for* $1 \leq i \leq m-1$ *are called* binding edges.

An embedding of p *into* $\mathcal{L}$ *is called a* conformation, *if no two binding edges cross each other (see Figure 2(a)).*

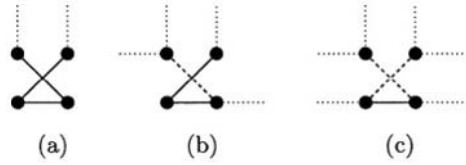


Fig. 2. In (a) a crossing of two binding edges is shown, this situation is forbidden in a valid conformation, in (b) and (c) the crossings between a contact edge and a binding edge as well as between two contact edges are shown, these crossings are allowed in our model.

An edge $\{x, x'\}$ *of* $\mathcal{L}$ *is called a* contact edge, *if it is no binding edge, but there exist* $i, j \in \{1, \ldots, m\}$ *such that* $\phi(i) = x$, $\phi(j) = x'$, *and* $p_i = p_j = 1$ *(see Figure 2(b) and (c)).*

The biological motivation for forbidding the crossing of binding edges is that these edges represent chemical bindings and such a crossing is not possible due to the spatial constraints of these chemical bindings.

Whenever a conformation of a string is given, we will call a vertex of the lattice to which there was assigned a one [zero] by this conformation a *one-vertex* [*zero-vertex*] or simply a *one* [*zero*]. The vertices of the lattice to which there was not assigned a one or a zero are called *unused.* A binding edge connecting a one with a zero will be called an *alternation edge* and a non-binding edge adjacent to a one that is no contact edge is called a *loss edge.*

The *protein folding problem in the HPd model* is now to find a conformation of a given string in the lattice $\mathcal{L}_{2D}$ or $\mathcal{L}_{3D}$ with a maximum number of contact edges. We will call this problem the HPd-2D [HPd-3D] problem in the two-dimensional [three-dimensional] case.

Throughout this paper we will use the following convention for figures: Black circles denote ones, white circles denote zeros, solid lines denote binding edges, and dashed lines denote contact edges. Dotted lines are used to denote parts of the conformation that are not considered in this figure.

In the next sections we will attack the protein folding problem in the HPd model by proposing some approximation algorithms. Please note that we will only analyze the asymptotic approximation ratio of our algorithms. This will clarify the analysis since we do not have to consider situations on the border of the embedding in detail which may result in a small constant increase or decrease of the number of contact edges in the constructed feasible solution, since these will be negligible for increasing input lengths. Furthermore, we will assume that every input string starts and ends with a zero. This will simplify the analysis of our algorithms and does not affect their asymptotic approximation ratio.

3 Approximation Algorithms for Two Dimensions

In this section we will present some approximation algorithms for the HPd-2D problem. To measure the approximation ratio we have to count the number of

contact edges achieved by our algorithms and to compare it to the maximum number of possible contact edges. We will always locally count for every one-vertex of the lattice the number of incident contact edges, and we will call these incident contact edges the *contacts* of this one. Every edge incident to a one that is neither a contact nor a binding edge will be called a *loss* of this vertex.[2] By summing up the number of contacts over all ones, we will count every contact edge exactly twice. Since we will use this way of counting both for the contacts achieved by our algorithm and for the hypothetically possible contacts, this will not affect the approximation ratio.

We will now start with the presentation of two upper bounds on the number of possible contacts for any one. The first upper bound simply follows from the fact that every vertex in the lattice has eight neighbors, and that every one is incident to two binding edges.

Lemma 1. *Let $p \in \{0,1\}^*$ be a string starting and ending with a zero, let ϕ be a conformation of p in $\mathcal{L}_{2D}$. Then any one can have at most six contacts.* □

But this upper bound cannot be reached in general. More precisely, it is possible to show that every alternation edge between zeros and ones in the given string on average guarantees at least one third of a loss edge and thus prevents at least one third of a contact.

Lemma 2. *Let $p = 0^+b_1 0^+ b_2 0^+ \ldots 0^+ b_k 0^+$ be an input string for the HPd-2D problem, where $b_i \in \{1\}^+$ for $1 \leq i \leq k$, let $n = \sum_{i=1}^{k} |b_i|$ be the number of ones in p. Then the overall number of contacts, summed up over all ones, in any conformation is at most $6 \cdot n - \frac{2}{3} \cdot k$.*

Proof idea. To show the claim, we first count the number of inevitable loss edges in the neighborhood of any alternation edge. Although the neighborhoods of different alternation edges may overlap, we can then show in a second step that every loss edge can occur in the neighborhood of at most four alternation edges. The details of the proof are omitted in this extended abstract. □

The next lemma shows that the upper bound from Lemma 2 is asymptotically the best possible general upper bound for the number of achievable contacts.

Lemma 3. *For all $n \in \mathbb{N}$, let $p_n = (01011010)^{2n+1}$. Then every conformation for p_n has $6 \cdot (2n+1)$ alternation edges, and there exists a conformation for p_n with $2 \cdot (2n+1) + 10$ loss edges.*

Proof. It is obvious that every conformation for p_n has exactly $6 \cdot (2n+1)$ alternation edges. We will now analyse a conformation with a minimal number of loss edges. As an example, a possible conformation for p_3 is shown in Figure 3. From this figure, it is easy to see that the sub-conformation of each block 01011010 contributes exactly two loss edges, which are marked by dashed lines in Figure 3,

[2] Thus, the losses of a one are the incident loss edges of this vertex, i.e., the edges connecting the vertex to a zero not adjacent in the string or to an unused vertex of the lattice.

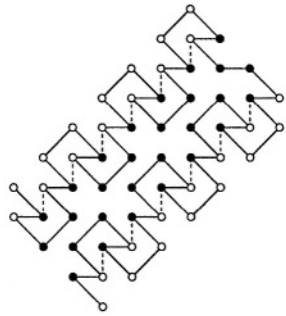


Fig. 3. An optimal conformation of the string $p_3 = (01011010)^7$.

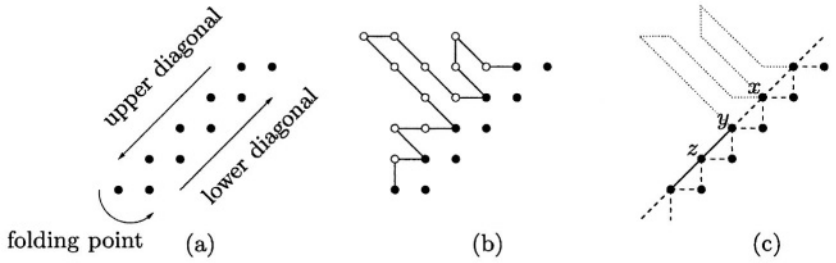


Fig. 4. The overall strategy of algorithm Simple2D is shown in (a), the arrangement of the side-arms is shown in (b), and (c) shows the achieved contacts.

except for the block around the "folding point". Additionally, this conformation has twelve loss edges between the ones at the ends of the conformation and their unused neighbors. This amounts to $2 \cdot (2n + 1) + 10$ loss edges in total. It is straightforward to generalize this conformation to higher values of n. □

Now we will present three approximation algorithms for the HPd-2D problem. We start with a simple 3-approximation algorithm.

The idea of this simple algorithm, denoted as *Simple2D* in the following, is to arrange all ones occurring in the input string along two diagonals. This will be done by arranging the ones in a prefix of the string up to a certain folding point on the upper diagonal setting the zeros aside, and arranging the ones of the corresponding suffix on the lower diagonal. This idea is depicted in Figure 4(a).

For this simple strategy the folding point can be determined by dividing the number of ones by two. Furthermore, it is possible to arrange the zeros in side-arms orthogonal to the diagonals in such a way that no interference occurs, as shown in Figure 4(b). Although we only presented the required arrangement for the upper diagonal in the figure, it is obvious that the situation on the lower diagonal is symmetric.

Obviously, the algorithm Simple2D computes a feasible solution for the HPd-2D problem in linear time. Let us now investigate the asymptotic approximation ratio achieved by algorithm Simple2D. Since all ones in the string lie on two

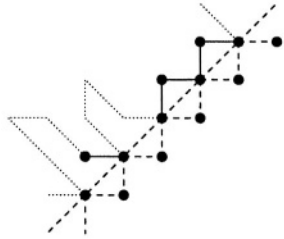


Fig. 5. The algorithm FourDiagonals guarantees four contacts for each one on the main diagonals.

adjacent diagonals, every one has at least two contacts. This is visualized in Figure 4 (c). Since each one can participate in at most six contacts we can directly conclude the following result.

Theorem 1. *The algorithm Simple2D is an asymptotic* 3*-approximation algorithm for the HPd-D2 problem.* □

If we take a closer look, we see that in general the ones may have four contacts, two to ones from the same diagonal and another two to ones from the other diagonal (see for example the one labeled by x in Figure 4 (c)). Thus, for substrings of many consecutive ones we will lose contacts since adjacent ones in the string lie on adjacent positions along the diagonal. To overcome this problem we will in our second algorithm place every second one in such a block of consecutive ones in the next (outer) diagonal. In this way we will achieve four contacts for all ones remaining on the main diagonals, but now we have to deal with those ones that we laid aside. This situation is depicted in Figure 5.

Following this improved strategy we can summarize our second algorithm as follows.

Algorithm FourDiagonals

Input: A string p over $\{0, 1\}$.

1. Decompose p into maximal blocks of consecutive ones separated by zeros, i.e., $p = 0^+b_1 0^+ b_2 0^+ \ldots 0^+ b_k 0^+$, $b_i \in \{1\}^+$. Let $g = \sum_{i=1}^{k} \lceil |b_i|/2 \rceil$, and $f = \lceil g/2 \rceil$.
2. Choose the folding point F as the first position in p after the fth one. Denote by $\mathrm{pref}_F(p)$ the prefix of p up to position F and by $\mathrm{suff}_F(p)$ the corresponding suffix.
3. For every block in $\mathrm{pref}_F(p)$ [$\mathrm{suff}_F(p)$], adjust the ones at odd positions in the block along the upper [lower] main diagonal, putting the ones at even positions in the block aside to the next upper [lower] diagonal, and arranging the side-arms of intermediate zeros as in algorithm Simple2D.

The following theorem shows that the algorithm FourDiagonals indeed gives an improved approximation ratio.

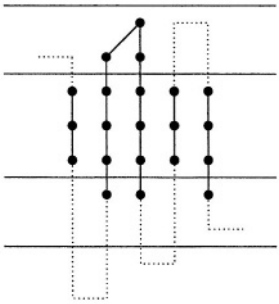


Fig. 6. A small example for a conformation found by the algorithm SevenRows.

Theorem 2. *The algorithm FourDiagonals is a linear-time asymptotic* $\frac{26}{15}$*-approximation algorithm for the HPd-2D problem.* □

The proof of Theorem 2 is based on counting the number of contacts for each one, and comparing the total number of contacts to the upper bound from Lemma 2. We are able to show that all ones except for the last one in a block of odd length achieve four contacts on average. A detailed proof of Theorem 2 is omitted in this extended abstract.

Note that in many cases the algorithm FourDiagonals will even obtain better results than this approximation ratio of $\frac{26}{15}$. Among the worst-case inputs for this algorithm are obviously strings with many odd-length maximal blocks of ones, or especially with many triples of ones. One opportunity to improve the approximation ratio is therefore to design another algorithm that guarantees more contacts on average for every one inside a triple or another block of odd length.

In the sequel of this section we will show that this is indeed possible by developing an algorithm that achieves at least $\frac{30}{7} \approx 4.29$ contacts on average for each one inside a maximal block of length ≥ 3. Unfortunately, this algorithm does not guarantee any contact for all solitary ones and pairs of ones.

The basic idea of this algorithm, called SevenRows, is to place all blocks of consecutive ones of length ≥ 3 side by side into seven consecutive rows of the lattice as shown for a small example in Figure 6, folding long blocks in an appropriate way such that the three middle rows consist of ones only. The solitary ones and the pairs of ones are treated as zeros by this algorithm. The actual folding rules are quite complicated. Due to space restrictions, they are omitted here.

The algorithm SevenRows guarantees more contacts for ones inside long blocks than the algorithm FourDiagonals, but it does not at all take into account solitary ones and pairs of ones. The following theorem shows that, in spite of this drawback, it improves over the algorithm FourDiagonals, if there are not too much solitary ones and pairs of ones.

Theorem 3. *Let* n *be the number of ones in the given input and let* $\bar{k}$ *be the sum of the number of solitary ones and the number of ones in pairs. Then the*

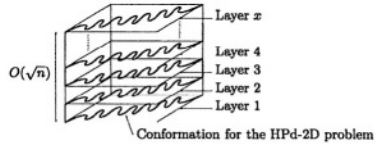


Fig. 7. Schematic view how to stack layers of two-dimensional solutions to form a three-dimensional conformation for the HPd-3D problem.

algorithm SevenRows achieves an asymptotic approximation ratio of $\frac{7}{30} \cdot \frac{6n - \frac{1}{3} \cdot \bar{k}}{n - \bar{k}}$. *This improves over the guaranteed approximation ratio of the algorithm FourDiagonals, if* $\bar{k} < \frac{30}{149} \cdot n \approx 0.201 \cdot n$. □

Please note that in practice the algorithm SevenRows will be useful also for input strings where the fraction of solitary ones and ones in pairs is greater as $\frac{30}{149}$, since our approach of treating all long blocks as blocks of length 7 in the proof was a rather rough estimation. Using the calculations in the proof of Theorem 3, one can exploit any knowledge about the actual distribution of the block lengths to prove a better approximation ratio.

4 Approximation Algorithms for Three Dimensions

In this section we will briefly sketch how our approach described in the previous section can be extended to the three-dimensional HPd problem.

We have designed several different algorithms, and we can guarantee an approximation ratio of $\frac{8}{5}$ for the best of them. This is the best known approximation ratio for any protein folding problem in a three-dimensional HP model without side-chains. The best previously known approximation ratio is $\frac{8}{3}$ for the standard HP-3D model [12] and $\frac{5}{3}$ for the HP model on the triangular lattice [1].

We have used the following general approach to extend algorithms for the HPd-2D problem to algorithms for the HPd-3D problem.[3]

Let $\mathcal{A}$ be any algorithm for the HPd-2D problem. For an input p for the HPd-3D problem run $\mathcal{A}$ on p. Let l be the length of the resulting conformation $\phi_{\mathcal{A}}$, i.e., the distance between the two most apart vertices in $\phi_{\mathcal{A}}$. Now, fold the conformation $\phi_{\mathcal{A}}$ into $\lceil \sqrt{l} \rceil$ layers to obtain a three-dimensional conformation of p. A schematic view of this procedure is presented in Figure 7.

The application of this idea to our algorithms described in Section 3 gives the following results. We consider three algorithms, namely Simple-3d as the three-dimensional version of Simple-2d, FourDiagonals-3d as a three-dimensional version of FourDiagonals, and an algorithm ThreeRows-3d, whose two-dimensional counterpart ThreeRows works similarly to SevenRows, but places all blocks of ones of length ≥ 3 into three consecutive rows of the lattice.[4]

[3] This is a well-known idea also considered for protein folding problems on other lattice types.

[4] The use of three rows instead of seven rows has the advantage that we can guarantee more contacts to the layers above and below in the three-dimensional conformation.

Theorem 4.

(a) Simple-3d is an asymptotic 2*-approximation algorithm for the HPd-3D problem.*

(b) FourDiagonals-3d is an asymptotic 1.6*-approximation algorithm for the HPd-3D problem.*

(c) ThreeRows-3d is an asymptotic $\left(\frac{3}{38} \cdot \frac{16n - \bar{k}}{n - \bar{k}}\right)$*-approximation algorithm for the HPd-3D problem, where* $\bar{k}$ *denotes the number of ones in blocks of length* 1 *and* 2*. This ratio improves over FourDiagonals-Sd for* $\bar{k} < \frac{64}{289} \cdot n \approx 0.221 \cdot n$. □

5 Open Problems

One direction of future research could be to refine the considered model by giving a specific weight to the contacts, where e.g. a contact via a diagonal edge of the lattice is weighted less, since this edge is longer, or where the different hydrophobicity of the amino acids is taken into account.

Another problem that still remains open is the question whether the considered problems are NP-hard. Moreover, there is no result known about the hardness of approximation for any variant of the HP model.

References

1. R. Agarwala, S. Batzoglou, V. Dančík, S. E. Decatur, S. Hannenhalli, M. Farach, S. Muthukrishnan, and S. Skiena: Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model. *Journal of Computational Biology* 4(2), 1997, pp. 275–296.
2. C. B. Anfinsen: Principles that govern the folding of protein chains. *Science* 181(4096), 1973, pp. 223–230.
3. C. B. Anfinsen, E. Haber, M. Sela, and F. H. White: The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proceedings of the National Academy of Sciences, USA* 47, 1961, pp. 1309–1314.
4. J. Atkins and W. E. Hart: On the intractability of protein folding with a finite alphabet of amino acids. *Algorithmica* 25, 1999, pp. 279–294.
5. B. Berger and F. T. Leighton: Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Proc. of the 2nd Annual Internat. Conference on Research in Computational Molecular Biology (RECOMB'98),* 1998, pp. 30–39.
6. H. S. Chan and K. A. Dill: The protein folding problem. *Physics today,* 1993, pp. 24–32.
7. V. Chandra, A. DattaSharma, and V. S. A. Kumar: The algorithmics of folding proteins on lattices. *Discrete Applied Mathematics* 127, 2003, pp. 145–161.
8. P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis: On the complexity of protein folding. *Journal of Computational Biology* 5(3), 1998, pp. 423-466.
9. K. A. Dill: Theory for the folding and stability of globular proteins. *Biochemistry* 24, 1985, p. 1501.

10. K. A. Dill, S. Bromberg, K. Yue, K. Fiebig, D. Yee, P. Thomas, and H. Chan: Principles of protein folding – a perspective from simple exact models. *Protein Science* 4, 1995, pp. 561–602.
11. H. J. Greenberg, W. E. Hart, and G. Lancia: Opportunities for combinatorial optimization in computational biology. *INFORMS Journal of Computing,* to appear.
12. W. E. Hart and S. Istrail: Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal. *Journal of Computational Biology* 3(1), 1996, pp. 53–96.
13. W. E. Hart and S. Istrail: Robust proofs of NP-hardness for protein folding: General lattices and energy potentials. *Journal of Computational Biology* 4(1), 1997, pp. 1–22.
14. W. E. Hart and S. Istrail: Lattice and off-lattice side chain models of protein folding: linear time structure prediction better than 86% of optimal. *Journal of Computational Biology* 4(3), 1997, pp. 241–259.
15. V. Heun: Approximate protein folding in the HP side chain model on extended cubic lattices. *Discrete Applied Mathematics* 127(1), 2003, pp. 163-177. Extended abstract in *Proc. of the 7th Annual European Symposium on Algorithms (ESA '99),* Springer LNCS 1643, 1999, pp. 212–223.
16. G. Mauri, A. Piccolboni, and G. Pavesi: Approximation algorithms for protein folding prediction. *Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '99),* 1999, pp. 945–946.
17. A. Nayak, A. Sinclair, and U. Zwick: Spatial Codes and the hardness of string folding problems. *Journal of Computational Biology* 6(1), 1999, pp. 13–36.
18. A. Newman: A new algorithm for protein folding in the HP model. *Proc. of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02),* 2002, pp. 876–884.
19. J. T. Ngo, J. Marks, and M. Karplus: Computational complexity, protein structure prediction, and the Levinthal paradox. In: K. Merz, Jr. and S. LeGrand (eds.): *The Protein Folding Problem and Tertiary Structure Prediction,* Birkhäuser Boston, 1994, pp. 433–506.

Optimization, Games, and Quantified Constraint Satisfaction

Hubie Chen and Martin Pál*

Department of Computer Science
Cornell University
Ithaca, NY 14853, USA
{hubes,mpal}@cs.cornell.edu

Abstract. Optimization problems considered in the literature generally assume a passive environment that does not react to the actions of an agent. In this paper, we introduce and study a class of optimization problems in which the environment plays an active, adversarial role and responds dynamically to the actions of an agent; this class of problems is based on the framework of quantified constraint satisfaction. We formalize a new notion of approximation algorithm for these optimization problems, and consider certain restricted versions of the general problem obtained by restricting the types of constraints that may appear. Our main result is a dichotomy theorem classifying exactly those restricted versions having a constant factor approximation algorithm.

1 Introduction

Optimization is what we do all of the time. We strive for better products, faster cars, more efficient networks, and cheaper solutions. Research in both continuous and discrete optimization has seen tremendous growth over the past two decades. It has long been known that many problems of interest are intractable to solve to optimality; this fact has been the driving force behind the study of approximation algorithms.

Most optimization problems considered by researchers assume a passive environment that does not react to the actions of an agent. The state of the environment is assumed to be fully known (deterministic optimization), partially random (stochastic optimization), or unknown in advance but independent of an agent's actions (on-line algorithms). In this paper, we introduce and study a class of optimization problems in which the environment plays an active role. In contrast to previously considered optimization problems, the problems we study are capable of modelling scenarios where the environment actively responds to the moves of an agent in an adversarial manner.

The framework of problems that we study is based on the *quantified constraint satisfaction problem* (QCSP). An instance of the QCSP consists of a quantified formula, which is a set of constraints along with an ordered list of

* Research supported by ONR grant N00014-98-1-0589.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 239–250, 2004.
© Springer-Verlag Berlin Heidelberg 2004

decision variables, each having an existential or universal quantifier associated with it. A quantified formula can be viewed as a game between two players, a *universal player,* who strives to falsify a constraint, and an *existential player,* who attempts to satisfy all constraints; these players set their variables in the specified order, and the existential player is said to have a *winning strategy* if she can always succeed in satisfying all of the constraints, despite the moves of the universal player. In the vanilla QCSP, the problem is to decide whether or not the existential player has a winning strategy. In our framework, which generalizes the QCSP, the existential player strives to satisfy all constraints and, subject to this, minimize an objective function. No assumptions are made about the environment (the universal player), which may set variables in an adversarial manner, trying to invalidate a constraint or worsen the objective function value.

We formalize and prove results concerning a new notion of approximation algorithm for these game-based minimization problems. How do we formalize this notion? When universal quantification is removed from the described games, a strategy is simply a static assignment to the variables, and is winning if it satisfies all of the constraints. The established notion of approximation algorithm for this special case is a polynomial-time algorithm that always computes an assignment satisfying all constraints, when such an assignment exists; such an algorithm is said to achieve approximation ratio α if on every satisfiable problem instance, the computed assignment has cost no more than α times the minimum cost assignment. When we include universal quantification, an approximation algorithm still ought to efficiently generate a winning strategy. However, we impose computational bounds not just on the generation of the strategy, but also on the generated strategy itself, in order to ensure that it is effectively usable. (A strategy specified succinctly as the description of a Turing machine having an exponential running time is an example of a strategy that should not be considered acceptable.) We therefore mandate that an approximation algorithm output a winning strategy (when one exists) specified as a boolean circuit, in polynomial time; such a circuit, of course, is necessarily of polynomial size, and hence can itself be evaluated in polynomial time. We take the (arguably pessimistic!) approach of defining the cost of a strategy to be the maximum cost assignment it produces, over all adversaries; that is, a strategy is graded according to its worst adversary.[1] An approximation algorithm is considered to achieve approximation ratio α if on every problem instance (with a winning strategy), the computed strategy has cost no more than α times the minimum cost winning strategy. It is straightforward to verify that this notion of approximation algorithm generalizes the discussed, established notion for the case where there is no universal quantification.

The particular class of problems we study in this paper is as follows. We investigate quantified formulas where all decision variables are over the boolean domain $\{0, 1\}$ and have weights associated with them, and the objective function to be minimized is the total weight of the variables set to 1. We search for

[1] We leave the investigation of other strategy measures as an interesting issue for future work.

constant factor approximable classes of these formulas by studying restricted classes of quantified formulas where the types of constraints that may appear must come from a fixed *constraint language.* This form of restriction was first studied by Schaefer [9] and has since been investigated heavily, in several different contexts (see [5] for examples); it is widely acknowledged as a useful and rich means of obtaining restricted classes of the constraint satisfaction problem and its variants. Khanna et al. [8] have studied the non-quantified versions of the class of problems we consider here, and obtained a theorem classifying, for all constraint languages *S,* the approximability of such problems having constraints built using *S.* Their motivation was to systematically classify and unify the many existing results on minimization problems.

Our main result is a full dichotomy theorem characterizing the constraint languages for which a constant factor approximation exists for our QCSP minimization problems. Interestingly, every non-trivial constraint language for which there is a constant factor approximation in the non-quantified case is also constant factor approximable in the quantified case. Since an α-approximation algorithm for the quantified case is also an α-approximation algorithm for the non-quantified case, our results strictly generalize the constant factor approximation results given in [8].

2 Preliminaries

Let $[n]$ denote the set containing the first n positive integers, $\{1, \ldots, n\}$.

A subset of $\{0,1\}^k$ (for some $k \geq 1$) is a *logical relation* of arity k. A *constraint* is an expression of the form $R(w_1, \ldots, w_k)$ where R is a logical relation of arity k and the w_i are variables; it is satisfied by a mapping $f : V \to \{0,1\}$ defined on all of its variables if $(f(w_1), \ldots, f(w_k)) \in R$.

A *quantified formula* is an expression of the form $\phi = Q_1 v_1 \ldots Q_n v_n \mathcal{C}$ where for all $i \in [n]$, Q_i is quantifier from the set $\{\exists, \forall\}$ and v_i is a variable; and, $\mathcal{C}$ is a set of constraints over the variable set $V = \{v_1, \ldots, v_n\}$. A weighted quantified formula is a pair $\Phi = (\phi, w)$, where $w : V \to \mathbb{N}$ is a mapping assigning every variable of ϕ a nonnegative weight.

Let $|\phi|_\exists$ ($|\phi|_\forall$) denote the number of existentially (universally) quantified variables in a quantified formula ϕ. For all $i \in [|\phi|_\exists]$, let Y_i^ϕ denote the set of universally quantified variables coming before the ith existentially quantified variable (in the quantification order of ϕ), and for all $j \in [|\phi|_\forall]$, define X_j^ϕ dually.

A quantified formula can be viewed as a two-player game where an existential player and universal player set their variables in the specified order; the existential player wins if all constraints are satisfied, and the universal player wins if some constraint is falsified. Formally, define an existential strategy (for ϕ) to be a sequence of mappings $\Sigma = \{\sigma_i : \{0,1\}^{Y_i^\phi} \to \{0,1\}\}_{i \in [|\phi|_\exists]}$, and similarly define a universal strategy (for ϕ) to be a sequence of mappings $\Pi = \{\pi_i : \{0,1\}^{X_i^\phi} \to \{0,1\}\}_{i \in [|\phi|_\forall]}$. Define a *game* to be a triple (Φ, Σ, Π) where $\Phi = (\phi, w)$ is a weighted quantified formula and Σ (Π) is an existential strategy (universal strategy) for ϕ. The *outcome* of a game $G = (\Phi, \Sigma, \Pi)$ is is

the assignment $A_G : V \to \{0,1\}$ to the variable set V of the formula defined by setting the variables in the order specified by ϕ, according to the strategies Σ and Π. A game $G = (\Phi, \Sigma, \Pi)$ is a *win* if the outcome $A_G : V \to \{0,1\}$ satisfies every constraint $R(w_1, \ldots, w_k)$ in ϕ. An existential strategy Σ is *winning* for a formula Φ if the game (Φ, Σ, Π) is a win for all universal strategies Π. A quantified formula is *satisfiable* if there is a winning strategy for it.

The *cost* of a game G is the total weight of all variables set to true, that is, $\mathsf{cost}(G) = \sum_{A_G(v)=1} w(v)$. In the problem that we will consider, the goal of the existential player is to win the game, and subject to this, minimize the cost of the game. The *cost* of a winning existential strategy Σ on weighted formula Φ is defined to be $\mathsf{cost}(\Phi, \Sigma) = \max_\Pi \mathsf{cost}(\Phi, \Sigma, \Pi)$, that is, the cost of the game where Σ plays a "worst possible" adversary. Note that there is always an *ignorant strategy* Π maximizing the right-hand side expression, where an ignorant strategy is a strategy that contains only constant functions and does not react to its opponent. Finally, the cost of a weighted formula Φ is defined to be $\mathsf{cost}(\Phi) = \min_\Sigma \mathsf{cost}(\Phi, \Sigma)$, where the minimum is taken over all winning existential strategies Σ. That is, the cost of the formula is the cost of the best existential strategy for the formula.

3 Dichotomy Theorem

As discussed in the introduction, the problem we study is that of computing, given a weighted quantified formula Φ, an existential strategy Σ that minimizes $\mathsf{cost}(\Phi, \Sigma)$. We consider restricted versions of this problem where the constraints present in the input quantified formulas must come from a specified *constraint language,* where a constraint language is defined to be a set of logical relations. Formally, when S is a constraint language, we define an instance of the MIN-QCSP(S) problem to be a weighted quantified formula Φ where all of the constraints in Φ have relation from S. We say that MIN-QCSP(S) is α-*approximable* if there exists a polynomial time algorithm that on all satisfiable instances Φ outputs a winning existential strategy Σ (in the form of a sequence of circuits) such that $\mathsf{cost}(\Phi, \Sigma) \leq \alpha \cdot \mathsf{cost}(\Phi)$.

Our main result is a dichotomy theorem on constraint languages, describing exactly those constraint languages S for which MIN-QCSP(S) is constant factor approximable. In order to formulate our dichotomy theorem, we will need to define various classes of constraint languages. These definitions are fairly standard [5]. A constraint is *weakly negative* if it is expressible as a CNF-formula having at most one unnegated variable in each clause; *width-2 affine* if it is expressible as a system of linear equations over the two-element field with at most two variables per equation; *bijunctive* if it is expressible as a 2-CNF formula; and *k-IHS-B+* if it is expressible as a CNF formula where each clause is of the form $\neg v$, $\neg v \vee w$, or $w_1 \vee \ldots \vee w_k$. We extend these definitions to constraint languages by stating, for instance, that a constraint language S is weakly negative if any constraint with relation from S is weakly negative. Also, we will state that a quantified formula

is weakly negative (bijunctive, etc.) if all of its constraints are weakly negative (bijunctive, etc.).

Theorem 1. *Let S be a constraint language.*

1. *If S is weakly negative or width-2 affine, then MIN-QCSP(S) is 1-approximable.*
2. *Else if S is bijunctive, then MIN-QCSP(S) is 2-approximable.*
3. *Else if S is k-IHS-B+, then MIN-QCSP(S) is k-approximable.*
4. *Else MIN-QCSP(S) is not constant factor approximable, unless NP has deterministic algorithms running in time $n^{\log^{O(1)} n}$.*

4 Approximation Algorithms and Strategies

In this section, we give the positive approximability results of our dichotomy theorem. We first give the results for the cases where we have a constant factor approximation but no 1-approximation (that is, the bijunctive and k-IHS-B+ cases), and then give the weakly negative case. (The width-2 affine case is straightforward, and omitted.) Note that in each of the cases, we do not discuss extensively how to actually implement a polynomial-time algorithm which outputs the desired circuits, and leave it to the reader to convince herself that this is possible.

We will use the following notation throughout this section. When $\phi = Qv\phi'$ is a quantified formula and $b \in \{0,1\}$, let $\phi[v \leftarrow b]$ denote the formula obtained by substituting all occurrences of the variable v with b in ϕ. Observe that if the formula $\phi = Qv\phi'$ is bijunctive, then $\phi[v \leftarrow b]$ is also bijunctive (and likewise for k-IHS-B+ and weakly negative formulas). Extending this notation, when $\Phi = (\phi, w)$ is a weighted quantified formula, we use $\Phi[v \leftarrow b]$ to denote the weighted quantified formula consisting of $\phi[v \leftarrow b]$ together with the weight function w restricted to $V - \{v\}$. Also, when $\Phi = (\phi, w)$ is a weighted quantified formula, we extend the weight function w to negations of variables by defining $w(\neg v) = 0$, and then to sets of literals S by defining $w(S) = \sum_{l \in S} w(l)$. (Recall that a literal is a variable v or the negation of a variable $\neg v$.)

4.1 The Bijunctive Case

By definition, any bijunctive constraint is equivalent to a 2-CNF formula. Hence, without loss of generality we assume that all constraints in a bijunctive formula have the form $u \Rightarrow v$ where u and v are literals. We define the implication graph $G(\phi)$ of a bijunctive formula ϕ with variable set V to be the directed graph with $2|V|$ vertices, containing a vertex for each variable $v \in V$ and a vertex for each negated variable $\neg v$. For each constraint $u \Rightarrow v$, it has two edges, $u \rightarrow v$ and $\neg v \rightarrow \neg u$. The closure $T(v)$ of a vertex v in $G(\phi)$ is simply the set of vertices reachable from v via a directed path; we consider the vertex v to be reachable from itself.

To give a strategy for this case, it will be useful to have a subroutine that decides whether or not a quantified formula is satisfiable. This decision can be performed in polynomial time, and hence by polynomial size circuits.

Theorem 2. *[1] The quantified bijunctive formula $\phi = Q_1 v_1 \ldots Q_n \mathcal{C}$ is satisfiable if and only if none of the following three conditions hold for the graph $G(\phi)$:*

1. *An existential literal l is in the same strongly connected component as $\neg l$.*
2. *A universal literal l is in the same strongly connected component as some existential vertex l' such that the variable of l' comes before the variable of l in the quantification order of ϕ.*
3. *There is a path from a universal literal l to another universal literal l' (this includes the possibility $l = \neg l'$).*

It is straightforward to verify that the conditions of Theorem 2 can be checked in polynomial time.

The strategy. We give a description of the strategy, which we denote by Σ_0. As the game progresses and variables are being set, Σ_0 transforms the formula Φ by replacing variables with the corresponding values. When it is due to make a move and set a variable v, the strategy Σ_0 first checks if both values $\{0, 1\}$ are feasible–that is, if both $\phi[v \leftarrow 1]$ and $\phi[v \leftarrow 0]$ are true. If only one value is feasible, v is set to that value. Otherwise, a decision has to be made.

The strategy strives to set the variable v to the value that minimizes the "implied" costs. Suppose that Σ_0 is about to set a variable v and that the weight of all vertices "implied by" setting v to 1 is zero, that is, $w(T(v)) = 0$. Assuming that $\phi[v \leftarrow 1]$ is true, there is no harm in setting v to 1, as this choice does not commit the algorithm to paying any charges later in the game. Similarly, if $\phi[v \leftarrow 0]$ is true, and $w(T(\neg v)) = 0$, an optimum move is to set v to false.

In case $\phi[v \leftarrow b]$ is true for $b \in \{0, 1\}$, we simultaneously decrease the weight of vertices in $T(v)$ and $T(\neg v)$ by the same amount, until the weight of one of the sets becomes zero; note that if ϕ is true, $T(v)$ and $T(\neg v)$ must be disjoint. That is, setting $\delta = \min_{b \in \{0,1\}} w(T(v \leftarrow b))$, we decrease the weights of variables in each of $T(v \leftarrow b)$ and $T(v \leftarrow \neg b)$ by δ in an arbitrary way that does not decrease any of the variable weights below zero. (When v is a variable, we use $T(v \leftarrow 1)$ to denote $T(v)$, and $T(v \leftarrow 0)$ to denote $T(\neg v)$.) Note that the total weight of variables has been reduced by $2 \cdot \delta$; this is where the approximation factor of 2 arises. After the reduction of weights has been performed, there is some value b for which $w(T(v \leftarrow b)) = 0$, and we set v to such a b.

The analysis. Note that if the formula ϕ is satisfiable, the strategy Σ_0 playing against any adversary always produces an outcome satisfying all constraints, as it never makes a move that could falsify the formula. The approximation guarantee of Σ_0 is expressed in the following theorem.

Theorem 3. *Let Φ be a weighted bijunctive quantified formula, and assume that Φ is satisfiable. Then* $\mathsf{cost}(\Phi, \Sigma_0) \leq 2 \cdot \mathsf{cost}(\Phi)$.

Proof. We shall prove the theorem by induction on the number of variables, and the sum of the variable weights. We consider three cases.

- $\Phi = \forall y \ldots$
  In this case we have

$$\begin{aligned}\mathsf{cost}(\Phi, \Sigma_0) &\leq \max_b \mathsf{cost}(\Phi[y \leftarrow b], \Sigma_0) \\ &\leq \max_b 2 \cdot \mathsf{cost}(\Phi[y \leftarrow b]) \\ &\leq 2 \cdot \mathsf{cost}(\Phi).\end{aligned}$$

  Note that when $b = 1$, we assume the cost of $\Phi[y \leftarrow b]$ to include $w(y)$, the weight of y.
- $\Phi = \exists x \ldots$, $\Phi[x \leftarrow b]$ is true, and $\Phi[x \leftarrow \neg b]$ is false (where $b \in \{0, 1\}$). In this case we have

$$\mathsf{cost}(\Phi, \Sigma_0) = \mathsf{cost}(\Phi[x \leftarrow b], \Sigma_0) \leq 2 \cdot \mathsf{cost}(\Phi[x \leftarrow b]) = 2 \cdot \mathsf{cost}(\Phi).$$

- $\Phi = \exists x \ldots$, and $\Phi[x \leftarrow b]$ is true for both $b \in \{0, 1\}$.
  There are two subcases.
  If $\delta = \min_{b \in \{0,1\}} w(T(x \leftarrow b)) > 0$, then Σ_0 decreases the weight of variables in each of $T(x \leftarrow 0)$ and $T(x \leftarrow 1)$ by δ. Let Φ' denote the formula with decreased weights. Observe that $\mathsf{cost}(\Phi') + \delta \leq \mathsf{cost}(\Phi)$, since every strategy must pay for setting to true the variables either in $T(x \leftarrow 0)$ or $T(x \leftarrow 1)$. Observe also that $\mathsf{cost}(\Phi, \Sigma_0) \leq \mathsf{cost}(\Phi', \Sigma_0) + 2 \cdot \delta$, because the total decrease in weights from Φ to Φ' is $2 \cdot \delta$. From these two facts it follows that

$$\mathsf{cost}(\Phi, \Sigma_0) \leq \mathsf{cost}(\Phi', \Sigma_0) + 2 \cdot \delta \leq 2 \cdot \mathsf{cost}(\Phi') + 2 \cdot \delta \leq 2 \cdot \mathsf{cost}(\Phi).$$

  If $w(T(x \leftarrow b)) = 0$ for some b, then Σ_0 sets $x \leftarrow b$. In this case we claim

$$\mathsf{cost}(\Phi[x \leftarrow b]) \leq \mathsf{cost}(\Phi),$$

  and hence that Σ_0 has made an optimal move. From the induction hypothesis, the claim implies

$$\mathsf{cost}(\Phi, \Sigma_0) = \mathsf{cost}(\Phi[x \leftarrow b], \Sigma_0) \leq 2 \cdot \mathsf{cost}(\Phi[x \leftarrow b]) \leq 2 \cdot \mathsf{cost}(\Phi).$$

  To justify the claim, take any existential winning strategy Σ which sets $x \leftarrow \neg b$. Based on Σ we can define a new existential winning strategy Σ' that sets all literals in $T(x \leftarrow b)$ to true, and otherwise behaves just like Σ. Let Π' be the worst ignorant adversary for Σ'; it can be verified that $\mathsf{cost}(\Phi, \Sigma', \Pi') \leq \mathsf{cost}(\Phi, \Sigma, \Pi')$, from which it follows that $\mathsf{cost}(\Phi, \Sigma') \leq \mathsf{cost}(\Phi, \Sigma)$. □

4.2 The Implicative Hitting Set Case

We shall assume that each constraint of a k-IHS-B+ formula is either a disjunction of up to k variables, or an implication $u \to v$, where u and v are variables. We call a constraint of the former type a *set clause.* We define the implication graph $G(\phi)$ of such a formula ϕ to have one vertex for each variable of ϕ, and a directed edge $u \to v$ for each constraint $u \Rightarrow v$. Note that, in contrast to the implication graph of a bijunctive formula, the implication graph of a k-IHS-B+ formula does not contain any vertices for negative literals. The closure $T(v)$ of a vertex v is of $G(\phi)$ is the set of vertices reachable from v via a directed path; the vertex v is considered to be reachable from itself.

Before describing our strategy, we define a normal form for an implicative hitting set formula. We say that a quantified IHS-B+ formula ϕ is in *normal form,* if the following three conditions hold.

- There is no directed path from a universally quantified variable v_i to another universal variable v_j in the graph $G(\phi)$. Otherwise, the adversary could play $v_i = 1$, $v_j = 0$, hence falsifying at least one constraint on the path form v_i to v_j.
- When v_i is an existential variable, and v_j is an universal variable, such that v_1 comes before v_j in the quantification order, there is not a directed path from v_i to v_j, nor a directed path from v_j to v_i.
  To justify this assumption, suppose there is a path from v_i to v_j. Any winning existential strategy must set $v_i = 0$, since otherwise the universal player could falsify some constraint on the path by playing $v_j = 0$. Similarly, if there is a path from v_j to v_i the value of v_i is forced to 1. Hence we can replace v_i with the appropriate value and propagate the changes.
- The implication graph $G(\phi)$ is acyclic. If $G(\phi)$ is not acyclic, consider some strongly connected component C. Let v_i be the variable in C that occurs first in the quantification order. Note that the only way how to satisfy all constraints of C is to play $v_j = v_i$ for all variables $v_j \in C$ (note that all variables in C, with the possible exception of v_i, are by our previous assumptions existential). Hence we can replace, all occurences of v_j by v_i and set the weight of v_i to be $w(C)$.

Throughout, we assume IHS-B+ formulas to be in normal form. We have the following characterization of satisfiable IHS-B+ formulas.

Theorem 4. *Let ϕ be a IHS-B+ quantified formula in normal form. Let E be the set of all its existential variables v, such that $T(v)$ contains no universal variable. Then Φ is satisfiable if and only if each set clause of Φ contains a variable from E.*

Proof. If each set clause of Φ contains a variable from E, the existential player can win by playing $v \leftarrow 1$ for all $v \in E$, and $v \leftarrow 0$ for $v \notin E$. If the condition of the lemma is not met, there must be a set clause C that contains no variable in E. By setting all universal variables to 0, the universal player can force the existential player to either violate an implicative constraint, or set all variables outside E to 0, violating the clause C. □

The strategy. Our strategy, denoted by Σ_0, is constructed for a given formula Φ in a recursive fashion. In particular, we show how for a formula Φ to construct a simpler formula Φ', and, given a strategy for Φ', construct a strategy for the original formula Φ. The strategies we construct are ignorant: each existential variable has its value predetermined, and is set independently of the universal player's moves. Clearly, such a strategy can be described in space polynomial in the size of the original formula Φ. It can easily be verified that all of the reduction steps we will give can be performed in polynomial time.

The simplification process proceeds as follows. If Φ contains no set clauses, the optimal strategy for the universal player is to set all of the universal variables to 1. Our strategy in this case is to set all variables v such that $T(v)$ contains a universal variable to 1, and all other variables to 0.

If Φ contains a set clause, we do the following. Pick an arbitrary set clause $u_1 \vee u_2 \vee \ldots \vee u_l$, and decrease the weight of all sets $T(u_i)$ in such a way that the weight in each set is decreased by at least the amount $\delta = \min_i w(T(u_i))$, and the total decrease in the weight of all variables is no more than $l\delta$. (This can be done, for example, by considering the sets $T(u_i)$ in any order, and decreasing the weight of variables in each set until either the weight of the variables in the set is decreased by δ, or the weights of all variables in the set become zero.) Note that after the weight reduction, there must be a variable u_i such that $w(T(u_i)) = 0$. There are two cases: if $T(u_i)$ contains only existential variables, we set all variables in $T(u_i)$ to 1. Otherwise, $T(u_i)$ must contain a universal variable y such that $w(T(y)) = 0$. In this case we can assume that the universal player plays $y \leftarrow 0$. Hence, we set $y \leftarrow 0$, and set $u \leftarrow 0$ for all existential variables u such that $y \in T(u)$, including the existential variable u_i.

The analysis. As in the previous case, it is straightforward to verify that the described strategy Σ_0 is winning on satisfiable formulas. We establish the claimed approximation ratio in the following theorem.

Theorem 5. *Let Φ be a weighted k-IHS-B+ quantified formula, and assume that Φ is satisfiable. Then* $\mathsf{cost}(\Phi, \Sigma_0) \leq k \cdot \mathsf{cost}(\Phi)$.

Proof. We consider three cases.

- If Φ contains no set clauses, our strategy Σ_0 sets $v \leftarrow 1$ if $T(v)$ contains a universal variable and $v \leftarrow 0$ otherwise. We claim that $\mathsf{cost}(\Phi, \Sigma_0) = \mathsf{cost}(\Phi)$. This is because against an adversary that sets all universal variables to 1, every winning strategy must set to 1 all variables that Σ_0 sets to 1.
- If Φ contains a set clause $C = u_1 \vee \ldots \vee u_l$ such that $\delta = \min_i w(T(u_i)) > 0$, our algorithm constructs a new formula Φ', where the weights of the sets $T(u_i)$ have been decreased by δ. Since Φ' is a simpler formula, we can assume by induction that $\mathsf{cost}(\Phi', \Sigma_0) \leq k \cdot \mathsf{cost}(\Phi')$. Observe that $\mathsf{cost}(\Phi') + \delta \leq \mathsf{cost}(\Phi)$, because any winning existential strategy must ensure that one of the variables in C is set to 1. Observe also that $\mathsf{cost}(\Phi, \Sigma_0) \leq \mathsf{cost}(\Phi', \Sigma_0) + k \cdot \delta$, because the total decrease in weights from Φ to Φ' is no more than $k\delta$. From

these two facts it follows that

$$\mathsf{cost}(\Phi, \Sigma_0) \leq \mathsf{cost}(\Phi', \Sigma_0) + k \cdot \delta \leq k \cdot \mathsf{cost}(\Phi') + k \cdot \delta \leq k \cdot \mathsf{cost}(\Phi).$$

- If Φ contains a set clause $C = u_1 \vee \ldots \vee u_l$ such that $w(T(u_i)) = 0$ for some i, there are two subcases.
  If $T(u_i)$ does not contain any universal variables, our strategy Σ_0 sets all variables in $T(u_i)$ to 1. Let $\Phi' = \Phi[T(u_i) \leftarrow 1]$ be the simplified formula. From induction we get $\mathsf{cost}(\Phi', \Sigma_0) \leq k \cdot \mathsf{cost}(\Phi')$. By definition of Σ_0, it holds that $\mathsf{cost}(\Phi, \Sigma_0) = \mathsf{cost}(\Phi', \Sigma_0)$. It also holds that $\mathsf{cost}(\Phi') \leq \mathsf{cost}(\Phi)$. This is because any strategy Σ can be modified into one that sets all variables in $T(u_i)$ to 1, and has value less than or equal to the value of Σ. Hence, $\mathsf{cost}(\Phi, \Sigma_0) \leq k \cdot \mathsf{cost}(\Phi)$.
  On the other hand, if $T(u_i)$ contains an universal variable y, our strategy Σ_0 sets y and all variables v such that $y \in T(v)$ to 0. Again, let Φ' be the formula with these substitutions made. We have $\mathsf{cost}(\Phi, \Sigma_0) = \mathsf{cost}(\Phi', \Sigma_0)$ by definition of Σ_0. It is straightforward to verify that $\mathsf{cost}(\Phi') \leq \mathsf{cost}(\Phi)$, so we have $\mathsf{cost}(\Phi, \Sigma_0) = \mathsf{cost}(\Phi', \Sigma_0) \leq k \cdot \mathsf{cost}(\Phi') \leq k \cdot \mathsf{cost}(\Phi)$, as desired.

4.3 The Weakly Negative Case

The Strategy. In this case, the strategy Σ_0 is quite simple. As in the bijunctive case, as the game progresses and variables are set, Σ_0 instantiates the variables with the values played. When the formula has the form $\phi = \exists x \ldots$ and it is time for the existential player to set a variable x, the strategy simply checks to see if the formulas $\phi[x \leftarrow 0]$, $\phi[x \leftarrow 1]$ are satisfiable. If only one is, the strategy sets x accordingly; otherwise, the strategy simply sets x to 0. Note that determining satisfiability of a weakly negative quantified formula can be performed in polynomial time [7].

The analysis. We now demonstrate that the described strategy is optimal in the sense that it yields a 1-approximation.

Theorem 6. *Let Φ be a weighted weakly negative quantified formula, and assume that Φ is satisfiable. Then* $\mathsf{cost}(\Phi, \Sigma_0) \leq \mathsf{cost}(\Phi)$.

Proof. Without loss of generality, we focus on weakly negative quantified formulas of the form $\phi = \forall y_1 \exists x_1 \ldots \forall y_n \exists x_n \mathcal{C}$. Let $\phi[a_1, b_1, \ldots, a_k, b_k]$ denote the formula

$$\forall y_{k+1} \exists x_{k+1} \ldots \forall y_n \exists x_n \mathcal{C}[y_1 = a_1, x_1 = b_1, \ldots, y_k = a_k, x_k = b_k],$$

that is, the formula where the variables $y_1, x_1, \ldots, y_k, x_k$ are instantiated with the values $a_1, b_1, \ldots, a_k, b_k \in \{0, 1\}$. We make the observation that if the formulas $\varphi = \phi[a_1, b_1, \ldots, a_k, b_k]$ and $\varphi' = \phi[a'_1, b'_1, \ldots, a'_k, b'_k]$ are both satisfying, then $\varphi'' = \phi[a_1 \wedge a'_1, b_1 \wedge b'_1, \ldots, a_k \wedge a'_k, b_k \wedge b'_k]$ is also satisfying: given a winning

strategy $\Sigma = \{\sigma_i\}$ for φ and a winning strategy $\Sigma' = \{\sigma'_i\}$ for φ', it can be verified using [5, Lemma 4.8] that $\{\sigma_i \wedge \sigma'_i\}$ is a winning strategy for φ''.

Fix a quantified formula ϕ and let Σ' be an arbitrary winning strategy for ϕ. To demonstrate that Σ_0 is a 1-approximation, it suffices to show the following: for any ignorant strategy Π, and for all existentially quantified variables x in ϕ, it holds that $A_{(\phi,\Sigma_0,\Pi)}(x) \leq A_{(\phi,\Sigma',\Pi)}(x)$. (Here, $\leq$ denotes the usual total ordering on $\{0,1\}$ where $0 \leq 1$.) Suppose that this is not true, that is, there is an ignorant strategy Π and an existentially quantified variable x such that $A_{(\phi,\Sigma_0,\Pi)}(x) = 1$ and $A_{(\phi,\Sigma',\Pi)}(x) = 0$. Let x_i denote the first variable with this property, and define $a_j = A_{(\phi,\Sigma_0,\Pi)}(v)$ and $a'_j = A_{(\phi,\Sigma',\Pi)}(v)$ for $j \in [i-1]$. Moreover, let b_j denote the value given to y_j by Π (for $j \in [n]$). We have that the formulas $\phi[a_1, b_1, \ldots, b_{k-1}, a_k, 1]$ and $\phi[a_1, b'_1, \ldots, b'_{k-1}, a_k, 0]$ are satisfiable, from which it follows by the initial observation that $\phi[a_1, b_1, \ldots, b_{k-1}, a_k, 0]$ is also satisfiable, contradicting the definition of Σ_0. □

5 Inapproximability

In this section, we establish the inapproximability of the remaining constraint languages. Note that the class of width-2 affine constraints is contained in the class of bijunctive constraints.

Theorem 7. *If S is a constraint language that is not weakly negative, bijunctive, nor k-IHS-B+, then MIN-QCSP(S) is not constant factor approximable, unless NP has deterministic algorithms running in time $n^{\log^{O(1)} n}$.*

Proof. As in [3], we define $[S]$ to be the class of all relations definable from S using relations from $S \cup \{=\}$ (where = is the equality relation on $\{0, 1\}$), conjunction, existential quantification, and universal quantification; we refer the reader to [3] for more details. Note that if S is weakly negative (bijunctive, k-IHS-B+), then $[S]$ is also weakly negative (respectively bijunctive, k-IHS-B+); see [5, Lemma 4.5] for a part of the proof.

As in [3, Proposition 1], it can be demonstrated that for two constraint languages S_1, S_2, if $[S_1] \subseteq [S_2]$, then MIN-QCSP(S_1) is polynomial-time reducible to MIN-QCSP(S_2). From this, it follows that MIN-QCSP(S) and MIN-QCSP($[S]$) are polynomial-time reducible to each other.

Let S be a constraint language satisfying the hypothesis of the theorem. Then, $[S]$ is not weakly negative, bijunctive, nor k-IHS-B+. We make use of the Pol and Inv operators as defined (for instance) in [4], as well as the description and notation of Post's lattice in [2]. It can be verified that either $\mathsf{Pol}([S]) \subseteq S_0$, $\mathsf{Pol}([S]) \subseteq V$, or $\mathsf{Pol}([S]) \subseteq L$. Let MIN-CSP($S$) denote the restriction of the problem MIN-QCSP(S) to instances where there is no universal quantification.[2] We break into three cases depending on which of these containments holds, and show that in any of these cases, MIN-CSP(T) can be reduced to MIN-QCSP(S) for some T that is not weakly negative, bijunctive, k-IHS-B+, nor 0-valid.

[2] Note that MIN-CSP(S) is denoted by MIN-ONES(S) in [8].

- If $\mathsf{Pol}([S]) \subseteq S_0$, then $[S]$ is not 0-valid. We have that MIN-CSP($[S]$) reduces to MIN-QCSP(S).
- If $\mathsf{Pol}([S]) \subseteq V$, then the relation $R = \{0,1\}^3 \setminus \{(0,0,1\}$ is in $[S]$. It follows by the definition of $[S]$ that the relation $R' = \{0,1\}^2 \setminus \{(0,0)\}$ is also in $[S]$, implying that $[S]$ is not 0-valid. We have that MIN-CSP($[S]$) reduces to MIN-QCSP(S).
- If $\mathsf{Pol}([S]) \subseteq L$, then we assume that $[S]$ is 0-valid (for if it is not, we can simply use the reduction from MIN-CSP($[S]$) to MIN-QCSP(S)). We claim that MIN-CSP($[S] \cup \{N\}$) reduces to MIN-QCSP($[S]$), where $N = \{(1)\}$. We first observe that $M = \{(0,0),(1,1)\} \in [S]$. Given an instance of MIN-CSP($[S] \cup \{N\}$), we create an instance of MIN-QCSP($[S]$) having the same cost by introducing a single universally quantified variable y (which comes before all of the existential variables) and converting each constraint $N(v)$ over N to the constraint $M(y,v)$. Thus MIN-CSP($[S] \cup \{N\}$) reduces to MIN-QCSP($[S]$), which in turn reduces to MIN-QCSP(S).

From the classification theorem for MIN-CSP along with the associated hardness of approximation results in [8], it follows that MIN-CSP(T), and hence MIN-QCSP(S), is not constant factor approximable, unless NP has deterministic algorithms running in time $n^{\log^{O(1)} n}$. □

References

1. B. Aspvall, M.F. Plass, R.E. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters,* vol 8 no. 3, 1979.
2. E. Böhler, N. Creignou, S. Reith and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM SIGACT-Newsletter,* 34(4):38-52, 2003.
3. Ferdinand Börner, Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Quantified Constraints: Algorithms and Complexity. *Computer Science Logic,* 2003.
4. Andrei Bulatov, Andrei Krokhin, Peter Jeavons. Constraint satisfaction problems and finite algebras. *Proceedings of ICALP'00,* Lecture Notes in Computer Science 1853, 272-282.
5. Nadia Creignou, Sanjeev Khanna, Madhu Sudan. Complexity Classifications of Boolean Constraint Satisfaction Problems. *SIAM Monographs on Discrete Mathematics and Applications,* 2001.
6. D. Gusfield, L. Pitt. A Bounded Approximation for the Minimum Cost 2-Sat Problem. *Algorithmica,* 8:103–117, 1992.
7. Marek Karpinski, Hans Kleine Büning, Peter H. Schmitt. On the Computational Complexity of Quantified Horn Clauses. *Computer Science Logic,* 1987, 129-137.
8. S. Khanna, M. Sudan, L. Trevisan and D.P. Williamson. The Approximability of Constraint Satisfaction Problems. *SIAM Journal on Computing,* 30:1863–1920, 2001.
9. Thomas J. Schaefer. The complexity of satisfiability problems. *Proceedings of the Tenth ACM Symposium on Theory of Computing,* pp. 216–226, 1978.

Approximating Boolean Functions by OBDDs

Andre Gronemeier*

Lehrstuhl Informatik 2
Universität Dortmund, Germany

Abstract. In learning theory and genetic programming, OBDDs are used to represent approximations of Boolean functions. This motivates the investigation of the OBDD complexity of approximating Boolean functions with respect to given distributions on the inputs. We present a new type of reduction for one–round communication problems that is suitable for approximations. Using this new type of reduction, we prove the following results on OBDD approximations of Boolean functions:

1. We show that OBDDs approximating the well–known hidden weighted bit function for uniformly distributed inputs with constant error have size $2^{\Theta(n^{1/4})}$, improving a previously known result.
2. We prove that for every variable order π the approximation of some output bits of integer multiplication with constant error requires π-OBDDs of exponential size.

1 Introduction

Branching programs (BPs), also called binary decision diagrams (BDDs), are both a theoretical model for nonuniform sequential computation and a data structure for Boolean functions in applications like symbolic verification and other CAD problems. Especially restricted BP types like OBDDs have good algorithmic properties [1] and proof methods for strong lower bounds on the size of restricted BPs for concrete functions have been developed [2].

Definition 1. *A deterministic branching program (BP) or binary decision diagram (BDD) on the variable set $X = \{x_1, \ldots, x_n\}$ is a directed acyclic graph with one source and two sinks. The sinks are labeled by the constants 0 and 1, respectively, interior nodes are labeled by variables from X and have two outgoing edges labeled by the constants 0 and 1. The BP G computes a function $G : \{0,1\}^n \longrightarrow \{0,1\}$ defined on X in the following way: For an input $a \in \{0,1\}^n$ the output $G(a)$ is defined as the label of the sink which is reached from the source of the graph by following the edge labeled by a_i for nodes labeled by x_i. The size $|G|$ of a BP G is the number of its nodes. Let π be a permutation on the set $\{1, \ldots, n\}$. A π-OBDD is a BP where $\pi(i) < \pi(j)$ for each edge leading from a node labeled by x_i to a node labeled by x_j. In this context π is called a variable order. A π-OBDD for some unspecified variable order is simply called OBDD.*

* Supported by DFG grant SA 1053/1-1

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 251–262, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In some applications, e.g. learning theory and genetic programming, OBDDs are used to represent approximations of Boolean functions [3]. Motivated by these applications, Krause, Savický and Wegener [4] started investigating lower bounds on the size of OBDDs approximating Boolean functions.

Definition 2. *Let μ be a probability distribution on $\{0,1\}^n$. A π-OBDD approximates f with error ε with respect to μ if* $\mathrm{Prob}_\mu(G(x) \neq f(x)) \leq \varepsilon$. *The π-OBDD complexity $\pi\text{-}OBDD_\varepsilon(f_\mu)$ of approximating f with error ε with respect to μ is the size of a smallest π-OBDD which approximates f with error ε with respect to μ. The OBDD complexity $OBDD_\varepsilon(f_\mu)$ of approximating f with error ε with respect to μ is $\min_\pi\{\pi\text{-}OBDD_\varepsilon(f_\mu)\}$. In the above notation μ is omitted if it describes the uniform distribution.*

Although approximations of Boolean functions have been studied to prove lower bounds on the size of randomized OBDDs by Yao's min–max principle [5], these results only show lower bounds for very specific input distributions. There are few results on the OBDD complexity of approximations for given input distributions, especially important distributions like the uniform distribution [4,6,7,8]. In [7] Bollig, Sauerhoff and Wegener ask how the known lower bound techniques for the exact case can be adopted to work also for approximations. In this paper, we define an appropriate type of reduction for approximation problems (Sec. 3). Then these reductions are used to obtain new lower bounds on the size of OBDDs approximating the hidden weighted bit function and integer multiplication with respect to the uniform distribution (Sec. 4).

2 Survey of the Results

Bryant's hidden weighted bit function [9] is a well–known benchmark function in the BP literature.

Definition 3. *For a vector $x \in \{0,1\}^n$ let $\|x\|$ denote the number of ones in x. On the input $x = (x_1, \ldots, x_n)$ the hidden weighted bit function* $\mathrm{HWB}_n : \{0,1\}^n \longrightarrow \{0,1\}$ *is defined by* $\mathrm{HWB}(x) := x_{\|x\|}$ *where $x_0 := 0$.*

While HWB_n is simple for many restricted BP types only slightly more general than OBDDs, e.g. FBDDs and k-OBDDs [10], nonetheless OBDDs computing HWB_n have size exponential in n [9]. So the HWB–function exposes a specific weakness of OBDDs. Bollig, Sauerhoff and Wegener [7] have shown that even approximations of HWB_n with constant error $\varepsilon \in]0, \frac{1}{2}[$ with respect to the uniform distribution require OBDDs of size at least $2^{\Omega(n^{1/6-\delta})}$ for arbitrary constants $\delta > 0$. Here this lower bound is improved and a matching upper bound is shown.

Theorem 1. *For every constant $\varepsilon \in]0, \frac{1}{2}[$ the OBDD complexity of approximating HWB_n with error ε with respect to the uniform distribution is $2^{\Theta(n^{1/4})}$.*

Multiplication is one of the basic arithmetic functions. Naturally, the BP-complexity of multiplication has been investigated.

Definition 4. *For a vector* $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$ *let* $(x)_2$ *denote the interpretation of* x *as a binary number. Then* $\mathrm{MUL}_{i,n} : \{0,1\}^n \times \{0,1\}^n \longrightarrow \{0,1\}$ *is the function that maps the inputs* $x, y \in \{0,1\}^n$ *to the* i*-th bit of the binary representation of* $(x)_2 \cdot (y)_2$.

In his pioneering paper [1] Bryant was the first to investigate the OBDD complexity of integer multiplication. Later he proved that the computation of the middle bit of integer multiplication requires OBDDs of exponential size [9]. Since then this bound has been improved [11] and exponential lower bounds on the size of various BP types computing the middle bit of integer multiplication have been shown, e.g. for randomized OBDDs [12], FBDDs [13,14] and read-k BPs and linear length multiway BPs [8]. Surprisingly, approximating the middle bit of integer multiplication with respect to the uniform distribution is easy even for OBDDs [8]. Complementing this result, we show that no variable order is suitable for the approximation of all output bits: For each variable order there is an output bit which requires exponential OBDD size.

Theorem 2. *Let* $\varepsilon \in]0, \frac{1}{2}[$ *be a constant. For each sequence* π_n *of variable orders there exists a sequence* $i_n \in \{0, \ldots, n-1\}$ *of output bits of integer multiplication with* $\pi_n\text{-}OBDD_\varepsilon(\mathrm{MUL}_{i_n,n}) = 2^{\Omega(n)}$.

These results are proved using a new type of reduction for 1–round communication problems which is described in the following section. The proofs are generalizations of proofs for analogous theorems for exact computations. So we think that these reductions answer, at least partially, the aforementioned question raised in [7].

3 Proof Methods

3.1 Communication Complexity

We will use Yao's two–player communication complexity [15] to prove lower bounds on the OBDD-complexity of Boolean functions. A thorough introduction to communication complexity can be found in [16]. Here we are only interested in 1–round protocols. For the definition of reductions we extend the classical definition of randomized public coin 1–round protocols [16] by an oracle–function.

Definition 5. *Let* X, Y, X', Y', M *and* R *be finite sets,* $g : X' \times Y' \longrightarrow Z'$ *be a function and* ρ *be a probability distribution on the set* R. *A randomized 1–round communication protocol* $P[g]$ *with oracle* g *is a communication game between two players Alice and Bob. For an input* $(x, y) \in X \times Y$ *and a random input* $r \in R$, *which is chosen with respect to distribution* ρ, *the output* $P[g](x, y, r)$ *of the protocol is computed according to the following rules:*

The input (x, y) *is distributed among Alice and Bob. Alice gets the private input* x *and Bob gets the private input* y. *Both players have access to the public random input* r. *Then the following computation and communication steps are performed:*

1. *Alice computes a message* $m := P_A(x,r) \in M$ *and sends* m *to Bob.*
2. *Alice computes her oracle input* $q_A := Q_A(x,r) \in X'$.
3. *Bob computes his oracle input* $q_B := Q_B(m,y,r) \in Y'$.
4. *Alice and Bob query the oracle* g *for the input* (q_A, q_B). *Bob gets the oracle's output* $z' := g(q_A, q_B) \in Z'$.
5. *Bob computes the output* $P[g](x,y,r) := P_B(m,y,r,z') \in Z$.

The protocol $P[g]$ *computes a function* $f : X \times Y \longrightarrow Z$ *with error* ε *if* $\mathrm{Prob}_\rho(P[g](x,y,r) \neq f(x,y)) \leq \varepsilon$ *for all inputs* $(x,y) \in X \times Y$. *The cost of the protocol is* $c(P[g]) := \lceil \log_2 |M| \rceil$. *If the output of the protocol does not depend on the oracle, then the corresponding part of the notation is omitted. A deterministic 1–round communication protocol is a randomized protocol, as defined above, where the output of the protocol does not depend on the random input* r.

Given a probability distribution on the input set $X \times Y$ of a communication problem $f : X \times Y \longrightarrow Z$, one can define approximations of functions by deterministic 1–round communication protocols. This is done analogously to approximations by OBDDs. Note that the cost of a protocol for a given Boolean function may strongly depend on how the input is distributed among the players.

Definition 6. *Let* f *be a Boolean function on the variable set* $X = \{x_1, \ldots, x_n\}$. *Each partition* Π *of the set* X *into two sets* X_A *and* X_B *defines a corresponding communication problem* Π-f *where Alice receives the variables from* X_A *while Bob receives the variables from* X_B. *Given a probability distribution* μ *on the inputs of* f, *let* $D_\varepsilon^{A \to B}(\Pi\text{-}f_\mu)$ *denote the cost of a cheapest deterministic 1–round communication protocol* P *that approximates* Π-f *with error* ε *with respect to* μ.

Communication complexity has been used by many authors to prove lower bounds on the size of BPs computing Boolean functions (see [2]). Bollig, Sauerhoff and Wegener [7] observed that the same proof method can be applied to approximations of Boolean functions by BPs.

Theorem 3. *Let* $f : \{0,1\}^n \longrightarrow \{0,1\}$ *be a Boolean function on the variable set* $X = \{x_1, \ldots, x_n\}$ *and let* μ *be a probability distribution on the inputs of* f. *Further let* $\Pi = (X_A, X_B)$ *be a partition of* X *into two sets and let* π *be a variable order on* X *that satisfies* $\pi(a) < \pi(b)$ *for all* $x_a \in X_A$ *and* $x_b \in X_B$. *Then* $\pi\text{-}OBDD_\varepsilon(f_\mu) \geq 2^{D_\varepsilon^{A \to B}(\Pi\text{-}f_\mu)}$.

A partition $\Pi = (X_A, X_B)$ *of* X *is called* α*–balanced if* $|X_A| = \lfloor \alpha n \rfloor$. *Then* $OBDD_\varepsilon(f_\mu) \geq \min\{2^{D_\varepsilon^{A \to B}(\Pi\text{-}f_\mu)} \mid \Pi \text{ is } \alpha\text{–balanced}\}$ *for any* $\alpha \in [0,1]$.

Since a similar theorem is proved in [7], a proof of Theorem 3 is not necessary.

3.2 Randomized Rectangular Reductions

In communication complexity theory the relative complexity of problems is usually investigated with *rectangular reductions* [17]: A rectangular reduction from a problem $f : X \times Y \longrightarrow Z$ to a problem $g : X' \times Y' \longrightarrow Z'$ is a pair of functions $\rho_X : X \longrightarrow X'$ and $\rho_Y : Y \longrightarrow Y'$ with the property $g(\rho_X(x), \rho_Y(y)) = f(x,y)$.

If f is reducible to g in this way, then lower bounds on the communication complexity of f imply lower bounds on the communication complexity of g. Unfortunately, this method does not work well for approximations of Boolean functions with respect to arbitrary input distributions: For a fixed reduction the input distribution on $X \times Y$ uniquely defines the input distribution on $X' \times Y'$. This complicates the proof of lower bounds for approximations of g with respect to given distributions on $X' \times Y'$. Here we try to solve this problem by randomizing the reduction and by allowing additional communication.

Definition 7. *Let $f_\mu : X \times Y \longrightarrow Z$ and $g_{\mu'} : X' \times Y' \longrightarrow Z'$ be functions with probability distributions μ and μ' on their finite input sets $X \times Y$ and $X' \times Y'$. The function f_μ is $\mathrm{RD}(\varepsilon, k)$–reducible to $g_{\mu'}$, written $f_\mu \leq_{\mathrm{RD}}^{\varepsilon,k} g_{\mu'}$, if there is a randomized 1–round communication protocol $P[g]$ with oracle g which has the following properties: Let the oracle inputs q_A and q_B be defined as in Definition 5. Then*

1. $c(P[g]) \leq k$,
2. $\mathrm{Prob}_{\mu,\rho}(\ (q_A, q_B) = (x', y')\ ) = \mu'(x', y')$ *for every* $(x', y') \in X' \times Y'$ *and*
3. $\mathrm{Prob}_{\mu,\rho}(\ P[g](x, y, r) \neq f(x, y)\ ) \leq \varepsilon$.

Note that the R in RD–reduction is an abbreviation for 'rectangular' while D stands for 'distributional'. The purpose of the extensions in Definition 7 compared to the simpler rectangular reductions is to control the probability distribution on the oracle inputs. The following theorem shows how $\mathrm{RD}(\varepsilon, k)$–reductions can be used to prove lower bounds on the 1–round communication complexity of approximations.

Theorem 4. *Let $f_\mu : X \times Y \longrightarrow Z$ and $g_{\mu'} : X' \times Y' \longrightarrow Z'$ be functions with probability distributions μ and μ' on their finite input sets. If $f_\mu \leq_{\mathrm{RD}}^{\varepsilon,k} g_{\mu'}$, then $D_{\varepsilon+\varepsilon'}^{A \to B}(f_\mu) \leq D_{\varepsilon'}^{A \to B}(g_{\mu'}) + k$ for any constant $\varepsilon' \in]0, \frac{1}{2}[$.*

Proof. Let $F[g]$ denote the randomized 1–round protocol with oracle g that proves $f_\mu \leq_{\mathrm{RD}}^{\varepsilon,k} g_{\mu'}$ according to Definition 7. By the definition of $\mathrm{RD}(\varepsilon, k)$–reductions, the cost of $F[g]$ is bounded by k. For every constant $\varepsilon' \in]0, \frac{1}{2}[$ there exists a deterministic 1–round protocol G which approximates g with error ε' with respect to μ' whose cost is bounded by $D_{\varepsilon'}^{A \to B}(g_{\mu'})$. Then we can obtain a deterministic 1–round protocol P that approximates f with error $\varepsilon + \varepsilon'$ with respect to μ in the following way: The query of the g–oracle in $F[g]$ is replaced by the execution of the protocol G. The resulting protocol P_{rand} is a randomized 1–round protocol that approximates f. By construction, the cost of P_{rand} is bounded by $D_{\varepsilon'}^{A \to B}(g_{\mu'}) + k$. The output $P_{\mathrm{rand}}(x, y, r)$ of the constructed protocol can be different from $f(x, y)$ because of the following reasons:

(1) The protocol $F[g]$ computes the right output, but $P_{\mathrm{rand}}(x, y, r)$ is different from $F[g](x, y, r)$.
(2) Even the protocol $F[g]$ computes the wrong result for inputs x, y and r.

By Definition 7, the probability of (2) is bounded by ε. By the definition of the protocol G and by Definition 7, the probability of (1) is bounded by ε' because in this case the output of G differs from the oracle output. In all, the approximation error of protocol P_{rand} is $\text{Prob}_{\mu,\rho}(P_{\text{rand}}(x,y,r) \neq f(x,y)) \leq \varepsilon + \varepsilon'$. Then a well–known averaging argument [5] shows that for some fixed $r^* \in R$ $\text{Prob}_{\mu}(P_{\text{rand}}(x,y,r^*) \neq f(x,y)) \leq \varepsilon + \varepsilon'$. Replacing the random input r in protocol P_{rand} by the constant r^* yields the desired deterministic protocol P that approximates f with error $\varepsilon + \varepsilon'$. □

4 Proofs of the Main Results

Now we will prove the results from Section 2 by RD-reductions from the so–called index function for different distributions on the inputs.

Definition 8. *For inputs* $(x_1,\ldots,x_n) \in \{0,1\}^n$ *and* $i \in \{1,\ldots,n\}$ *the index function* IND_n *is defined by* $\text{IND}_n(x,i) := x_i$.

The proofs are essentially randomized versions of proofs of similar theorems for exact computations. This underlines our claim that randomized reductions can be used to adopt the proof methods for exact computations to approximations.

4.1 Hidden Weighted Bit Function

Bollig, Sauerhoff and Wegener [7] observed that if Alice owns many variables with index close to $\frac{n}{2}$, then the hidden weighted bit function HWB_n for uniformly distributed inputs is very similar to the index function for uniformly distributed values of the data variables x and binomially distributed values of the index i.

Definition 9. *Let* $\text{bin}(n)$ *denote the product distribution of the uniform distribution on the set* $\{0,1\}^n$ *and the binomial distribution with parameters* $n-1$ *and* $\frac{1}{2}$. *Then for the inputs* $x = (x_0,\ldots,x_{n-1}) \in \{0,1\}^n$ *and* $i \in \{0,\ldots,n-1\}$ *the index function with respect to* $\text{bin}(n)$ *is defined by* $\text{IND}_{\text{bin}(n)}(x,i) := x_i$.

Compared to Definition 8, here the numbering of the indices is adjusted to the usual definition of the binomial distribution. Bollig *et al.* [7] also proved that $D_{\varepsilon}^{\text{A}\to\text{B}}(\text{IND}_{\text{bin}(n+1)}) = \Omega(n^{\frac{1}{2}-\delta})$ for arbitrary constants $\delta > 0$. We will use a slightly improved version of this result.

Theorem 5. $D_{\varepsilon}^{\text{A}\to\text{B}}(\text{IND}_{\text{bin}(n+1)}) = \Theta(n^{\frac{1}{2}})$ *for every constant* $\varepsilon \in]0,\frac{1}{2}[$.

To prove this result, the proof from [7] only needs to be modified in a few places. Details are given in the Appendix. Now Theorem 5 and the similarity of HWB_n and $\text{IND}_{\text{bin}(n)}$ stated above can be used to prove Theorem 1 by a randomized reduction.

Proof (Theorem 1). We claim that for every constant $\varepsilon' \in]0,\frac{1}{2}[$ there are constants $\alpha \in]0,1[$, $c_{\varepsilon'} \in]0,\infty[$ and $n_{\varepsilon'} \in \mathbb{N}$ such that for all $n \geq n_{\varepsilon'}$ and all α–balanced partitions Π of the input variables of HWB_n

$$\text{IND}_{\text{bin}(\lfloor c_{\varepsilon'} n^{1/2} \rfloor)} \leq_{\text{RD}}^{\varepsilon',0} \Pi\text{-HWB}_n \ . \qquad (*)$$

Then, by Theorem 4, $D_{\varepsilon}^{A\to B}(\Pi\text{-HWB}_n) \geq D_{\varepsilon+\varepsilon'}^{A\to B}(\text{IND}_{\text{bin}(\lfloor c_{\varepsilon'} n^{1/2} \rfloor)})$. Since $\varepsilon + \varepsilon' < \frac{1}{2}$ for $\varepsilon' := \frac{1}{2}(\frac{1}{2} - \varepsilon)$, the lower bound from Theorem 1 is implied by Theorem 5. We still have to show that $(*)$ holds.

Let $n' := \lfloor c_{\varepsilon'} n^{1/2} \rfloor$ for the constant $c_{\varepsilon'}$ which will be fixed later on. For the proof of $(*)$ we consider α–balanced partitions $\Pi = (X_A, X_B)$ of the input variables $X = \{x_1, \ldots, x_n\}$ of HWB_n for constants α where $|X_A| = n - n' + 1$ and $|X_B| = n' - 1$. We use the following randomized $\Pi\text{-HWB}_n$–oracle protocol to show that $\text{IND}_{\text{bin}(n')}$ is $\text{RD}(\varepsilon', 0)$–reducible to $\Pi\text{-HWB}_n$:

Let $(x', i) \in \{0,1\}^{n'} \times \{0, \ldots, n'-1\}$ be the input of $\text{IND}_{\text{bin}(n')}$. The random inputs of the protocol are a random vector $r \in \{0,1\}^{n-2n'+1}$ and a random permutation π on the set $\{1, \ldots, n'-1\}$. Both random inputs are chosen with respect to the uniform distribution. The output of the protocol is computed according to the following rules:

1. Alice computes $i_0 := \|x'\| + \|r\|$ and assigns the following values to the input variables X_A of her oracle input:
   - For $x_k \in X_A$ with $k = i_0 + j$ and $j \in \{0, \ldots, n'-1\}$ Alice sets $x_k := x'_j$.
   - The remaining unassigned variables from X_A are assigned values from the input variables in x' and the random input r in an arbitrary but fixed order, such that each of the input bits from x' and r is used exactly once.
2. Bob computes $b = (b_1, \ldots, b_{n'-1}) := \pi\left(\sum_{j=1}^{i} e_j\right)$ where e_j denotes the j-th unit vector and $\pi(v)$ denotes the permutation of the components of a vector v with respect to the permutation π. Then Bob assigns the values b_j, $j = 1, \ldots, n'-1$ to the variables X_B of his oracle input with respect to an arbitrary but fixed one–to–one mapping.
3. Alice and Bob query the $\Pi\text{-HWB}_n$–oracle. Bob uses the result z as the output of the protocol.

Obviously, the cost of the protocol is 0 since no communication takes place. To prove $(*)$ we have to show that the oracle inputs are uniformly distributed and that the approximation error of the protocol is bounded by ε': By construction, the mapping from inputs x' and r to Alice's oracle inputs is a bijection. So Alice's oracle inputs are uniformly distributed because the inputs x' and r are uniformly distributed. A simple calculation shows that Bob's oracle inputs are uniformly distributed: Let $v := \sum_{j=1}^{i} e_j$ for the input i of the index function. By the distribution of the input i, the probability of v having exactly i bits with value 1 is $\binom{n'-1}{i} 2^{-(n'-1)}$. The random permutation π maps v to a fixed vector $b \in \{0,1\}^{n'-1}$ that satisfies $\|b\| = i$ with probability $\binom{n'-1}{i}^{-1}$. Multiplying both probabilities yields the uniform distribution.

Let x be the oracle input of the protocol for input (x', i) and random inputs r and π. The output of the protocol is $z = x_{\|x\|} = x_{\|x'\|+\|r\|+\|b\|} = x_{i_0+i}$. By the assignment of the variables from X_A in the first step of the protocol, the output of the protocol is correct if $x_{i_0+i} \in X_A$. Thus an error of the protocol implies $x_{i_0+i} \in X_B$. We use the following estimates to bound the probability of this event (see [18]): For any $k \in \{0, \ldots, n\}$

$$\binom{n}{k}2^{-n} \leq \binom{n}{\lfloor \frac{n}{2} \rfloor}2^{-n} = \left(\frac{\pi}{2}n\right)^{-\frac{1}{2}}\left(1+O\left(\frac{1}{n}\right)\right) .$$

The random variable $i_0 + i$ is distributed with respect to the binomial distribution with parameters n and $\frac{1}{2}$ since the random vectors x', r and b are uniformly distributed. Then, by the above inequalities and by the choice of $|X_B|$, we get

$$\begin{aligned}\mathrm{Prob}(x_{i_0+i} \in X_B) &= \sum_{x_j \in X_B} \mathrm{Prob}(i_0 + i = j) \\ &\leq |X_B|\left(\frac{\pi}{2}n\right)^{-\frac{1}{2}}\left(1+O\left(\frac{1}{n}\right)\right) \\ &\leq c_{\varepsilon'}\left(\frac{\pi}{2}\right)^{-\frac{1}{2}}\left(1+O\left(\frac{1}{n}\right)\right) .\end{aligned}$$

If the constant $c_{\varepsilon'}$ is chosen sufficiently small to satisfy $c_{\varepsilon'}\left(\frac{\pi}{2}\right)^{-\frac{1}{2}} < \varepsilon'$, then $\mathrm{Prob}(x_{i_0+i} \in X_B \leq \varepsilon')$ for sufficiently large n. Fixing $c_{\varepsilon'}$ and $n_{\varepsilon'}$ to appropriate values completes the proof of $(*)$. The proof of the matching upper bound is contained in the Appendix. □

4.2 Integer Multiplication

We will prove Theorem 2 by a randomized reduction from the index function for uniformly distributed inputs to integer multiplication. Approximations of the index function for uniformly distributed inputs were studied by Kremer, Nisan and Ron [19] and Krause, Savický and Wegener [4]. In [7], the index function is used to prove lower bounds on the OBDD complexity of approximations. The following theorem is implicitly contained in [7].

Theorem 6. *The 1–round communication complexity of approximating* IND_n *with respect to the uniform distribution is* $D_\varepsilon^{A\to B}(\mathrm{IND}_n) = \Theta(n)$ *for every constant* $\varepsilon \in]0, \frac{1}{2}[$.

To simplify the proof of Theorem 2, we first define some notation to translate between integers and the corresponding binary representation.

Definition 10. *Given a nonnegative integer* x, *let* $x_{[i]}$ *denote the* i*-th bit of the binary representation of* x *and let* $x_{[i,j]}$ *denote the vector* $(x_{[i]}, x_{[i-1]}, \ldots, x_{[j]})$. *As usual, bits are counted from the least significant bit starting with 0.*

In the proof of Theorem 2 we will need a lemma about the distribution of the first n output bits of the product $x \cdot y$ when the inputs x and y are chosen independently and uniformly at random from $\mathbb{Z}_{2^n}$.

Lemma 1. *Let* $\mathbb{Z}_{2^n} := \{0, 1, 2, \ldots, 2^n - 1\}$ *and let* $i, m \in \{0, \ldots, n-1\}$ *with* $i \geq 2m$. *If* x *and* y *are chosen independently and uniformly at random from* $\mathbb{Z}_{2^n}$, *then* $\mathrm{Prob}((x \cdot y)_{[i,i-m+1]} = a) \leq 2^{-m+1}$ *for every* $a \in \{0,1\}^m$.

A similar result was shown by Dietzfelbinger *et al.* [20] in the analysis of a simple class of multiplicative universal hash functions. The proof of this lemma is contained in the Appendix. The following combinatorial lemma, which will be useful in the proof of Theorem 2, is also proved in the Appendix.

Lemma 2. *Given sets* $X = Y = \{0, \ldots, n-1\}$ *and subsets* $X_A \subseteq X$, $Y_A \subseteq Y$ *with* $|X_A| + |Y_A| = n$, *let* $X_B := X \setminus X_A$ *and* $Y_B := Y \setminus Y_A$. *Then for some* $i \in \{\lceil c \cdot n \rceil - 1, \ldots, n-1\}$ *there is a subset* $P \subseteq X_A \times Y_B$ *or* $P \subseteq Y_A \times X_B$ *with* $a + b = i$ *for all* $(a, b) \in P$ *and* $|P| \geq c \cdot n$ *where* $c = 17 - \sqrt{17^2 - 1}$.

Now we can prove Theorem 2. Note that this proof is essentially a randomized version of Bryant's first result [1] on the OBDD complexity of multiplication. By randomized reductions we are able to extend Bryant's proof method to approximation problems.

Proof (Theorem 2). For notational convenience we sometimes do not distinguish vectors $x \in \{0,1\}^n$ from the corresponding integers $(x)_2 \in \mathbb{N}$. In both cases we simply write x. The meaning of the variables should be evident from the operators applied to the variables.

We claim that for arbitrary constants $\varepsilon' \in]0, \frac{1}{2}[$ the following holds for constants $c_{\varepsilon'} \in]0,1[$ and $n_{\varepsilon'} \in \mathbb{N}$: For every $\frac{1}{2}$–balanced partition Π of the input variables of $\mathrm{MUL}_{i,n}$ with $n \geq n_{\varepsilon'}$ there exists an output bit $i \in \{0, \ldots, n-1\}$ with

$$\mathrm{IND}_{\lfloor c_{\varepsilon'} n \rfloor} \leq_{\mathrm{RD}}^{\varepsilon', O(1)} \Pi\text{-}\mathrm{MUL}_{i,n} \quad . \qquad (*)$$

Then, by Theorem 4, $D_{\varepsilon}^{\mathrm{A}\to\mathrm{B}}(\Pi\text{-}\mathrm{MUL}_{i,n}) \geq D_{\varepsilon+\varepsilon'}^{\mathrm{A}\to\mathrm{B}}(\mathrm{IND}_{\lfloor c_{\varepsilon'} n \rfloor}) - O(1)$. Since $\varepsilon + \varepsilon' < \frac{1}{2}$ for $\varepsilon' := \frac{1}{2}(\frac{1}{2} - \varepsilon)$, the lower bound from Theorem 2 is implied by Theorem 6. We still have to show that $(*)$ holds for every constant $\varepsilon' \in]0, \frac{1}{2}[$.

For $X = \{x_0, \ldots, x_{n-1}\}$ and $Y = \{y_0, \ldots, y_{n-1}\}$ let $X \cup Y$ be the set of the input variables of $\mathrm{MUL}_{i,n}$. Given the $\frac{1}{2}$–balanced partition Π of $X \cup Y$, let X_A and Y_A denote the set of Alice's input variables from X and Y, respectively. Let X_B and Y_B be analogously defined for Bob's input variables. By Lemma 2, there exists a subset $P \subseteq X_A \times Y_B$ (or $P \subseteq Y_A \times X_B$) where $|P| \geq (17 - \sqrt{17^2 - 1})n$ and $k + l = i^*$ for some $i^* \in \{0, \ldots, n-1\}$ and all $(x_k, y_l) \in P$ (or $(y_k, x_l) \in P$). Let the output bit i in $(*)$ be defined by $i := i^*$. W.l.o.g. assume $P \subseteq X_A \times Y_B$ and restrict P to a subset $P' \subseteq P$ in the following way: Let $m_{\varepsilon'} \in \mathbb{N}$ be a constant which will be fixed later on. If $(x_k, y_l) \in P'$, then $k \geq 2m_{\varepsilon'}$ and for all $j \in \{1, \ldots, 2m_{\varepsilon'}\}$ no pair in P' contains the variable x_{k-j}. Obviously, this restriction can be satisfied by sets P' of size $|P'| \geq \left\lfloor \frac{|P|}{2m_{\varepsilon'}+1} \right\rfloor \geq \left\lfloor \frac{1}{2m_{\varepsilon'}+1}(17 - \sqrt{17^2-1})n \right\rfloor$. We fix the constant $c_{\varepsilon'}$ to $\frac{1}{2m_{\varepsilon'}+1}(17 - \sqrt{17^2-1})$. Let X'_A be the set of variables from X_A which are contained in pairs from P' and let Y'_B be the set of variables from Y_B which are contained in pairs from P'. Let $n' := |P'|$. Then, by construction, $|X'_A| = |Y'_B| = n'$. Define the function $\lambda : \{1, \ldots, n'\} \longrightarrow \{0, \ldots, n-1\}$ that maps the integers from $\{1, \ldots, n'\}$ to the indices of the variables from X'_A in ascending order. If all input variables from $Y \setminus Y'_B$ have value 0, then Bob can choose $x_{\lambda(k)}$ as the output of $\mathrm{MUL}_{i,n}$ by assigning the value 1 to $y_{i-\lambda(k)}$ and assigning the value 0 to the remaining variables from Y'_B. Since we are interested

in uniformly distributed assignments of the variables from Y, we have to modify this strategy: Let x and y be the numbers that are multiplied. Instead of setting $y := 2^{i-\lambda(k)}$ we choose $y := r \pm 2^{i-\lambda(k)}$ for a random number $r \in \mathbb{Z}_{2^n}$ and compute $z := (x \cdot (r \pm 2^{i-\lambda(k)}))_{[i]}$. Then, with some additional communication, we try to estimate $(x \cdot 2^{i-\lambda(k)})_{[i]}$ from this value. The following protocol shows that $\mathrm{IND}_{n'}$ can be approximated with error ε' by a 1–round Π-$\mathrm{MUL}_{i,n}$-oracle protocol of cost $O(1)$:

Let $x' = (x'_1, \ldots, x'_{n'}) \in \{0,1\}^{n'}$ and $j \in \{1, \ldots, n'\}$ be the inputs of $\mathrm{IND}_{n'}$. The protocol uses the random inputs $r_{X_\mathrm{A}} \in \{0,1\}^{|X_\mathrm{A}|-n'}$, $r_{X_\mathrm{B}} \in \{0,1\}^{|X_\mathrm{B}|}$ and $r_Y \in \{0,1\}^n$. The random inputs are chosen with respect to the uniform distribution. The protocol uses a Π-$\mathrm{MUL}_{i,n}$-oracle. Let $x = (x_{n-1}, \ldots, x_0)$ and $y = (y_{n-1}, \ldots, y_0)$ denote the numbers that are multiplied by the oracle. Alice and Bob compute the output of the protocol according to the following rules:

1. Alice computes her oracle input:
   - Alice sets $(x_{\lambda(1)}, \ldots, x_{\lambda(n')}) := (x'_1, \ldots, x'_{n'})$.
   - Variables from $X_\mathrm{A} \setminus X'_A$ are assigned values from r_{X_A} with respect to some arbitrary but fixed one–to–one mapping of the individual bits.
   - Variables from Y_A are assigned values from r_Y subject to the condition $y = r_Y$.
2. Bob computes his oracle input:
   - Variables from X_B are assigned values from r_{X_B} with respect to some arbitrary but fixed one–to–one mapping of the individual bits.
   - Bob chooses the assignment of the variables from Y_B that satisfies $y = r_Y$, but then he complements the variable $y_{i-\lambda(j)}$. Note that $y = r_Y + 2^{i-\lambda(j)}$ if the value of $y_{i-\lambda(j)}$ was 0 before it was complemented and $y = r_Y - 2^{i-\lambda(j)}$ otherwise.
3. Alice knows the assignment of all variables from X because the variables from X_B are assigned values from r_{X_B} in some fixed way. Alice computes $p = (p_{m_{\varepsilon'}}, \ldots, p_0) := (x \cdot r_Y)_{[i, i-m_{\varepsilon'}]}$ and sends p to Bob.
4. Alice and Bob query the oracle. Bob gets $z := \mathrm{MUL}_{i,n}(x, y)$.
5. Bob computes the output of the protocol: Note that Bob knows the assignment of the variables $x_{\lambda(j)-1}, \ldots, x_{\lambda(j)-m_{\varepsilon'}}$ since, by the choice of $P' \subseteq P$, these variables are assigned values from the public random input r_{X_A} in a fixed way. If $y = r_Y + 2^{i-\lambda(j)}$, then Bob computes $z' := ((0, x_{\lambda(j)-1}, \ldots, x_{\lambda(j)-m_{\varepsilon'}})_2 + (p)_2)_{[m_{\varepsilon'}]}$, otherwise Bob computes $z' := ((1, \overline{x}_{\lambda(j)-1}, \ldots, \overline{x}_{\lambda(j)-m_{\varepsilon'}})_2 + (p)_2)_{[m_{\varepsilon'}]}$. Then Bob uses $z \oplus z'$ as the output of the protocol.

By construction, the cost of the protocol is $m_{\varepsilon'} + 1$. Since the input x' and the random inputs r_{X_A}, r_{X_B} and r_Y are uniformly distributed, by construction, the oracle inputs are also uniformly distributed. To prove $(*)$, we have to show that the approximation error of the protocol is bounded by ε': The oracle of the protocol computes $z = (x \cdot (r_Y \pm 2^{i-\lambda(j)}))_{[i]} = (x \cdot r_Y \pm x \cdot 2^{i-\lambda(j)})_{[i]}$ while we are interested in $(x \cdot 2^{i-\lambda(j)})_{[i]}$. For brevity, let $a := x \cdot r_Y$ denote the first term

in the above sum and let $b := x \cdot 2^{i-\lambda(j)}$ denote the second term. Since we are only interested in the i-th output bit of the multiplication, all computations concerning the multiplication can be done modulo 2^{i+1}. We inspect the cases $z = (a+b)_{[i]}$ and $z = (a-b)_{[i]}$ separately.

Case 1. $z = (a+b)_{[i]}$: Obviously, $a_{[i,0]} = 2^{i-m_{\varepsilon'}} \cdot a_{[i,i-m_{\varepsilon'}]} + a_{[i-m_{\varepsilon'}-1,0]}$ and $b_{[i,0]} = 2^{i-m_{\varepsilon'}} \cdot b_{[i,i-m_{\varepsilon'}]} + b_{[i-m_{\varepsilon'}-1,0]}$. Let $c \in \{0,1\}$ be the carry bit that is propagated into digit $i - m_{\varepsilon'}$ when adding $a_{[i-m_{\varepsilon'}-1,0]}$ and $b_{[i-m_{\varepsilon'}-1,0]}$. Then the following equation holds modulo $2^{m_{\varepsilon'}+1}$:

$$(a+b)_{[i,i-m_{\varepsilon'}]} = a_{[i,i-m_{\varepsilon'}]} + b_{[i,i-m_{\varepsilon'}]} + c. \tag{1}$$

Bob knows the value of $a_{[i,i-m_{\varepsilon'}]}$ since it is equal to Alice's message p. Bob knows the value of $b_{[i-1,i-m_{\varepsilon'}]}$ because, by the choice of the set P', the variables $x_{\lambda(j)-1}, \ldots, x_{\lambda(j)-m_{\varepsilon'}}$ are assigned values from the public random input r_{X_A} in a fixed way. Additionally, Bob knows the oracle's output $z = (a+b)_{[i]}$. Now suppose that $c = 0$ and let z' be defined like in the first case of the last step of the protocol. A simple calculation shows that in this case (1) holds for $b_{[i,i-m_{\varepsilon'}]} = (z \oplus z', x_{\lambda(j)-1}, \ldots, x_{\lambda(j)-m_{\varepsilon'}})$. Thus, under the assumption $c = 0$, Bob computes the correct output $x_{\lambda(j)} = z \oplus z'$. If on the other hand the addition of c has an effect on $(a+b)_{[i]}$, then $(a+b)_{[i-1,i-m_{\varepsilon'}]} = (0, \ldots, 0)$ by the carry–rules of addition. Note that $i \geq \lceil c_{\varepsilon'} n \rceil - 1$ by the choice of P. If we fix the constant $n_{\varepsilon'}$ to a value that satisfies $i \geq \lceil c_{\varepsilon'} n_{\varepsilon'} \rceil - 1 \geq 2m_{\varepsilon'} + 1$, then $\mathrm{Prob}((x \cdot y)_{[i-1,i-m_{\varepsilon'}]} = (0, \ldots, 0)) \leq 2^{-m_{\varepsilon'}+1}$ by Lemma 1. If we ignore the effect of c by fixing c to the constant 0, the approximation error of the protocol is increased by at most $2^{-m_{\varepsilon'}+1}$.

Case 2. $z = (a-b)_{[i]}$: Note that $-b \bmod 2^{i+1}$ is the two's complement of $b \bmod 2^{i+1}$. Thus $-b = -x \cdot 2^{i-\lambda(j)} = (\overline{x}_{\lambda(j)}, \ldots, \overline{x}_0, 1, \ldots, 1)_2 + 1$ holds modulo 2^{i+1}. By the choice of P', there are at least $2m_{\varepsilon'}$ variables in X with an index smaller than $\lambda(j)$. If the addition of 1 has an effect on the value of $(-b \bmod 2^{i+1})_{[i,i-m_{\varepsilon'}]}$, then $(\overline{x}_{\lambda(j)-m_{\varepsilon'}-1}, \ldots, \overline{x}_{\lambda(j)-2m_{\varepsilon'}}) = (1, \ldots, 1)$ must hold due to the carry–rules of addition. By the distribution on the assignments of the variables from X, the probability of this event is smaller than $2^{-m_{\varepsilon'}+1}$. Then, by assuming that $(-b \bmod 2^{i+1})_{[i,i-m_{\varepsilon'}]} = (\overline{x}_{\lambda(j)}, \ldots, \overline{x}_{\lambda(j)-m_{\varepsilon'}})$ and ignoring the effect of the addition of 1, the approximation error is increased by at most $2^{-m_{\varepsilon'}+1}$. Under this assumption we can proceed like in the case $z = (a+b)_{[i]}$ increasing the approximation error by at most $2^{-m_{\varepsilon'}+1}$ again. In all, the approximation error is bounded by $2^{-m_{\varepsilon'}+2}$. Fixing the constant $m_{\varepsilon'}$ to an appropriate value yields $2^{-m_{\varepsilon'}+2} \leq \varepsilon'$ which completes the proof. □

Acknowledgements. I would like to thank Thorsten Bernholt, Martin Sauerhoff and Ingo Wegener for their valuable comments and advice on draft versions of this paper.

References

1. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Transactions on Computers **C-35** (1986) 677–691
2. Wegener, I.: Branching Programs and Binary Decision Diagrams: Theory and Applications. SIAM, Philadelphia, PA (2000)
3. Droste, S., Heutelbeck, D., Wegener, I.: Distributed hybrid genetic programming for learning boolean functions. In: Parallel Problem Solving from Nature - PPSN VI 6th International Conference, Springer Verlag (2000) 181–190
4. Krause, M., Savický, P., Wegener, I.: Approximations by OBDDs and the variable ordering problem. In: Proc. of 26th ICALP. Number 1644 in LNCS, Springer (1999) 493–502
5. Yao, A.C.: Lower bounds by probabilistic arguments. In: Proc. of 24th FOCS. (1982) 420–428
6. Bollig, B., Wegener, I.: Approximability and nonapproximability by binary decision diagrams. Electronic Colloquium on Computational Complexity (ECCC) **7** (2000)
7. Bollig, B., Sauerhoff, M., Wegener, I.: On the nonapproximability of boolean functions by OBDDs and read-k-times branching programs. Information and Computation **178** (2002) 263–278
8. Sauerhoff, M., Woelfel, P.: Time-space tradeoff lower bounds for integer multiplication and graphs of arithmetic functions. In: Proc. of 35th STOC. (2003) 186–195
9. Bryant, R.E.: On the complexity of VLSI implementations and graph representations of boolean functions with application to integer multiplication. IEEE Transactions on Computers **40** (1991) 205–213
10. Bollig, B., Löbbing, M., Sauerhoff, M., Wegener, I.: On the complexity of the hidden weighted bit function for various BDD models. RAIRO Theoretical Computer Science **33** (1999) 103–115
11. Woelfel, P.: New bounds on the OBDD-size of integer multiplication via universal hashing. In: Proc. of 18th STACS. Volume 2010 of Lecture Notes in Computer Science., Springer (2001) 563–574
12. Ablayev, F.M., Karpinski, M.: A lower bound for integer multiplication on randomized read-once branching programs. Proc. of CSIT '99, Electronic Edition (1999)
13. Ponzio, S.: A lower bound for integer multiplication with read-once branching programs. In: Proc. of 27th STOC. (1995) 130–139
14. Bollig, B., Woelfel, P.: A read-once branching program lower bound of $\Omega(2^{n/4})$ for integer multiplication using universal hashing. In: Proc. of 33rd STOC. (2001) 419–424
15. Yao, A.C.: Some complexity questions related to distributed computing. In: Proc. of 11th STOC. (1979) 209–213
16. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)
17. Babai, L., Frankl, P., Simon, J.: Complexity classes in communication complexity theory. In: Proc. of 27th FOCS, IEEE (1986) 337–347
18. Sedgewick, R., Flajolet, P.: An Introduction to the Analysis of Algorithms. Addison-Wesley (1996)
19. Kremer, I., Nisan, N., Ron, D.: On randomized one-round communication complexity. In: Proc. of 27th STOC. (1995) 596–605
20. Dietzfelbinger, M., Hagerup, T., Katajainen, J., Penttonen, M.: A reliable randomized algorithm for the closest pair problem. Journal of Algorithms **25** (1997) 19–51

On Approximation Hardness of the Minimum 2SAT-DELETION Problem

Miroslav Chlebík[1] and Janka Chlebíková[2*]

[1] MPI for Mathematics in the Sciences, D-04103 Leipzig, Germany,
chlebik@mis.mpg.de
[2] Department of Informatics Education, Faculty of Mathematics, Physics and Informatics, Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia,
chlebikj@dcs.fmph.uniba.sk

Abstract. The MINIMUM 2SAT-DELETION problem is to delete the minimum number of clauses in a 2SAT instance to make it satisfiable. It is one of the prototypes in the approximability hierarchy of minimization problems [8], and its approximability is largely open. We prove a lower approximation bound of $8\sqrt{5} - 15 \approx 2.88854$, improving the previous bound of $10\sqrt{5} - 21 \approx 1.36067$ by Dinur and Safra [5]. For highly restricted instances with exactly 4 occurrences of every variable we provide a lower bound of $\frac{3}{2}$. Both inapproximability results apply to instances with no mixed clauses (the literals in every clause are both either negated, or unnegated).
We further prove that any k-approximation algorithm for MINIMUM 2SAT-DELETION polynomially reduces to a $(2 - \frac{2}{k+1})$-approximation algorithm for the MINIMUM VERTEX COVER problem.
One ingredient of these improvements is our proof that for the MINIMUM VERTEX COVER problem restricted to graphs with a perfect matching its threshold on polynomial time approximability is the same as for the general MINIMUM VERTEX COVER problem. This improves also on results of Chen and Kanj [3].

Keywords: Vertex cover, perfect matching, satisfiability, combinatorial optimization problems, approximation algorithm, inapproximability, Nemhauser-Trotter Theorem

1 Introduction

The proof of the PCP Theorem and subsequent improvements in PCP constructions have led for many optimization problems to optimal bounds on their efficient approximability (unless P = NP). However, in spite of a great deal of efforts, for several fundamental problems the tight bound on their approximability by a polynomial time algorithm is left open, for example: *Is there a polynomial time approximation algorithm for the* MINIMUM VERTEX COVER *problem whose approximation factor is bounded above by a constant less than* 2 ? *Is there*

* The author has been partially supported by the grant VEGA 1/0131/03.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 263–273, 2004.
© Springer-Verlag Berlin Heidelberg 2004

a polynomial time algorithm that approximates the MINIMUM 2SAT-DELETION *problem to within a constant factor?*

Recall that the task in the MINIMUM 2SAT-DELETION problem (shortly, MIN-2SAT-DELETION) is to delete the minimum number of clauses in a 2SAT instance to make it satisfiable. The problem is important as one of the prototypes in a complete classification of the approximability of minimization problems derived from Boolean constraint satisfaction [8].

Currently, the best lower bound on polynomial time approximability is the same for both problems, namely $10\sqrt{5}-21 \approx 1.36067$, due to Dinur and Safra [5]. For the MIN-2SAT-DELETION problem we can improve on this bound, even on highly restricted instances where every variable occurs in very small number of clauses. We show how two questions mentioned above are related to one another: the affirmative answer to the second question would imply the affirmative answer to the first one.

Our Results

We concentrate mainly on the MINIMUM VERTEX COVER problem (MIN-VC) on graphs with a perfect matching (MIN-VC-PM). It turns out that the problems to solve MIN-VC exactly, and to approximate MIN-VC within a factor on general graphs, reduce to the corresponding problems on graphs with a perfect matching. In particular, the threshold on polynomial time approximability of MIN-VC is the same as for MIN-VC-PM. Moreover, we observe that the NP-hard gap results of Dinur and Safra [5] for MIN-VC apply to MIN-VC-PM as well. Using this fact and the powerful reduction from MIN-VC-PM to MIN-2SAT-DELETION we can improve inapproximability results for MIN-2SAT-DELETION. We prove that it is NP-hard to approximate MIN-2SAT-DELETION to within any constant factor less than $8\sqrt{5}-15 \approx 2.88854$. We provide interesting lower bound also for small occurrence instances: it is NP-hard to approximate MIN-2SAT-DELETION to within any constant factor less than $\frac{3}{2}$ on instances with exactly 4 occurrences of every variable. Both inapproximability results apply to instances with no mixed clauses (i.e., only clauses $x \vee y$, $\overline{x} \vee \overline{y}$, with x and y distinct variables, are allowed).

We further prove that the existence of a polynomial time approximation algorithm for MIN-2SAT-DELETION with a constant factor k, $k \in (1, \infty)$, would imply the existence of a polynomial time $(2-\frac{2}{k+1})$-approximation algorithm for MIN-VC. Even slight improvement to $o\left(\frac{\log n}{\log \log n}\right)$ of the approximation factor for MIN-2SAT-DELETION will improve on currently the best polynomial time approximation factor $\left(2-\frac{\log \log n}{2 \log n}\right)$ for MIN-VC ([1], [12]). Moreover, we observe that the existence of an algorithm for MAX-2SAT that for some $t > \frac{1}{2}$ for every instance finds an assignment satisfying at least a fraction $(1-O(\varepsilon^t))$ of clauses, assuming that optimal assignment satisfies a fraction $(1-\varepsilon)$ of them, will improve on known results for MIN-VC in sparse graphs.

Preliminaries

Let us start with the formulation of problems we will deal with and some known results about them.

Definition 1. *Consider a set of* clauses $C_1, C_2, \ldots, C_N$ *with each clause* C_i *of the form* $l_1 \vee l_2$ *and nonnegative weights* w_i. *Each literal* l_j *is either one of Boolean variables* $x_1, x_2, \ldots, x_n$, *or its negation. A common variant of the above is that all weights are the same but clauses can be repeated. The goal of the* MAX-2SAT *problem is to assign Boolean values* 0 *and* 1 *to variables* $x_1, x_2, \ldots, x_n$ *so that the total weight of the satisfied clauses is maximized. For the complementary problem of* minimum unsatisfiability, MIN-2SAT-DELETION *(called also* MIN-2CNF-DELETION *by some authors), the goal is to minimize the total weight of unsatisfied clauses over all assignments.*

While MAX-2SAT is approximable within 1.0638 [11] and it is NP-hard to approximate within 1.0476 [6], the approximability of the MIN-2SAT-DELETION problem is still widely open problem. The problem is known to be approximable in polynomial time to within polylogarithmic factor $O(\log n \log\log n)$ (Klein et al. [9]) and, on the other hand, to be NP-hard to approximate to within 1.36067 [5].

Let $G = (V, E)$ be a simple graph. For a set of vertices $U \subseteq V$, let $\Gamma(U) := \{v \in V : \exists u \in U \text{ such that } \{u, v\} \in E\}$ stand for the set of its neighbors, and $G[U]$ denote the subgraph of G induced by U.

MINIMUM VERTEX COVER
Instance: A simple graph $G = (V, E)$.
Feasible solution: A vertex cover C for G, i.e., a subset $C \subseteq V$ such that for each $e \in E$, $e \cap C \neq \emptyset$.
Objective function: The cardinality $|C|$ of the vertex cover C.

Let $vc(G)$ stand for the cardinality of the minimum vertex cover for G. Similarly, a half-integral vertex cover for $G = (V, E)$ is a function $x : V \to \{0, \frac{1}{2}, 1\}$ satisfying edge constraints $x(u) + x(v) \geq 1$ for each edge $\{u, v\} \in E$. Let $vc_*(G)$ stand for the cardinality of the minimum half-integral vertex cover for G, i.e., the minimum of $w(x) := \sum_{u \in V} x(u)$ over all half-integral vertex covers x.

Clearly, $vc_*(G) \leq vc(G)$, as for any vertex cover C its indicator function x^C is also a half-integral vertex cover with $w(x^C) = |C|$. Further, $vc_*(G) \leq \frac{1}{2}|V|$, as the function $x \equiv \frac{1}{2}$ on V is always the half-integral vertex cover for G.

The following theorem of Nemhauser and Trotter [13] is of great importance for many problems related to MIN-VC problems. It allows to reduce the problem to instances, where the value of a minimum vertex cover is at least $\frac{1}{2}|V|$.

Nemhauser-Trotter Theorem (NT-Theorem). *There exists a polynomial time algorithm that partitions the vertex set* V *of any graph* G *into three subsets* $V_0, V_1, V_{\frac{1}{2}}$ *with no edges between* V_0 *and* $V_{\frac{1}{2}}$ *or within* V_0 *such that*

(i) $vc(G[V_{\frac{1}{2}}]) \geq vc_*(G[V_{\frac{1}{2}}]) = \frac{1}{2}|V_{\frac{1}{2}}|$; *and*

(ii) there exists some minimum vertex cover C for G such that $V_1 \subseteq C \subseteq V_1 \cup V_{\frac{1}{2}}$ and $C \cap V_{\frac{1}{2}}$ is a minimum vertex cover for $G[V_{\frac{1}{2}}]$.

In what follows we define for a graph G its bipartite version G^b and observe that the minimum half-integral vertex covers for G are generated by minimum vertex covers for the corresponding bipartite graph G^b.

Definition 2. *For a graph $G = (V, E)$ we define its bipartite version $G^b = (V^b, E^b)$, as follows: there are two copies u^L and u^R of each vertex $u \in V$ in G^b, $V^L := \{u^L : u \in V\}$, $V^R := \{u^R : u \in V\}$, and $V^b := V^L \cup V^R$. Each edge $\{u, v\} \in E$ of G creates two edges in G^b, namely $\{u^L, v^R\}$ and $\{v^L, u^R\}$. Hence $E^b := \{\{u^L, v^R\}, \{v^L, u^R\} : \{u, v\} \in E\}$. For $U \subseteq V$ we use also U^L, U^R, and $U^b := U^L \cup U^R$ for the corresponding sets of vertices.*

For any set $C \subseteq V^{\mathrm{L}} \cup V^{\mathrm{R}}$ we associate a map $x_C : V \to \{0, \frac{1}{2}, 1\}$ in the following way: $x_C(u) = \frac{1}{2}|C \cap \{u^{\mathrm{L}}, u^{\mathrm{R}}\}|$ for any $u \in V$. Clearly $w(x_C) = \frac{1}{2}|C|$ for any $C \subseteq V^{\mathrm{L}} \cup V^{\mathrm{R}}$.

Lemma 1. *(i) If C is a vertex cover for G^b then x_C is a half-integral vertex cover for G of weight $\frac{1}{2}|C|$. In particular, $vc_*(G) \leq \frac{1}{2}vc(G^b)$.*
(ii) If $x : V \to \{0, \frac{1}{2}, 1\}$ is a half-integral vertex cover for G then there is a vertex cover C for G^b such that $x_C = x$. Hence $\frac{1}{2}vc(G^b) \leq vc_*(G)$.
(iii) $vc_*(G) = \frac{1}{2}vc(G^b)$

Proof. (i) being obvious, it is sufficient to show (ii) and the rest will trivially follow. Given $x : V \to \{0, \frac{1}{2}, 1\}$ as required, let $V_i^x := \{u \in V : x(u) = i\}$ for $i \in \{0, \frac{1}{2}, 1\}$. Obviously, there are no edges between V_0^x and $V_{\frac{1}{2}}^x$ or within V_0^x. It follows that a set C defined by $C := \bigcup_{u \in V_1^x} \{u^{\mathrm{L}}, u^{\mathrm{R}}\} \cup \bigcup_{u \in V_{\frac{1}{2}}^x} \{u^{\mathrm{L}}\}$ is a vertex cover for G^b such that $x_C = x$. □

2 Min-VC for Graphs with Perfect Matching

This section concentrates on the MINIMUM VERTEX COVER problem on graphs with a perfect matching. We will show that the problems to solve MIN-VC exactly and to approximate MIN-VC to within a factor in general graphs reduce to the corresponding MIN-VC problems on graphs with a perfect matching.

Definition 3. *Given a graph $G = (V, E)$, let $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ be a 2-padding of G with the same vertex set as G^b (Definition 2), i.e., $\widetilde{V} = V^b$ and*

$$\widetilde{E} = E^b \cup \{\{u^L, v^L\}, \{u^R, v^R\} : \{u, v\} \in E\}.$$

Theorem 1. *The threshold on polynomial time approximability of the* MIN-VC *problem is the same as the one for the* MIN-VC-PM *problem.*

Proof. Nemhauser-Trotter Theorem reduces the MIN-VC problem for $G = (V, E)$ in approximation preserving way to its induced subgraph $G[V_{\frac{1}{2}}]$ with the property $vc(G[V_{\frac{1}{2}}]) \geq vc_*(G[V_{\frac{1}{2}}]) = \frac{1}{2}|V_{\frac{1}{2}}|$.

Let $\widetilde{G}$ be a 2-padding of G. Obviously if $C \subseteq V$ is a vertex cover in G then $\widetilde{C} := \cup_{u \in C}\{u^{\mathrm{L}}, u^{\mathrm{R}}\}$ is a vertex cover in a $\widetilde{G}$. Moreover, every minimal (on inclusion) vertex cover of $\widetilde{G}$ is of the form above for some vertex cover C of G.

Using Lemma 1(iii) and König-Egerváry Theorem (see, e.g., [14]) the fact $vc_*(G[V_{\frac{1}{2}}]) = \frac{1}{2}|V_{\frac{1}{2}}|$ applied to $G[V_{\frac{1}{2}}]$ implies that $(G[V_{\frac{1}{2}}])^b$ has a perfect matching, which in turn implies that $\widetilde{G[V_{\frac{1}{2}}]}$ has a perfect matching. But problems MIN-VC for $G[V_{\frac{1}{2}}]$ and $\widetilde{G[V_{\frac{1}{2}}]}$ are essentially equivalent. Hence the MIN-VC problem reduces to the MIN-VC-PM problem. □

We can show explicitly that the NP-hard gap result of Dinur and Safra [5] applies to the MIN-VC-PM problem as well. As their construction is rather involved, we don't reproduce it and refer the reader to [5] for more details. Consider one of weighted graphs they generate, then there is clearly a perfect matching within every long-code block by connecting each subset to its complement. However, after standard multiplication the vertices according to (positive rational) weights to get the unweighted version of their NP-hard gap result, such matching is no longer perfect. The question of whether there is a perfect matching in this graph reduces to the one of whether there is a perfect matching in the Kneser graph, the main building block of their construction. But it is well known even for every connected vertex transitive graph that it has a perfect matching if it has even number of vertices, and it has a nearly perfect matching if the number of vertices is odd (see [10]). Therefore their NP-hard gap result applies to MIN-VC-PM as well. It can be stated as follows

Theorem 2. (Dinur and Safra) *Let p, q be constants such that $\frac{3-\sqrt{5}}{2} > p > q > \max\{p^2, 4p^3 - 3p^4\}$. It is* NP*-hard for graphs $G = (V, E)$ with a perfect matching to distinguish between the following two cases: $vc(G) < (1-p)|V|$, or $vc(G) > (1-q)|V|$.*

The Theorem 1 is true for general graphs, or for any class $\mathcal{G}$ of graphs that is closed on operation of taking a 2-padding of an induced subgraph. If the class $\mathcal{G}$ of graphs does not have this property, but it is at least closed on the operation of taking an induced subgraph (e.g., $\mathcal{G} = \{\text{graphs with the maximum degree} \leq B\}$, or $\mathcal{G} = \{$everywhere B-sparse graphs, for some constant $B\}$), we can conclude that MIN-VC restricted to $\mathcal{G}$ is as hard to approximate as MIN-VC restricted to $\mathcal{G}'$, where $\mathcal{G}'$ consists of graphs $G = (V, E)$ from $\mathcal{G}$ with $vc_*(G) = \frac{1}{2}|V|$. In such classes $\mathcal{G}$ we do not know if MIN-VC-PM is as hard to approximate as MIN-VC, or easier.

3 Application to 2SAT Problems

In what follows we relate the MINIMUM VERTEX COVER problem to MAX-2SAT and similar problems. Similarly as in [3] we use the following reduction from MIN-VC-PM to MAX-2SAT.

2SAT-construction. Let $G = (V, E)$ be an instance of MIN-VC-PM and let M be a fixed perfect matching in G. Define an instance $\mathcal{F}(G, M)$ of 2SAT as follows: the Boolean variable set $X_V := \{x_u : u \in V\}$ and

$$\mathcal{F}(G, M) = \bigcup_{\{u,v\} \in E} \{(x_u \vee x_v)\} \cup \bigcup_{\{u,v\} \in M} \{(\overline{x}_u \vee \overline{x}_v)\}.$$

Hence $\mathcal{F}(G, M)$ consists of $N := |E| + \frac{1}{2}|V|$ clauses. All the clauses have exactly 2 (different) literals and are non-mixed, i.e., none of the clauses has both negated and unnegated literals. This variant of 2SAT is sometimes referred to as E2-NM-SAT (NM stands for non-mixed clauses). Obviously the instance $\mathcal{F} := \mathcal{F}(G, M)$ of E2-NM-SAT defined above has three additional properties:

(P1) clauses are not repeated,

(P2) each variable appears exactly once as negated; and

(P3) if $(\overline{a} \vee \overline{b}) \in \mathcal{F}$ for variables a and b, then $(a \vee b) \in \mathcal{F}$ as well.

Let $\mathrm{unsat}(\sigma)$ be the number of clauses of $\mathcal{F}(G, M)$ that are unsatisfied by an assignment $\sigma : X_V \to \{0, 1\}$, and $\mathrm{OPT}(\mathcal{F}(G, M))$ be the minimum of $\mathrm{unsat}(\sigma)$ over all assignments $\sigma : X_V \to \{0, 1\}$. Then the following lemma holds

Lemma 2. *Let M be a perfect matching in a graph $G = (V, E)$ and the collection of clauses $\mathcal{F}(G, M)$ of Boolean variables X_V be according to the construction above. Then*

(i) $\mathrm{OPT}(\mathcal{F}(G, M)) = vc(G) - \frac{1}{2}|V|$,

(ii) From any assignment $\sigma : X_V \to \{0, 1\}$ a vertex cover C for G of cardinality at most $(\frac{1}{2}|V| + \mathrm{unsat}(\sigma))$ can be constructed in time $O(|E|)$.

Proof. If C is a vertex cover in G, let $\sigma_C : X_V \to \{0, 1\}$ denote the following assignment: $\sigma_C(x_u) = 1$ iff $u \in C$. The fact that C was a vertex cover means exactly that all clauses with unnegated variables are satisfied by σ_C. Such assignment to variables will be called *standard*. Clearly, standard assignments $\sigma : X_V \to \{0, 1\}$ are in one-to-one correspondence with vertex covers in G.

It is clear that for a standard assignment σ_C, the edges in a perfect matching M are partitioned into two sets M_1 and M_2 such that each edge $\{u, v\}$ in M_i has exactly i endpoints in C, $i = 1, 2$. Thus $|C| = |M_1| + 2 \cdot |M_2|$, and clearly $\mathrm{unsat}(\sigma_C) = |M_2| = |C| - (|M_1| + |M_2|) = |C| - \frac{1}{2}|V|$. Consequently, $\mathrm{OPT}(\mathcal{F}(G, M)) \leq vc(G) - \frac{1}{2}|V|$.

Let now an assignment $\sigma : X_V \to \{0, 1\}$ be given. We can modify σ (in time $O(|E|)$) to the standard assignment σ_C such that $\mathrm{unsat}(\sigma_C) \leq \mathrm{unsat}(\sigma)$. In

another words, we can provide in time $O(|E|)$ a vertex cover C with $\mathrm{unsat}(\sigma_C) = |C| - \frac{1}{2}|V| \leq \mathrm{unsat}(\sigma)$.

We will process edges $\{u,v\} \in E$, one after another, as follows: If $\sigma(x_u) = \sigma(x_v) = 0$ for an edge $\{u,v\} \in E$, then we modify σ at exactly one endpoint of $\{u,v\}$, say u, setting $\sigma(x_u) = 1$ instead. This change does not increase the value $\mathrm{unsat}(\sigma)$: the unsatisfied clause $(x_u \vee x_v)$ becomes satisfied, and at most one satisfied clause can become unsatisfied, namely the one containing the literal $\overline{x}_u$. Having this done, one after another, for all edges $\{u,v\} \in E$, the resulting assignment will be standard, hence of the form σ_C for a vertex cover C of G, and $\mathrm{unsat}(\sigma_C) \leq \mathrm{unsat}(\sigma)$.

In particular, it follows (taking σ with $\mathrm{unsat}(\sigma) = \mathrm{OPT}(\mathcal{F}(G,M))$ that $vc(G) - \frac{1}{2}|V| = \mathrm{OPT}(\mathcal{F}(G,M))$. □

Using the 2SAT-construction above from MIN-VC-PM to special instances of MIN-2SAT-DELETION we can obtain from Theorem 2

Theorem 3. *Let p, q be constants such that $\frac{3-\sqrt{5}}{2} > p > q > \max\{p^2, 4p^3 - 3p^4\}$. It is NP-hard for instances $\mathcal{F}$ of* MIN-2SAT-DELETION *to distinguish between two cases: $\mathrm{OPT}(\mathcal{F}) < (\frac{1}{2} - p)|X|$ or $\mathrm{OPT}(\mathcal{F}) > (\frac{1}{2} - q)|X|$, where X is the set of Boolean variables of $\mathcal{F}$. Consequently, it is* NP*-hard to approximate the* MIN-2SAT-DELETION *problem to within any constant approximation factor smaller than $8\sqrt{5} - 15 \approx 2.88854$. The same* NP*-hardness result applies to instances with no mixed clauses, and satisfying conditions* (P1)–(P3).

Proof. The $((\frac{1}{2}-p)|X|, (\frac{1}{2}-q)|X|)$-gap result follows directly from the reduction $G \mapsto \mathcal{F}(G,M)$ using Theorem 1 and Lemma 2. Hence inapproximability to within $\frac{1-2q}{1-2p}$ follows, for any p, q satisfying our assumptions. Notice that for $p \in (\frac{1}{3}, 1)$, $\max\{p^2, 4p^3 - 3p^4\} = 4p^3 - 3p^4$, and as q can approach $4p^3 - 3p^4$ (from above) and p can approach $\frac{3-\sqrt{5}}{2}$ (from below), $\frac{1-2q}{1-2p}$ can approach $8\sqrt{5}-15$ (from below). Hence NP-hardness to approximate the problem to within any constant factor smaller than $8\sqrt{5} - 15$ follows. □

Assume now that we have an approximation algorithm $\mathcal{A}$ for the MIN-2SAT-DELETION problem. The above polynomial time 2SAT-construction suggests an approximation algorithm (based on $\mathcal{A}$) for MIN-VC-PM and, consequently, for the MIN-VC problem.

Theorem 4. *Given an algorithm that approximates the solution of the* MIN-2SAT-DELETION *problem within approximation factor $f(n,N) \geq 1$ on instances with n variables and N clauses, all non-mixed and satisfying* (P1)–(P3). *(Here $f : \mathbb{N}_+^2 \to \langle 1, \infty)$ is a function separately nondecreasing in every variable.) It can be reduced to the one that approximates* MIN-VC-PM *(respectively,* MIN-VC*) on instances with n vertices and m edges within $2 - \frac{2}{f(n, m+\frac{n}{2})+1}$ (respectively, $2 - \frac{2}{f(2n,4m+n)+1}$).*

Proof. Let $G = (V, E)$ be a graph with $n := |V|$ and $m := |E|$. We can assume that G has a perfect matching. Otherwise we can work with a graph $\widetilde{G}[V_{\frac{1}{2}}]$, that

is a graph with $n' \leq 2n$ vertices, $m' \leq 4m$ edges, and with a perfect matching. Let $\mathcal{A}$ be an f-approximation algorithm for MIN-2SAT-DELETION, and consider the following algorithm:

Step 1: Construct a perfect matching M in G,
Step 2: construct the corresponding instance $\mathcal{F}(G, M)$ of the MIN-2SAT-DELETION problem, with n variables and $N := m + \frac{n}{2}$ clauses,
Step 3: applying the algorithm $\mathcal{A}$ to $\mathcal{F}(G, M)$ construct an assignment $\sigma : X_V \to \{0, 1\}$ that approximates the optimal solution for $\mathcal{F}(G, M)$ within $f := f(n, N)$,
Step 4: construct a vertex cover C in G of cardinality at most $(\frac{1}{2}|V| + \mathrm{unsat}(\sigma))$, according to Lemma 2,
Step 5: return a vertex cover C of G.

Our aim is to show that the algorithm returns a vertex cover C of G, with the property $|C| \leq vc(G) \cdot (2 - \frac{2}{f+1})$. The assumptions on $\mathcal{A}$ guarantee that the assignment σ provided in Step 3 satisfies

$$\mathrm{unsat}(\sigma) \leq \mathrm{OPT}(\mathcal{F}(G, M)) \cdot f. \tag{1}$$

If $\mathrm{unsat}(\sigma) \geq \frac{1}{2}|V|$, we conclude from (1) and from Lemma 2 that

$$vc(G) = \frac{1}{2}|V| + \mathrm{OPT}(\mathcal{F}(G, M)) \geq \frac{1}{2}|V| + \frac{1}{2}\frac{|V|}{f}.$$

As clearly $|C| \leq |V|$, $\frac{|C|}{vc(G)} \leq 2 - \frac{2}{f+1}$ easily follows.

If $\mathrm{unsat}(\sigma) = \frac{1}{2}|V| \cdot t$ for $t \in \langle 0, 1\rangle$, we conclude similarly that

$$vc(G) \geq \frac{1}{2}|V| + \frac{1}{2}|V|\frac{t}{f},$$

but now we use better estimate on $|C|$ from Step 4, $|C| \leq \frac{1}{2}|V| + \frac{1}{2}|V|t$. Hence

$$\frac{|C|}{vc(G)} \leq \frac{1+t}{1+\frac{t}{f}} = \frac{(1+t)f}{f+t}.$$

The right hand side as a function of $t \in \langle 0, 1\rangle$ achieves its maximum at $t = 1$ (we use that $f \geq 1$ for this argument), hence

$$\frac{|C|}{vc(G)} \leq \frac{2f}{f+1} = 2 - \frac{2}{f+1}. \qquad \square$$

Corollary 1. *To approximate* MIN-2SAT-DELETION *within a constant k is at least as difficult (up to polynomial reduction between problems) as to approximate* MIN-VC *within a factor* $2 - \frac{2}{k+1}$.

If we apply Theorem 4 to the $O(\log n \log\log n)$-approximation algorithm given for MIN-2SAT-DELETION by Klein et al. [9], we obtain an algorithm for MIN-VC with the approximation factor $(2 - \frac{1}{c \log n \log\log n})$. An improvement to $o\left(\frac{\log n}{\log\log n}\right)$ of the above factor for MIN-2SAT-DELETION (at least on instances with non-mixed clauses, satisfying (P1)–(P3)) would improve on currently the best polynomial time approximation factor $(2 - \frac{\log\log n}{2 \log n})$ for the MIN-VC problem.

Theorem 4 and its proof can be modified also to situations when the quality of approximation of the algorithm $\mathcal{A}$ for MIN-2SAT-DELETION is measured in different parameters. Consider an algorithm $\mathcal{A}$ for MIN-2SAT-DELETION that is *robust on almost-satisfiable instances.* That means, for an instance $\mathcal{F}$ of MIN-2SAT-DELETION, whose optimum assignment leaves only ε fraction of clauses unsatisfied, $\mathcal{A}$ finds an assignment that leaves at most $g(\varepsilon)$ fraction of clauses of $\mathcal{F}$ unsatisfied, where $g : \langle 0,1\rangle \to \langle 0,1\rangle$ is a function with $\lim_{\varepsilon \to 0} g(\varepsilon) = g(0) = 0$.

Zwick's efficient algorithm [15] has this robustness property with $g(\varepsilon) = 5\varepsilon^{1/3}$. An interesting question is whether such algorithms exist with $g(\varepsilon) = o(\varepsilon^{1/2})$, and if yes, how far one can go beyond this bound. One can easily check (along the lines of the proof of Theorem 4) that any robust algorithm with $g(\varepsilon) = O(\varepsilon^t)$, $t \in (0,1\rangle$, gives us an $(2 - c \cdot (1+\bar{d})^{1-1/t})$-approximation algorithm for MIN-VC-PM on graphs with average degree $\bar{d} := \frac{2|E|}{|V|}$ (here $c \in (0,1)$ is an absolute constant). For any $t > \frac{1}{2}$ the existence of a robust algorithm with $g(\varepsilon) = O(\varepsilon^t)$ would significantly improve (for large $\bar{d}$) on currently the best approximation factor $(2 - \frac{5}{2\bar{d}+3})$ on polynomial time approximation for MIN-VC-PM on graphs with average degree $\bar{d}$ [7].

3.1 Bounded Occurrence Instances of 2SAT Problems

One can obtain inapproximability results also for bounded occurrence instances of MIN-2SAT-DELETION using inapproximability results for MIN-VC on bounded degree graphs. We will show that on an example of cubic graphs. One can check that instances produced in [4] to achieve the inapproximability results for the MIN-VC problem in cubic graphs, have a perfect matching. It is proven there that it is NP-hard for a cubic graph G with n vertices and having a perfect matching to distinguish the case $vc(G) < (\frac{1}{2} + 2\delta + \varepsilon)n$ from the case $vc(G) > (\frac{1}{2} + 3\delta - \varepsilon)n$ where $\delta > 0$ is a positive constant related to parameters of an amplifier used in the construction, and $\varepsilon \in (0, \frac{\delta}{2})$ can be arbitrarily small independently of δ. The instance $\mathcal{F}(G,M)$ that corresponds to such G (and an arbitrary matching M in G) has n variables, $2n$ clauses (all non-mixed and satisfying (P1)–(P3)) with exactly 4 occurrences of each variable. Due to Lemma 2, the corresponding NP-hard problem is to distinguish $\mathrm{OPT}(\mathcal{F}(G,M)) < (2\delta+\varepsilon)n$ from $\mathrm{OPT}(\mathcal{F}(G,M)) > (3\delta - \varepsilon)n$. We have just proved

Theorem 5. *It is* NP*-hard to approximate the* MIN-2SAT-DELETION *problem to within any constant approximation factor smaller than* $\frac{3}{2}$ *on instances with*

exactly 4 *occurrences of each variable, no mixed clauses, and satisfying conditions* (P1)–(P3).

In this way one can derive inapproximability results also for the complementary MAX-2SAT satisfiability problem with exactly four occurrence of each variable. The corresponding NP-hard problem is to decide whether the optimum is greater than $(2 - 2\delta - \varepsilon)n$, or less than $(2 - 3\delta + \varepsilon)n$. Now the inapproximability factor $\approx \frac{2-2\delta}{2-3\delta}$ depends crucially on parameters of an amplifier (hidden in δ) used in that hardness result for cubic graphs. From estimates of [4], the inapproximability to within $1 + \frac{1}{385}$ follows. It is worse than the recent hardness factor $1 + \frac{1}{268}$ obtained for this problem in [2], but on the other hand, it applies to instances with no mixed clauses.

Concluding Remarks

It has been conjectured by several authors that it is NP-hard to approximate MIN-VC within any constant factor smaller than 2. By Theorem 4 and Corollary 1 this would imply also NP-hardness of approximating MIN-2SAT-DELETION within any constant factor. We think that the latter problem is significantly easier than the former one. The methods used in this paper show that in order to prove NP-hardness to approximate MIN-2SAT-DELETION within (any fixed constant) k, it suffices to provide instances $G = (V, E)$ with a perfect matching for which it is NP-hard to distinguish $vc(G) < \frac{1}{2}|V| + \varepsilon(G)$ from $vc(G) > \frac{1}{2}|V| + k \cdot \varepsilon(G)$, for some efficiently computable function $\varepsilon = \varepsilon(G) > 0$. On the other hand, the problem MIN-VC requires to show, for arbitrarily small constant $\epsilon > 0$, NP-hardness to distinguish instances with $vc(G) < (\frac{1}{2} + \varepsilon)|V|$ from those with $vc(G) > (1 - \varepsilon)|V|$.

References

1. R. Bar-Yehuda and S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, Annals of Discrete Mathematics 25(1985), 27–46.
2. P. Berman and M. Karpinski, Improved approximation lower bounds on small occurence optimization, ECCC Report TR03-08, 2003.
3. J. Chen and I. A. Kanj, On approximating minimum vertex cover for graphs with perfect matching, Proc. of the 11st ISAAC, Taipei, Taiwan, LNCS 1969, Springer, 2000, pp. 132–143.
4. M. Chlebík and J. Chlebíková, Inapproximability results for bounded variants of optimization problems, Proc. of the 14th FCT, LNCS 2751, Springer Verlag, 2003, pp. 27–38, also ECCC Report TR03-26, 2003.
5. I. Dinur and S. Safra, The importance of being biased, STOC 2002, pp. 33–42, also ECCC Report TR01-104, 2001.
6. J. Håstad, Some optimal inapproximability results, Journal of ACM 48(2001), 798–859.
7. M. Halldórsson and J. Radhakrishnan, Greed is good: approximating independent sets in sparse and bounded-degree graphs, Algorithmica 18 (1997), 145–163.

8. S. Khanna, M. Sudan and L. Trevisan, Constraint Satisfaction: The approximability of minimization problems, Proceedings of the 12th Annual IEEE Conference on Computational Complexity, Ulm, Germany, 24-27 June, 1997, pp. 282–296.
9. P. Klein, S. Plotkin, S. Rao and E. Tardos, Approximation algorithms for Steiner and directed multicuts, Journal of Algorithms 22(2) (1997), 241–269.
10. C. H. Little, D. D. Grant and D. A. Holton, On defect d matchings in graphs, Discrete Mathematics 13(1975), 41–54. Erratum in Disc. Math. 14(1976), 203.
11. M. Lewin, D. Livnat and U. Zwick, Improved rounding techniques for the MAX-2SAT and MAX DI CUT problems, IPCO 2002, LNCS 2337, pp. 67–82.
12. B. Monien and E. Speckenmeyer, Ramsey numbers and an approximation algorithm for the vertex cover problem, Acta Informatica 22(1985), 115–123.
13. G. L. Nemhauser and L. E. Trotter, Vertex packings: structural properties and algorithms, Math. Programming 8(1975), 232–248.
14. D. B. West, Introduction to graph theory, Prentice-Hall, 2001.
15. U. Zwick, Finding almost satisfying assignments, Proc. of the 30th Annual ACM Symposium on Theory of Computing, 1998. pp. 551–560.

Group Coloring and List Group Coloring Are Π_2^P-Complete
(Extended Abstract)

Daniel Král' and Pavel Nejedlý

Department of Applied Mathematics and
Institute for Theoretical Computer Science*
Charles University
Malostranské náměstí 25, 118 00 Prague 1, Czech Republic.
{kral,bim}@kam.mff.cuni.cz

Abstract. A graph G is A-ℓ-choosable for an Abelian group A and an integer $\ell \leq |A|$ if for each orientation of G, each edge-labeling $\varphi : E(G) \to A$ and each list-assignment $L : V(G) \to \binom{A}{\ell}$, there exists a vertex-coloring $c : V(G) \to A$ with $c(v) \in L(v)$ for each vertex v and with $c(v) - c(u) \neq \varphi(uv)$ for each oriented edge uv of G. We prove a dichotomy result on the computational complexity of this problem. In particular, we show that the problem is Π_2^P-complete if $\ell \geq 3$ for any group A and it is polynomial-time solvable if $\ell = 1, 2$. This also settles the complexity of group coloring for all Abelian groups.

1 Introduction

Group colorings of graphs have been introduced by Jaeger, Linial, Payan and Tarsi [5]. For plane graphs, this concept is dual (in the usual sense) to group connectivity which generalizes the intensively studied concept of nowhere-zero flows in graphs. For an Abelian group A, a graph G is said to be *A-colorable* if for every orientation of G and for every *edge-labeling* $\varphi : E(G) \to A$, there is a vertex coloring $c : V(G) \to A$ such that $c(w) - c(v) \neq \varphi(vw)$ for each oriented edge $vw \in E(G)$. Note that the choice of an orientation of edges of G is not essential since reversing the orientation of the edge e can be replaced by changing the forbidden difference $\varphi(e)$ to $-\varphi(e)$. A plane graph is A-colorable iff its dual is A-connected [5]. We remark that it is unknown whether the property of being A-colorable (A-connected) depends on the structure or only on the order of the group A (like in the case of nowhere-zero flows [13]). The least number $\chi_g(G)$ such that G is A-colorable for each Abelian group A of order at least $\chi_g(G)$ is the *group chromatic number* of G.

Group colorings have attracted a lot of attention from the combinatorial point of view. Some theorems for ordinary colorings, e.g., Brooks' theorem, can

* Institute for Theoretical Computer Science (ITI) is supported by Ministry of Education of Czech Republic as project LN00A056.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 274–286, 2004.
© Springer-Verlag Berlin Heidelberg 2004

be extended to group colorings [7], some cannot. However, the concept of group coloring is closer to the concept of list-coloring [6]: The group chromatic number of a graph G with the average degree d is at least $\Omega(d/\log d)$. Hence, as in the concept of list coloring, the group chromatic number of graphs with large average degree is large. Lai and Zhang [8] proved that the group chromatic number of planar graphs is bounded by five. The first author with Pangrác and Voss [6] constructed a planar graph with the group chromatic number five. This is the same bound as in the case of list-coloring [12]. Planar graphs without 3-cycles and 4-cycles have the group chromatic number at most three and planar graphs without 4-cycles at most four [6] matching the same bounds for choosability [9, 11]. On the other hand, the choice number of planar bipartite graphs is at most three [1] but there is a planar bipartite graph with the group chromatic number equal to four [6].

In this paper, we address the complexity of group coloring. The problem is actually treated in a more general setting of list group coloring. A graph G is A-ℓ-choosable, $1 \leq \ell \leq |A|$, if it is A-colorable for any list-assignment L with $|L(v)| = \ell$ for every vertex v. We show that the problem to decide whether a given graph G is A-ℓ-choosable is Π_2^P-complete (under polynomial-time many-to-one reductions) if $\ell \geq 3$ (regardless the structure of A) and it is polynomial-time solvable otherwise. In particular, it is Π_2^P-complete to decide whether a given graph G is A-colorable for each Abelian group A of order at least three. Since our reduction does not rely on the structure of the group A, we can infer that the problem to decide whether the group chromatic number of a graph G is at most k is Π_2^P-complete for each $k \geq 3$.

2 Definitions and Notation

All groups considered in this paper are finite and Abelian. The neutral element is always denoted by 0. Elements of the group are sometimes also referred to as colors. If an orientation and an edge-labeling φ of a graph G are given, then a vertex coloring c is *proper* if $c(v) - c(u) \neq \varphi(uv)$ for every oriented edge uv of G. In particular, a graph G is A-colorable if there is a proper coloring for each orientation of it and each edge-labeling $\varphi : E(G) \to A$. An edge uv with $c(v) - c(u) \neq \varphi(uv)$ is said to be *properly* colored. On the other hand, if $c(v) - c(u) = \varphi(uv)$, the colors of u and v *conflict* with each other.

We first show that a variant of group coloring where sizes of lists for the vertices are part of the input is Π_2^P-complete and then we derive the Π_2^P-completeness of A-ℓ-choosability. A *list-size-assigning* function is a mapping $\psi : V(G) \to \{1, \ldots, |A|\}$. An *$A$-list-$\psi$-assignment* for a graph G and an Abelian group A is a mapping $L : V(G) \to 2^A$ with $|L(v)| = \psi(v)$ for every vertex $v \in V(G)$. If the group A is clear from the context, the list assignment L is just said to be a *list-ψ-assignment*. The graph G is said to be *A-ψ-choosable* if for each orientation of G, each edge-labeling $\varphi : E(G) \to A$ and each list-ψ-assignment L, there is a vertex coloring $c : V(G) \to A$ such that $c(v) \in L(v)$ for every vertex $v \in L(v)$ and $c(w) - c(v) \neq \varphi(vw)$ for every oriented edge

$vw \in E(G)$. We also say that G can be colored from the lists L or that it is *A-L-colorable* in such a situation. If ψ is a constant function equal to ℓ, then the list assignment L is said to be *list-ℓ-assignment* (similarly, the graph G is said to be *A-ℓ-choosable*). Note that a graph is A-$|A|$-choosable iff it is A-colorable. Group choosability has sometimes a somewhat strange behavior, e.g., a cycle of even length is A-2-choosable for every group A of odd order and for no group A of even order.

The complexity class Π_2^P is defined to be the class of the complements of problems which can be solved by non-deterministic algorithms running in polynomial time which have access to the oracle solving problems from the class NP [2,10]. Only very few natural combinatorial problems are known to be Π_2^P-complete, e.g., the problem to decide whether a graph is 3-choosable is Π_2^P-complete [3], to compute the generalized Ramsey number [2]. Similarly as ordinary satisfiability of formulas is the basic NP-complete problem, the decision problem whether a given Π_2-formula is true is the basic Π_2^P-complete problem. A formula Ψ is a *Π_2-3CNF-formula* if it is of the following form:

$$\Psi = \forall x_1 \cdots \forall x_m \exists y_1 \cdots \exists y_n \Psi_0(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

where Ψ_0 is a 3CNF-formula with variables $x_1, \ldots, x_m, y_1, \ldots, y_n$, i.e., a formula Ψ_0 is in the conjunctive normal form (CNF) and each clause of it has size exactly three. The *size* of a Π_2-3CNF-formula is the number of its clauses. We remind that occurrences of variables in clauses of Ψ_0 are called *literals*. A literal is *positive* if it is of the form x_i, $i = 1, \ldots, m$, or y_i, $i = 1, \ldots, n$. Similarly, a literal is *negative* if it is of the form $\neg x_i$, $i = 1, \ldots, m$, or $\neg y_i$, $i = 1, \ldots, n$.

3 Polynomial Cases

Clearly, a graph G is A-1-choosable iff it contains no edge. In the rest of this section, we characterize graphs G which are A-2-choosable. The discussion is split into two lemmas whose proofs are omitted due to space limitations:

Lemma 1. *Let A be a fixed Abelian group of even order. A graph G is A-2-choosable if and only if it is a forest.*

A core of a graph G as defined by Erdős et. al. [3] is the subgraph of G obtained from G by sequential removing vertices of degree one.

Lemma 2. *Let A be a fixed Abelian group of odd order $k \geq 3$. A graph G is A-2-choosable if and only if its core is a disjoint union of even cycles.*

The next theorem straightforwardly follows from Lemmas 1 and 2:

Theorem 1. *Let A be a fixed Abelian group. There exists a polynomial time algorithm which decides whether a given graph G is A-2-choosable.*

4 Reduction if the List Sizes Form Part of Input

We aim to construct a graph G and a list-size-assigning function ψ for a given Π_2-3CNF-formula Ψ such that G is A-ψ-choosable for each group A of order at least three if Ψ is true, and G is A-ψ-choosable for no group A of order at least three if Ψ is false. Moreover, the function ψ will take values only from the set $\{2,3\}$. In particular, this shows that A-choosability is Π_2^P-complete if sizes of lists for vertices form a part of the input. In the next section, these results are used to deduce that the decision problem whether a given graph is A-ℓ-choosable is Π_2^P-complete for all ℓ, $3 \leq \ell \leq |A|$.

The graph G will be glued from several types of gadgets. The *gadget* is a graph with several (usually two) distinguished *contact vertices* together with a list-size-assigning function defined for all vertices including the contact ones. Contact vertices are always non-adjacent. The vertices of a gadget which are not contact are called *inner vertices* of the gadget.

For an Abelian group A, an orientation of the edges of the gadget, an edge-labeling and a list-assignment for inner vertices, there exist one or more proper vertex-colorings of the gadget. Such a proper coloring is said to be *consistent* with the given orientation, the edge-labeling and the list-assignment. If the orientation, the edge-labeling and the list-assignment are clear from the context, the vertex-coloring is just said to be consistent. A contact vertex is *precolored* if it is colored with a color from its list. If all but one contact vertices are precolored, the remaining contact vertex is said to be *rigid* if only a single color from its list can be used to color it, i.e., every consistent coloring which extends the precoloring assigns this vertex the same color. The contact vertex is said to be *free* if there are at least two colors in its list with which it can be colored. The contact vertex is called *colorable* if it is rigid or free. In our construction, free vertices correspond to true literals and rigid vertices to false literals.

4.1 Transfer Gadget

First, we construct a so-called *transfer gadget* which is used to transfer the truth value of a literal throughout the constructed graph. The gadget can be found in Figure 1. It has two contact vertices IN and OUT. The list-size-assigning function ψ is equal to 2 for the vertex δ and to 3 for the remaining vertices. In the rest the transfer gadget is denoted by a bold double arrow joining the vertex IN to the vertex OUT (as in Figure 2). We describe some properties of the transfer gadget essential for our construction in the following propositions:

Proposition 1. *Let A be an Abelian group of order $k \geq 3$, $\rho \in A$ and $B \subseteq A$ with $|B| = 3$ and $0 \in B$. There exist an orientation, an edge-labeling and a list-assignment L of the transfer gadget with $L(\mathrm{OUT}) = B$ such that if the contact vertex IN is precolored with ρ, then the contact vertex OUT is rigid and it must be colored with* 0.

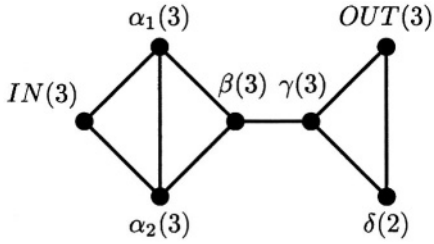


Fig. 1. The transfer gadget (the sizes of lists are represented by the numbers in the parenthesis).

Proposition 2. *Let A be an Abelian group of order $k \geq 3$. The following holds for each orientation, each edge-labeling and each list-assignment of the transfer gadget: If the contact vertex OUT is precolored, then the vertex IN is free.*

Proposition 3. *Let A be an Abelian group of order $k \geq 3$. The following holds for each orientation, each edge-labeling and each list-assignment of the transfer gadget: If the contact vertex IN is precolored, the vertex OUT is colorable.*

Proposition 4. *Let A be an Abelian group of order $k \geq 3$. Fix an orientation, an edge-labeling and a list-assignment of the transfer gadget. There is at most one color ρ in the list of the vertex IN such that if the vertex IN is precolored with ρ, the vertex OUT is rigid. In particular, for any other color $\rho' \neq \rho$ from the list of the vertex IN, if IN is precolored with ρ', the vertex OUT is free.*

4.2 Existential Gadget

The *existential gadget* is obtained from two transfer gadgets by identifying their IN vertices. The vertex obtained by identifying these two vertices is called the *middle vertex* of the existential gadget. The list-size-assigning function of the gadgets is modified by assigning the middle vertex two instead of three (the remaining vertices preserve their original sizes of lists). The contact vertices of the existential gadgets are the two OUT vertices of the transfer gadgets and they are denoted by T and F. The existential gadget can be found in Figure 2. The following two propositions summarize properties of the existential gadget which will be needed later:

Proposition 5. *Let A be an Abelian group of order $k \geq 3$. The following holds for each orientation, each edge-labeling, each list-assignment L of the existential gadget: There are colors $\alpha_F \in L(F)$ and $\alpha_T \in L(T)$ such that if F is precolored with α_F, then the vertex T is free and, on the other hand, if T is precolored with α_T, then the vertex F is free.*

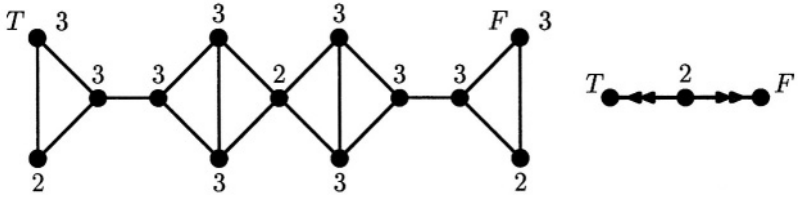


Fig. 2. The existential gadget. The numbers represent sizes of lists and the letters their names. The gadget is in the left part of the figure and its simplified diagram is in the right part.

Proposition 6. *Let A be an Abelian group of order $k \geq 3$. There exist an orientation, an edge-labeling and a list-assignment of the existential gadget such that in every consistent coloring one of the vertices T and F is rigid and forced to be colored with* 0.

4.3 Universal Gadget

We construct a so-called *universal gadget* in this subsection. The gadget is obtained by identifying the vertices IN of eight copies of the transfer gadget to a single vertex ε. Let α_T, β_T, γ_T, δ_T, α_F, β_F, γ_F and δ_F be the contact vertices OUT of the transfer gadgets. Introduce two new vertices T and F. Join the vertex T by an edge to each of the vertices α_T, β_T, γ_T and δ_T and the vertex F to each of the vertices α_F, β_F, γ_F and δ_F. The contact vertices of the universal gadget are the two vertices T and F. Both T and F are assigned three by the list-size-assigning function while the other vertices preserve their original values. The universal gadget is depicted in Figure 3. We now state two properties of the universal gadget needed in the rest:

Proposition 7. *Let A be an Abelian group of order $k \geq 3$. Fix an orientation, an edge-labeling and a list-assignment of the universal gadget. It is possible to precolor the vertex T by a color from its list so that the vertex F is free or it is possible to precolor the vertex F by a color from its list so that the vertex T is free.*

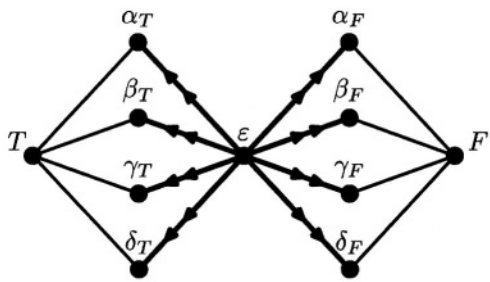


Fig. 3. The universal gadget. The letters represent the names of the vertices.

Proposition 8. *Let A be an Abelian group of order $k \geq 3$. There exist an orientation, an edge-labeling and a list-assignment of the universal gadget such that the vertex T is rigid and forced to be colored with* 0. *Similarly, there exist an orientation, an edge-labeling and a list-assignment of the universal gadget such that the vertex F is rigid and forced to be colored with* 0.

4.4 Literal Gadget

The *ℓ-literal gadget* is formed by ℓ copies of the transfer gadget and ℓ new vertices denoted by $\mathrm{OUT}_1, \ldots, \mathrm{OUT}_\ell$. Let α_i and β_i be the vertices IN and OUT of the i-th copy of the transfer gadget. The ℓ-literal gadget has $\ell+1$ contact vertices which are the vertices $\mathrm{OUT}_1, \ldots, \mathrm{OUT}_\ell$ and the vertex α_1 denoted further as the vertex IN. In addition to the edges of the transfer gadget, the ℓ-literal gadget contains edges $\beta_i\alpha_{i+1}$ for $i = 1, \ldots, \ell-1$ and edges $\beta_i\mathrm{OUT}_i$ for $i = 1, \ldots, \ell$. The list-size-assigning function ψ is modified so that $\psi(\alpha_2) = \ldots = \psi(\alpha_\ell) = 2$ and extended to the new vertices $\mathrm{OUT}_1, \ldots, \mathrm{OUT}_\ell$ so that $\psi(\mathrm{OUT}_1) = \ldots = \psi(\mathrm{OUT}_\ell) = 2$. An example of the ℓ-literal gadget with list sizes for $\ell = 5$ can be found in Figure 4. Some properties of the ℓ-literal gadget which are essential for our construction are stated in the following three propositions:

Proposition 9. *Let A be an Abelian group of order $k \geq 3$ and ℓ a positive integer. There exist an orientation, an edge-labeling and a list-assignment of the ℓ-literal gadget such that if the contact vertex IN is precolored with* 0, *then all contact vertices $OUT_1, \ldots, OUT_\ell$ must be colored with* 0.

Proposition 10. *Let A be an Abelian group of order $k \geq 3$ and ℓ a positive integer. Fix an orientation, an edge-labeling and a list-assignment of the ℓ-literal gadget. If the vertices $OUT_1, \ldots, OUT_\ell$ are precolored, then the vertex IN is free.*

Proposition 11. *Let A be an Abelian group of order $k \geq 3$ and ℓ a positive integer. Fix an orientation, an edge-labeling and a list-assignment of the ℓ-literal gadget. If the vertex IN is precolored, then there exists a consistent coloring.*

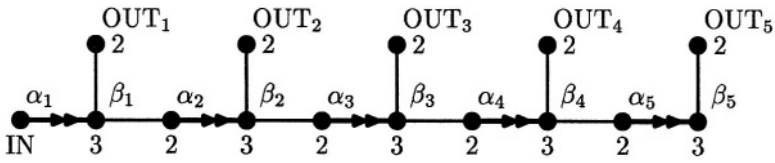


Fig. 4. The 5-literal gadget.

4.5 The Construction

Finally, we can present the construction of the desired graph G:

Theorem 2. *Let Ψ be a given Π_2-3CNF-formula. There exists a graph G with a list-size-assigning function ψ with the following property: If Ψ is true, then G is A-ψ-choosable for each Abelian group A of order at least three. If Ψ is false, then G is A-ψ-choosable for no Abelian group A of order at least three. In addition, the function ψ assigns vertices of G only the numbers two and three and the graph G and the list-size-assigning function ψ can be constructed in time polynomial in the size of the formula Ψ.*

Proof. Let $x_1, \ldots, x_m$ be the variables of Ψ which are universally quantified and $y_1, \ldots, y_n$ the variables which are existentially quantified. Let us consider a variable x_i contained in ℓ_T positive and ℓ_F negative literals in Ψ. The graph G contains a copy of the universal gadget, the ℓ_T-literal gadget and the ℓ_F-variable gadget where the contact vertex T of the universal gadget is identified with the contact vertex IN of the ℓ_T-literal gadget and the contact vertex F of the universal gadget is identified with the contact vertex IN of the ℓ_F-literal gadget. Each contact vertex OUT_j of the ℓ_T-literal gadget corresponds to one of the positive literals containing the variable x_i and each contact vertex OUT_j of the ℓ_F-literal gadget corresponds to one of the negative literals containing the variable x_i. Similarly, for a variable y_i contained in ℓ_T positive and ℓ_F negative literals, the graph G contains a copy of the existential gadget, the ℓ_T-literal gadget and the ℓ_F-literal gadget. Finally, for each clause C of Ψ, add a new vertex w_C to the graph G and join it to the three vertices corresponding to the literals contained in the clause C. This finishes the construction of G. The list-size-assigning function ψ is the same as the list-size-assigning function for the gadgets and it is extended to the clause vertices by assigning all of them the number three. Clearly, both the graph G and ψ can be constructed in time polynomial in the size of Ψ and the function ψ takes values only two and three.

Let A be an Abelian group of order at least three. First, we show that if Ψ is true, then G is A-ψ-choosable. Fix an orientation, the edge-labeling and the list-ψ-assignment of the graph G. By Proposition 7, for each universal gadget, it is possible to precolor the vertex T of it so the vertex F is free or it is possible to precolor the vertex F so the vertex T is free. In the first case, precolor the vertex T and set the corresponding variable x_i to be false. In the latter case, precolor the vertex F and set the corresponding variable x_i to be true. Since Ψ is true, there exists a truth-assignment for variables y_i such that each clause of Ψ is satisfied. Fix such a truth assignment for variables y_i. If y_i is true, precolor the vertex F of the existential gadget corresponding to the variable y_i so that its vertex T is free. If y_i is false, precolor the vertex T so that the vertex F is free. This is always possible by Proposition 5.

If the vertex IN of the literal gadget is now precolored, then extend the coloring to the whole literal gadget and color all its vertices OUT_i. This is possible by Proposition 11. In this way, contact vertices of exactly half of the literal gadgets (those corresponding to false literals) are precolored. We color

now the clause vertices. Let us consider a clause C of the formula Ψ. The size of the list of the vertex w_C is three and it has at most two precolored neighbors (otherwise all the three literals contained in the clause C are false). Hence, it is possible to color the vertex w_C by a color from its list. Once, the vertex w_C is colored, color also its neighbors corresponding to true literals by colors from their lists (this is possible since each of them has a list of size two and w_C is its only colored neighbor). In this way, all the contact vertices of literal gadgets corresponding to true literals are precolored. By Proposition 10, the vertices IN of these variable gadgets are free and since they are also free in their universal/existential gadgets and their lists have sizes three, they can be colored so that the coloring can be extended to inner vertices of the incident gadgets. In this way, we have obtained a proper coloring of the whole graph G.

Next, we show that if Ψ is false, then G is not A-ψ-choosable. By our assumption, there exists a truth assignment for $x_1, \ldots, x_m$ such that for any truth assignment for $y_1, \ldots, y_n$, there is a clause C of Ψ which is not satisfied. Consider the orientation, the edge-labeling and the list-assignment for existential gadgets, universal gadgets and literal gadgets contained in the graph G as in Propositions 6, 8 and 9. In the case of universal gadgets, choose the orientation, the edge-labeling and the list-assignment such that the color of the vertex T (F) is forced to be 0 if the corresponding variable x_i is false (true). At this moment, the only non-oriented edges and the edges without a label are the edges incident with the clause vertices. Let σ_1, σ_2 and σ_3 be any three distinct elements of A. Orient all these edges in the direction to the clause vertices and extend the edge-labeling ψ so that each clause is incident with exactly one edge labelled with σ_1, one edge with σ_2 and one edge with σ_3. Finally, assign all the clause vertices the same list $\{\sigma_1, \sigma_2, \sigma_3\}$. We show that G cannot be colored with respect to this orientation, this edge-labeling and this list-assignment.

Assume that G can be properly colored and fix such a coloring. By Proposition 6, at least one of the vertices F and T of each existential gadget is colored with 0. Set y_i to be true if the vertex F of the corresponding existential gadget is colored with 0 and set y_i to be false otherwise. Since the formula Ψ is false, there is a clause C for the just constructed truth-assignment which is not satisfied. Observe that all the three vertices OUT_j from literal gadgets which are joined to w_C must be colored with 0 by Proposition 9 since the vertices IN of those three literal gadgets are colored with 0 (this is immediately implied by the choice of the truth assignment and the fact that C is not satisfied). But then, the vertex w_C cannot be colored from its list because all the three elements σ_1, σ_2 and σ_3 appear on edges incident with it.

5 Group Coloring and List Group Coloring Are Hard

In this section, we prove our main results on hardness of the group coloring and the list group coloring. First, we state a simple lemma about list colorings of a certain special graph. The graph K_n^l for $n \geq 2$ and $l \geq 1$ is the graph obtained as follows: Consider l copies of the complete graph K_n, and choose in each copy a

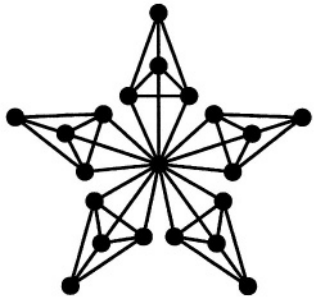


Fig. 5. The graph K_4^5.

single special vertex. Add to the graph a new vertex and join it to all the vertices of the complete graphs except for the special ones. This finishes the construction of K_n^l. The graph K_4^5 can be found in Figure 5.

Lemma 3. *Let $n \geq 2$ and l, $1 \leq l \leq n$, be fixed integers and let A be an Abelian group of order n. The graph K_n^l is A-n-choosable and there exist an orientation, an edge-labeling and a an list-n-assignment such that in each proper coloring, all the special vertices get the same color.*

We are now ready to prove our main hardness result:

Theorem 3. *Let $\ell \geq 3$ be an integer and Ψ a Π_2-3CNF-formula. There exists a ℓ-degenerate graph G_0 with the following property: If Ψ is true, then G_0 is A-ℓ-choosable for each Abelian group of order at least ℓ, and if Ψ is false, then G_0 is A-ℓ-choosable for no Abelian group of order at least ℓ. In addition, the graph G_0 can be constructed in time polynomial in ℓ and in the size of the formula Ψ.*

Proof. Let G be the graph and ψ the list-size-assigning function from Theorem 2 for the formula Ψ. Note that $\psi(v) \in \{2, 3\}$ for each vertex v of G. Let $G_1, \ldots, G_\ell$ be ℓ disjoint copies of G. Consider now a copy of the graph $K_\ell^{\ell-2}$ and let $w_1, \ldots, w_{\ell-2}$ be its $\ell - 2$ special vertices. Join each vertex $v \in V(G_i)$, $i = 1, \ldots, \ell$, with $\psi(v) = 2$ to the vertices $w_1, \ldots, w_{\ell-2}$ and each vertex v with $\psi(v) = 3$ to the vertices $w_1, \ldots, w_{\ell-3}$. Note that if $\ell = 3$, then no vertex v with $\psi(v) = 3$ is joined to any vertex of $K_\ell^{\ell-2}$. Let G_0 be the resulting graph.

We claim that G_0 has all the properties required in the statement of the lemma. It is straightforward to verify that G_0 is ℓ-degenerate. It remains to show the following: if Ψ is true, then G_0 is A-ℓ-choosable for each Abelian group of order at least three, and if Ψ is false, then G_0 is A-ℓ-choosable for no Abelian group of order at least three. Fix an Abelian group A of order at least three for the rest of the proof.

Assume first that Ψ is true. Fix an orientation, an edge-labeling and a list-ℓ-assignment L of the graph G_0. Since $K_\ell^{\ell-2}$ is A-ℓ-choosable, it is possible to color all the vertices of $K_\ell^{\ell-2}$. Consider the following list-assignments $L_1, \ldots, L_\ell$ for $G_1, \ldots, G_\ell$, respectively: The list $L_i(v) \subseteq L(v)$ of a vertex v of G_i consists of

those elements of $L(v)$ which do not conflict with any of the colors of the special vertices $w_1, \ldots, w_{\ell-2}$. Observe that $|L(v)| \geq \psi(v)$ for every $v \in V(G_i)$ where ψ is the list-size-assigning function for G. Consider the restriction of the orientation and the edge-labeling of G_0 to the graph G_i together with the list-assignment L_i. The graph G_i has a proper coloring with respect to this orientation, this edge-labeling and the list-assignment L_i because it is A-ψ-choosable by Theorem 2. This proper coloring and the proper coloring of the copy of $K_\ell^{\ell-2}$ form a proper coloring of the graph G_0.

Assume now that the formula Ψ is false. Fix an orientation, an edge-labeling and a list-ψ-assignment L_i of G_i for which G_i cannot be properly colored (they exist by Theorem 2). Enlarge each list $L_i(v)$ to $L_i'(v)$ by adding $\ell - \psi(v)$ arbitrary elements of A. Next, fix an orientation, an edge-labeling and a list-ℓ-assignment L_K of $K_\ell^{\ell-2}$ such that in any proper coloring all the special vertices get the same color (such an orientation, an edge-labeling and a list-ℓ-assignment exist by Lemma 3). Moreover, we can assume that L_K assigns each vertex of $K_\ell^{\ell-2}$ the same set $B = \{\rho_1, \ldots, \rho_\ell\}$, $B \subseteq A$.

Extend the orientations to G_0 so that each edge incident with a special vertex is oriented from the special vertex to its neighbor in G_i, $1 \leq i \leq \ell$. The edge-labeling is extended so that the labels of the edges $w_1 v, \ldots, w_{\ell-\psi(v)} v$ are precisely the differences $\alpha - \rho_i$ where α ranges through $L_i'(v) \setminus L_i(v)$. Assume for the sake of contradiction that G_0 can be colored with respect to this orientation, this edge-labeling and the constructed list-ℓ-assignment. Fix a proper coloring c. By Lemma 3, all the vertices $w_1, \ldots, w_{\ell-2}$ receive the same color ρ_i for some $i = 1, \ldots, \ell$. Then, the coloring c restricted to the graph G_i is a proper coloring of G_i such that $c(v) \in L_i(v)$ because at each vertex v the colors conflicting with the colors of the vertices $w_1, \ldots, w_{\ell-2}$ are exactly the colors of $L_i'(v) \setminus L_i(v)$. But there is no proper coloring of G_i from the lists L_i — a contradiction.

Note that since the graph G_0 constructed in Theorem 3 is ℓ-degenerate, it is A-$(\ell+1)$-choosable for any Abelian group A of order greater than ℓ.

The next three corollaries on the complexity of list group colorings and group colorings which straightforwardly follow from Theorem 3:

Corollary 1. *The decision problem of whether a given graph G is A-ℓ-choosable for a fixed Abelian group A of order $|A| \geq \ell \geq 3$ is Π_2^P-complete.*

Corollary 2. *The decision problem of whether a given graph G is A-colorable for a fixed Abelian group A of order $k \geq 3$ is Π_2^P-complete.*

Corollary 3. *The decision problem of whether the group chromatic number of a given graph G is at most k is Π_2^P-complete for each integer $k \geq 3$.*

6 Conclusion

We have investigated the complexity of list group colorings. As in the case of ordinary list colorings, one may ask about the complexity when restricted to

special classes of graphs. Gutner [4] proved that the decision problem of whether a given planar graph is 4-choosable is Π_2^P-complete. The same result holds for 3-choosability of planar triangle-free graphs. It seems natural to ask whether a similar statement holds for list group colorings. We are able to extend the technique presented in this paper only to some special cases, in particular, we can prove the following:

- If A is an Abelian group of even order, then the decision problem whether a given planar graph is A-3-choosable is Π_2^P-complete.
- If A is an Abelian group whose order is not divisible by three, then the decision problem whether a given planar graph is A-3-choosable is Π_2^P-complete.

We conjecture that this problem is Π_2^P-complete for all Abelian groups A:

Conjecture 1. For each Abelian group A of order at least three, the problem to decide whether a given planar graph is A-3-choosable is Π_2^P-complete.

Acknowledgement. The authors would like to thank Jan Kratochvíl and Jiří Sgall for discussions and helpful comments on the complexity of graph choosability and group coloring. This research was started by the first author during the Dagstuhl seminar 03391 on Graph Colorings in September 2003; he would like to thank the International Conference and Research Center Schloss Dagstuhl for support from the program High Level Scientific Conferences of the European Union.

References

1. N. Alon, M. Tarsi: Colorings and Orientations of Graphs, Combinatorica **12** (1992) 125–134.
2. D.-Z. Du, K.-I. Ko: Theory of Computational Complexity, John Wiley & Sons, New York (2000).
3. P. Erdős, A. L. Rubin, H. Taylor: Choosability in Graphs, Congress. Numer. **26** (1980) 122–157.
4. S. Gutner: The Complexity of Planar Graph Choosability, Discrete Math. **159** (1996) 119–130.
5. F. Jaeger, N. Linial, C. Payan, M. Tarsi: Group Connectivity of Graphs — A Non-homogeneous Analogue of Nowhere-zero Flow, J. Combin. Theory Ser. B **56** (1992) 165–182.
6. D. Král', O. Pangrác, H.-J. Voss: A Note on Group Colorings, submitted.
7. H.-J. Lai, X. Zhang: Group Colorability of Graphs, Ars Combin. **62** (2002) 299–317.
8. H.-J. Lai, X. Zhang: Group Chromatic Number of Graphs without K_5-minors, Graphs Comb. **18** (2002) 147–154.
9. P. C. B. Lam: The 4-choosability of Plane Graphs without 4-cycles, J. Combin. Theory Ser. B **76** (1999) 117–126.
10. C. H. Papadimitriou: Computational Complexity, Addison-Wesley, Reading (1994).
11. C. Thomassen: 3-list-coloring Planar Graphs of Girth 5, J. Combin. Theory Ser. B **64** (1995) 101–107.

12. C. Thomassen: Every Planar Graph is 5-choosable, J. Combin. Theory Ser. B **62** (1994) 180–181.
13. C. Q. Zhang: Integer Flows and Cycle Covers of Graphs, Marcel Dekker, New York (1996).

Complexity Results in Graph Reconstruction*

Edith Hemaspaandra[1], Lane A. Hemaspaandra[2], Stanisław Radziszowski[1], and Rahul Tripathi[2]

[1] Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA,
{eh,spr}@cs.rit.edu
[2] Department of Computer Science, University of Rochester, Rochester, NY 14627, USA,
{lane,rahult}@cs.rochester.edu

Abstract. We investigate the relative complexity of the graph isomorphism problem (GI) and problems related to the reconstruction of a graph from its vertex-deleted or edge-deleted subgraphs. We show that the problems are rather closely related for all amounts c of deletion:

1. For all $c \geq 1$, GI $\equiv^{l}_{iso}$ VDC_c, GI $\equiv^{l}_{iso}$ EDC_c, GI $\leq^{l}_{m}$ LVD_c, and GI $\equiv^{p}_{iso}$ LED_c.
2. For all $c \geq 1$ and $k \geq 2$, GI $\equiv^{p}_{iso}$ k-VDC_c and GI $\equiv^{p}_{iso}$ k-EDC_c.
3. For all $c \geq 1$ and $k \geq 2$, GI $\leq^{l}_{m}$ k-LVD_c. In particular, for all $c \geq 1$, GI $\equiv^{p}_{iso}$ 2-LVD_c.
4. For all $c \geq 1$ and $k \geq 2$, GI $\equiv^{p}_{iso}$ k-LED_c.

For many of these, even the $c = 1$ cases were not known.

Similar to the definition of reconstruction numbers $vrn_{\exists}(G)$ [10] and $ern_{\exists}(G)$ (see p. 120 of [17]), we introduce two new graph parameters, $vrn_{\forall}(G)$ and $ern_{\forall}(G)$, and give an example of a family $\{G_n\}_{n\geq 4}$ of graphs on n vertices for which $vrn_{\exists}(G_n) < vrn_{\forall}(G_n)$. For every $k \geq 2$ and $n \geq 1$, we show there exists a collection of k graphs on $(2^{k-1}+1)n + k$ vertices with 2^n 1-vertex-preimages, i.e., one has families of graph collections whose number of 1-vertex-preimages is huge relative to the size of the graphs involved.

1 Introduction

1.1 Background

The general form of a combinatorial reconstruction problem is the following: Given a mathematical structure $\mathcal{S}$ (e.g., a graph, a hypergraph, a characteristic polynomial of a graph, etc.) and a collection $\mathcal{D}(\mathcal{S})$ of its associated substructures (e.g., vertex-deleted subgraphs, edge-deleted subgraphs, characteristic polynomial of subgraphs, etc.), is it possible to reconstruct $\mathcal{S}$ from $\mathcal{D}(\mathcal{S})$ with some minor or no imperfections? This reconstruction problem is interesting not only

* Supported in part by grants NSF-CCR-9322513, NSF-INT-9815095/DAAD-315-PPP-gü-ab, and NSF-CCR-0311021.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 287–297, 2004.
© Springer-Verlag Berlin Heidelberg 2004

from the mathematical point of view but also for its diverse applicability in several fields. In bioinformatics, the multiple sequence alignment problem (MSA) [5] is to reconstruct a sequence with minimum gap insertion and maximum number of matching symbols, given a list of protein or DNA sequences. In computer networking, the reconstruction problem appears in the following scenario: Given a collection of sketches depicting partial network connections in a city from different locations, construct the network connection in the entire city.

In this paper, we are concerned with reconstruction problems arising in graph theory. The foremost open problems in the theory of reconstruction of graphs are the Reconstruction Conjecture and the Edge-Reconstruction Conjecture. The Reconstruction Conjecture, formulated by Kelly and Ulam in 1942 [12,27], asserts that every finite simple undirected graph on at least three vertices is determined uniquely (up to isomorphism—we treat our graphs broadly as unlabeled) by its collection of 1-vertex-deleted subgraphs. Harary [7] formulated the Edge-Reconstruction Conjecture, which states that a finite simple graph with at least four edges can be reconstructed from its collection of 1-edge-deleted subgraphs. For more on these conjectures, the reader can refer to a number of survey papers (see, for instance, [1,24,21,3]) and the book [17].

Nash-Williams [24] posed an interesting computational problem: Given a collection of graphs, how can we decide whether this has been generated from some graph by deleting one vertex every possible way, i.e., whether the collection is *legitimate*? A similar problem has been posed where we ask whether the collection is generated from some graph by deleting one edge every possible way. These problems are known as the *Legitimate Vertex-Deck Problem* (LVD) and the *Legitimate Edge-Deck Problem* (LED), respectively. Other, seemingly easier, problems are the *Vertex-Deck Checking Problem* (VDC) and the *Edge-Deck Checking Problem* (EDC) where, given a graph G and a collection $\mathcal{D}$ of graphs, we ask whether $\mathcal{D}$ can be generated from G by deleting one vertex, respectively one edge, every possible way.

Mansfield [20] and Kratsch and Hemaspaandra [14] studied complexity aspects of legitimate deck problems and deck checking problems. Kratsch and Hemaspaandra [14] showed that LVD is logspace many-one hard for the Graph Isomorphism problem (this extends the earlier result of Mansfield [20], of which [14] were unaware, that polynomial-time many-one hardness holds). Mansfield [20] showed that LED is polynomial-time Turing equivalent to the Graph Isomorphism problem (GI). Kratsch and Hemaspaandra [14] proved that GI is logspace isomorphic to VDC and obtained polynomial-time algorithms for LVD when restricted to certain classes of graphs—including graphs of bounded degree, partial k-trees for any fixed k, and graphs of bounded genus. Köbler, Schöning, and Torán [15] showed that if the Reconstruction Conjecture holds then LVD is in the class LWPP [6]. Thus, conditional on the truth of the Reconstruction Conjecture, they showed that LVD is low for PP, i.e., $\mathrm{PP}^{\mathrm{LVD}} = \mathrm{PP}$. This result can be viewed as suggesting that LVD cannot be NP-complete, since if it were NP-complete then the result of [15] would immediately imply that either the Reconstruction Conjecture fails or $\mathrm{PP}^{\mathrm{NP}} = \mathrm{PP}$. But both these claims are widely suspected to be false.

1.2 Our Contributions

A more general reconstruction problem deals with collections consisting of all subgraphs obtained through the deletion of (exactly) some fixed number $c \geq 1$ of vertices (or edges). Kelly [13] first raised the possibility of deleting several vertices from a graph, Manvel [19] made some observations on this problem, and Bondy (see Section 11.2 in [3]) surveyed results on this more general reconstruction problem. (Also, see a review [25] on the progress made on this problem in the past three decades.) In this paper, one of our investigations is of the complexity of legitimate deck problems and deck checking problems for the general case when these problems are defined in terms of deletion of some fixed number $c \geq 1$ of vertices (or edges) of a graph. We observe that the logspace isomorphism between GI and VDC [14] holds, for every $c \geq 1$, between GI and VDC_c and between GI and EDC_c. (Here and henceforth, the subscript "c" in the name of a problem refers to the more general problem based on the deletion of c vertices or edges of a graph.) We strengthen the result of [20] to show that, for every $c \geq 1$, GI is, in fact, polynomial-time isomorphic to LED_c. For LVD_c, we observe that for every $c \geq 1$, $\mathrm{GI} \leq_m^p \mathrm{LVD}_c$ (the $c = 1$ case of this already follows from [14]).

We next look at the question of reconstructing a graph from a subdeck (a subset of all possible vertex-deleted or edge-deleted subgraphs). See [9,2,16] for this line of investigation in the reconstruction of trees. Our results on the complexity aspects of the reconstruction of a graph from a subdeck are described in Section 3.2. Again, we obtain a strong relationship between these problems and the graph isomorphism problem. Harary and Plantholt [10] introduced a parameter, called the ally-reconstruction number $vrn_\exists(G)$ of a graph G, and defined it as the minimum number of 1-vertex-deleted subgraphs needed to identify G (as always, up to isomorphism). A similar definition is used for the reconstruction number $ern_\exists(G)$, which is defined in terms of 1-edge-deleted subgraphs (see p. 120 of [17]). We introduce two new parameters, $vrn_\forall(G)$ and $ern_\forall(G)$, for a graph G and give an example of a family $\{G_n\}_{n \geq 4}$ of graphs on n vertices for which $vrn_\exists(G_n) < vrn_\forall(G_n)$. We also give a family of collections of k graphs on $(2^{k-1} + 1)n + k$ vertices with 2^n 1-vertex-preimages, thus constructing an exponential richness of number of preimages. *(Due to space limitations, in this version most proofs are omitted; please refer to the (in-preparation) full version.)*

2 Preliminaries

2.1 Notation

Our alphabet is $\Sigma = \{0, 1\}$. Let $[.,\ldots,.]$ denote multisets. We use $\cup$ to denote set union as well as multiset union. Let $\langle \ldots \rangle$ be a multi-arity, polynomial-time computable and polynomial-time invertible pairing function. We tacitly assume that multisets and graphs are encoded in a standard fashion. For background in complexity theory and for notions such as P, NP, reductions and completeness, we refer the reader to any textbook on complexity theory [8]. We consider only finite, undirected graphs with no self-loops. Given a graph G, let $V(G)$ denote

the vertex set and $E(G)$ denote the edge set of G. For notational convenience, we alternatively represent a graph G by (V, E) where $V = V(G)$ and $E = E(G)$. The degree of a vertex v in G, denoted by $deg_G(v)$, is the number of edges incident on v. $\delta(G) = \min\{deg_G(v) \mid v \in V(G)\}$ and $\lambda(G)$ is the minimum number of edges whose deletion from G disconnects G. The neighborhood $N_G(v)$ of a vertex v in a graph G is the set of vertices that are at a distance at most one from v, that is, $N_G(v) = \{v\} \cup \{w \mid \{v, w\} \in E(G)\}$. The union of graphs $G_1, G_2, \ldots, G_k$, $k \geq 2$, is denoted by $G = G_1 \cup G_2 \cup \cdots \cup G_k$ where $V(G) = \bigcup_{i=1}^{k} V(G_i)$ and $E(G) = \bigcup_{i=1}^{k} E(G_i)$. For a graph G, and an integer $m > 0$, mG represents the union of m vertex-disjoint (isomorphic) copies of G. The join of graphs $G_1, \ldots, G_k$, $k \geq 2$, is denoted by $G = G_1 + \ldots + G_k$ where $V(G) = \bigcup_{i=1}^{k} V(G_i)$ and $E(G) = \bigcup_{i=1}^{k} E(G_i) \cup \bigcup_{i \neq j} \{\{u, v\} \mid u \in V(G_i) \wedge v \in V(G_j)\}$. The notions of union and join of graphs will always require disjoint sets of vertices and thus for graphs G and H with $V(G) \cap V(H) \neq \emptyset$, we assume that isomorphs $\hat{G}$ and $\hat{H}$ of G and H, respectively, with $V(\hat{G}) \cap V(\hat{H}) = \emptyset$, are used in place of G and H. For $n \geq 1$, K_n is the complete graph and P_n is the path graph on n vertices. That is, $(V(P_n), E(P_n)) = (\{1, \ldots, n\}, \{\{i, i+1\} \mid 1 \leq i \leq n-1\})$. The line graph $L(G)$ of a graph G is defined by: $V(L(G)) = E(G)$ and $E(L(G)) = \{\{e_1, e_2\} \mid e_1, e_2 \in E(G) \wedge e_1$ and e_2 have exactly one vertex in common$\}$.

Given a graph G and a set $S \subseteq V(G)$, G–S denotes a graph with $V(G-S) = V(G) - S$ and $E(G - S) = E(G) - \{\{u, v\} \mid \{u, v\} \in E(G) \wedge \{u, v\} \cap S \neq \emptyset\}$. Similarly, if $S \subseteq E(G)$, then $G - S$ denotes a graph with $V(G - S) = V(G)$ and $E(G - S) = E(G) - S$. We will call any collection of graphs with an identical number of vertices a "vertex-deck" and will use the term "edge-deck" to denote a collection of graphs with identical numbers of edges. The graphs in a vertex-deck are called vertex-cards and the graphs in an edge-deck are called edge-cards. For a graph G and for any $c \geq 1$, the c-vertex-deleted-deck of G, denoted by vertex-deck$_c(G)$, is the multiset $[G - S \mid S \subseteq V(G)$ and $||S|| = c]$, and the c-edge-deleted-deck of G, denoted by edge-deck$_c(G)$, is the multiset $[G - S \mid S \subseteq E(G)$ and $||S|| = c]$. We say that a vertex-deck $D_1 = [G_1, \ldots, G_n]$ is equivalent to a vertex-deck $D_2 = [G'_1, \ldots, G'_n]$, denoted by $D_1 = D_2$, if there exists a one-one mapping that maps each graph from D_1 to an isomorphic graph from D_2. We use a similar definition for the equivalence of two edge-decks: An edge-deck $D_1 = [G_1, \ldots, G_m]$ is equivalent to an edge-deck $D_2 = [G'_1, \ldots, G'_m]$, denoted by $D_1 = D_2$, if there exists a one-one mapping that maps each graph from D_1 to an isomorphic graph from D_2. The notion of $\mathcal{D}_1 \subseteq \mathcal{D}_2$ is defined analogously. For any $c \geq 1$, we say a graph G is a c-vertex-preimage of $[G_1, \ldots, G_k]$ if $[G_1, \ldots, G_k] \subseteq$ vertex-deck$_c(G)$, and say a graph G is a c-edge-preimage of $[G_1, \ldots, G_k]$ if $[G_1, \ldots, G_k] \subseteq$ edge-deck$_c(G)$. For any $c \geq 1$, we say that a graph H is a c-vertex-card (respectively, c-edge-card) of a graph G if $H \in$ vertex-deck$_c(G)$ (respectively, $H \in$ edge-deck$_c(G)$). $[G_1, \ldots, G_k]$ is a legitimate c-vertex-deck (c-vertex-subdeck) if there is a graph G such that $[G_1, \ldots, G_k] =$ vertex-deck$_c(G)$ ($[G_1, \ldots, G_k] \subseteq$ vertex-deck$_c(G)$). The notions of legitimate c-edge-deck and legitimate c-edge-subdeck, for any $c \geq 1$, are defined in a similar way. For any graph G, the endvertex-deck of G, denoted by

endvertex-deck(G), is the multiset consisting of the subgraphs $G - v$ where v is an endvertex of G, i.e., $deg_G(v) = 1$.

2.2 Graph Isomorphism

A graph G is isomorphic to a graph H if there is a bijective mapping $\psi : V(G) \to V(H)$ such that, for all $v_1, v_2 \in V(G)$, $\{v_1, v_2\} \in E(G)$ if and only if $\{\psi(v_1), \psi(v_2)\} \in E(H)$. In this case, ψ is called an isomorphism between graphs G and H, and we write $G \cong H$. If $G \cong H$ via the identity mapping, then we use $G = H$ to represent this fact. A yes-instance $\langle G, H \rangle$ of the graph isomorphism problem (GI) is an encoding of graphs G and H where G is isomorphic to H.

Definition 1 ([15], see also [11,26,18]). *An or-function for a set A is a function f mapping sequences of strings to strings such that for every sequence $x_1, \ldots, x_n$, it holds that $f(\langle x_1, \ldots, x_n \rangle) \in A \iff (\exists i \in \{1, \ldots, n\})[x_i \in A]$. An **and**-function for a set A can be defined similarly.*

Proposition 1 ([15]). GI *has a polynomial-time computable **or**-function and a polynomial-time computable **and**-function (both of them in the sense of Definition 1).*

Throughout the paper, we use $\mathbf{or}_{\mathrm{GI}}$ to denote the **or**-function and use $\mathbf{and}_{\mathrm{GI}}$ to denote the **and**-function of GI mentioned in Proposition 1. The existence of $\mathbf{or}_{\mathrm{GI}}$ implies that if a set L disjunctive truth-table reduces to GI, then L polynomial-time many-one reduces to GI. We will use this property to obtain polynomial-time many-one reductions from certain sets to GI.

2.3 Computational Problems on the Reconstruction of Graphs

Kelly [13] first proposed the idea of generalizing the Reconstruction Conjecture to c-vertex-deleted subgraphs for $c > 1$. Kelly showed that there are graphs that are not determined uniquely (up to isomorphism) by their 2-vertex-deleted subgraphs. However, it is believed that, for any $c > 1$, all sufficiently large graphs satisfy the general reconstruction problem for c-vertex-deleted subgraphs. From a computational complexity point of view, it is interesting to analyze the complexity of problems related to the reconstruction of a graph from its c-vertex-deleted or c-edge-deleted subgraphs for different values of c. With this motivation, we define the computational problems we study in this paper.

1. VERTEX-DECK CHECKING$_c$ (abbreviated VDC$_c$)
   $\mathrm{VDC}_c = \{\langle G; [G_1, \ldots, G_n] \rangle \mid [G_1, \ldots, G_n] = \text{vertex-deck}_c(G)\}$.
2. EDGE-DECK CHECKING$_c$ (abbreviated EDC$_c$)
   $\mathrm{EDC}_c = \{\langle G; [G_1, \ldots, G_m] \rangle \mid [G_1, \ldots, G_m] = \text{edge-deck}_c(G)\}$.
3. LEGITIMATE VERTEX-DECK$_c$ (abbreviated LVD$_c$)
   $\mathrm{LVD}_c = \{\langle [G_1, \ldots, G_n] \rangle \mid (\exists G)[[G_1, \ldots, G_n] = \text{vertex-deck}_c(G)]\}$.

4. LEGITIMATE EDGE-DECK$_c$ (abbreviated LED$_c$)
LED$_c$ = $\{\langle [G_1, \ldots, G_m] \rangle \mid (\exists G)[[G_1, \ldots, G_m] = \text{edge-deck}_c(G)]\}$

For any fixed $k \geq 2$, one can study the k-vertex-(edge-)card versions of the above-mentioned problems. These problems are denoted by k-VDC$_c$, k-EDC$_c$, k-LVD$_c$ and k-LED$_c$, respectively. We give the formal definition of k-VDC$_c$; the other problems are defined analogously.

1′. k-VERTEX-SUBDECK CHECKING$_c$ (abbreviated k-VDC$_c$)
k-VDC$_c$ = $\{\langle G; [G_1, \ldots, G_k] \rangle \mid [G_1, \ldots, G_k] \subseteq \text{vertex-deck}_c(G)\}$.

3 Reconstruction from Vertex-(Edge-)Deck

3.1 Reconstruction from a Complete Deck

In this section, we investigate the complexity of VDC$_c$, EDC$_c$, LVD$_c$ and LED$_c$ for any $c \geq 1$. Kratsch and Hemaspaandra [14] showed that GI is logspace isomorphic to VDC$_1$. By generalizing their proof, we strengthen this result, and show that, for any $c \geq 1$, GI is logspace isomorphic to VDC$_c$ as well as to EDC$_c$. Theorem 1(2) will be used in showing that for each $c \geq 1$, LED$_c \leq_m^p$ GI.

Theorem 1. *(1) For all $c \geq 1$,* GI *is logspace isomorphic to* VDC$_c$. *(2) For all $c \geq 1$,* GI *is logspace isomorphic to* EDC$_c$.

Kratsch and Hemaspaandra [14] showed that GI $\leq_m^l$ LVD$_1$. We strengthen this result and show that, for any $c \geq 1$, GI $\leq_m^l$ LVD$_c$. Mansfield [20] showed that GI is polynomial-time Turing equivalent to LED$_1$. We strengthen this result and show that, for any $c \geq 1$, GI is polynomial-time isomorphic to LED$_c$.

Theorem 2. *(1) For every $c \geq 1$,* GI $\leq_m^l$ LVD$_c$. *(2) For every $c \geq 1$,* GI *is polynomial-time isomorphic to* LED$_c$.

3.2 Reconstruction from a Subdeck

We next investigate the complexity of problems related to the reconstruction of a graph from its partial (incomplete) deck of vertex-deleted or edge-deleted subgraphs. Theorem 3 states that GI is polynomial-time isomorphic to the k-card versions of VDC$_c$ and EDC$_c$ for each $c \geq 1$ and $k \geq 2$.

Theorem 3. *For every $c \geq 1$ and $k \geq 2$,* GI *is polynomial-time isomorphic to* k-VDC$_c$ *and* k-EDC$_c$.

We now consider the relative complexity of GI and k-LVD$_c$, and that of GI and k-LED$_c$, for $k \geq 2$. Lemma 1 gives an alternate characterization of an instance of 2-LVD$_c$ in terms of polynomially many instances of GI. As an immediate consequence of Lemma 1, we have that 2-LVD$_c \leq_{dtt}^p$ GI. From the explanation given in Section 2.2, it follows that 2-LVD$_c \leq_m^p$ GI.

Lemma 1. *For each $c \geq 1$, $[G_1, G_2]$ is a legitimate c-vertex-subdeck if and only if there exist $U_1 \subseteq V(G_1)$ and $U_2 \subseteq V(G_2)$, where $1 \leq ||U_1|| = ||U_2|| \leq c$, such that $G_1 - U_1$ is isomorphic to $G_2 - U_2$.*

Proof Fix a $c \geq 1$. Suppose that $[G_1, G_2]$ is a legitimate c-vertex-subdeck. By definition, there exist a graph G, distinct sets $T_1, T_2 \subseteq V(G)$, where $||T_1|| = ||T_2|| = c$, and isomorphisms ψ_1 from $G - T_1$ to G_1 and ψ_2 from $G - T_2$ to G_2. Clearly, $G_1 - \psi_1(T_2 - T_1)$ is isomorphic to $G - (T_1 \cup T_2)$ and $G_2 - \psi_2(T_1 - T_2)$ is isomorphic to $G - (T_1 \cup T_2)$. Thus, $G_1 - \psi_1(T_2 - T_1)$ is isomorphic to $G_2 - \psi_2(T_1 - T_2)$.

Now assume that there exist $U_1 = \{u_{1,1}, u_{1,2}, \ldots, u_{1,\ell}\} \subseteq V(G_1)$ and $U_2 = \{u_{2,1}, u_{2,2}, \ldots, u_{2,\ell}\} \subseteq V(G_2)$, where $1 \leq \ell \leq c$, such that $G_1 - U_1$ is isomorphic to $G_2 - U_2$ via ψ. We now construct a graph $\mathcal{G}_2$ by adding new vertices $v_{2,1}, \ldots, v_{2,c}$ in G_2 and by including new edges incident on them. The graph $\mathcal{G}_2$ is defined as follows. Initially, $\mathcal{G}_2 := G_2$. For each $1 \leq i \leq \ell$, add a vertex $v_{2,i}$ in $\mathcal{G}_2$ and connect $v_{2,i}$ to every vertex in $\psi(N_{G_1}(u_{1,i}) - U_1)$. For each $1 \leq i < j \leq \ell$, add an edge $\{v_{2,i}, v_{2,j}\}$ in $\mathcal{G}_2$ if and only if $\{u_{1,i}, u_{1,j}\} \in E(G_1)$. Finally, for each $1 \leq i \leq c - \ell$, add a new vertex $v_{2,\ell+i}$ and connect it to every other vertex in $\mathcal{G}_2$. We construct another graph $\mathcal{G}_1$ in a similar way. Initially, $\mathcal{G}_1 := G_1$. For each $1 \leq i \leq \ell$, add a vertex $v_{1,i}$ in $\mathcal{G}_1$ and connect $v_{1,i}$ to every vertex in $\psi^{-1}(N_{G_2}(u_{2,i}) - U_2)$. For each $1 \leq i < j \leq \ell$, add an edge $\{v_{1,i}, v_{1,j}\}$ in $\mathcal{G}_1$ if and only if $\{u_{2,i}, u_{2,j}\} \in E(G_2)$. Finally, for each $1 \leq i \leq c - \ell$, add a new vertex $v_{1,\ell+i}$ and connect it to every other vertex in $\mathcal{G}_1$.

Let $\psi' : V(\mathcal{G}_1) \to V(\mathcal{G}_2)$ be defined as follows: $\psi'(V(G_1 - U_1)) = \psi(V(G_1 - U_1))$, for every $1 \leq i \leq \ell$, $\psi'(u_{1,i}) = v_{2,i}$ and $\psi'(v_{1,i}) = u_{2,i}$, and for every $\ell + 1 \leq i \leq c$, $\psi'(v_{1,j}) = v_{2,j}$. It can be verified that ψ' is an isomorphism from $\mathcal{G}_1$ to $\mathcal{G}_2$. Since G_1 $(= \mathcal{G}_1 - \{v_{1,1}, \ldots, v_{1,c}\})$ is a c-vertex-card of $\mathcal{G}_1$ and G_2 $(= \mathcal{G}_2 - \{v_{2,1}, \ldots, v_{2,c}\})$ is a c-vertex-card of $\mathcal{G}_2$, and since $\{v_{2,1}, \ldots, v_{2,c}\} \neq \psi'(\{v_{1,1}, \ldots, v_{1,c}\})$, it follows that $[G_1, G_2]$ is a legitimate c-vertex-subdeck. □

Corollary 1. *For every $c \geq 1$, 2-LVD$_c$ $\leq_m^p$ GI.*

In Theorem 4(1) and Theorem 5, we obtain the polynomial-time isomorphism from GI to 2-LVD$_c$ for each $c \geq 1$, and from GI to k-LED$_c$ for each $c \geq 1$ and $k \geq 2$.

Theorem 4. *(1) For every $c \geq 1$,* GI *is polynomial-time isomorphic to* 2-LVD$_c$. *(2) For every $c \geq 1$ and $k \geq 2$,* GI $\leq_m^l$ k-LVD$_c$.

Theorem 5. *For every $c \geq 1$ and $k \geq 2$,* GI *is polynomial-time isomorphic to* k-LED$_c$.

4 Reconstruction Number of Undirected Graphs

Definition 2 ([10,22]). *The ally-reconstruction number of a graph G is the minimum number of one-vertex-deleted subgraphs (or 1-vertex-cards) that identify G (up to isomorphism).*

Since the ally-reconstruction number of a graph G is characterized by the *existence* of the same number of 1-vertex-cards of G, we will denote this number for G by $vrn_{\exists}(G)$. Likewise, we use $ern_{\exists}(G)$ to denote the minimum number of 1-edge-cards that identify G. We also define an analogous definition of reconstruction number for a graph G, denoted by $vrn_{\forall}(G)$ (respectively, $ern_{\forall}(G)$), in which a certain number of 1-vertex-cards (respectively, 1-edge-cards), irrespective of their choice, suffice to recognize G. Thus, no matter which 1-vertex-cards (respectively, 1-edge-cards) an adversary selects for a graph G, $vrn_{\forall}(G)$ (respectively, $ern_{\forall}(G)$) many 1-vertex-cards (respectively, 1-edge-cards) are enough to identify G up to isomorphism. If such a number doesn't exist, we define it to be ∞.

It is clear that for any graph G for which $vrn_{\exists}(G) < \infty$ (respectively, $ern_{\exists}(G) < \infty$), $vrn_{\exists}(G) \leq vrn_{\forall}(G) \leq ||V(G)||$ (respectively, $ern_{\exists}(G) \leq ern_{\forall}(G) \leq ||E(G)||$). Note that $vrn_{\exists}(G)$ is finite for every graph G if and only if the Reconstruction Conjecture is true, and $ern_{\exists}(G)$ is finite for every graph G if and only if the Edge-Reconstruction Conjecture is true. Theorem 6 says that for any disconnected graph G, $vrn_{\exists}(G)$ (consequently, $vrn_{\forall}(G)$) is finite.

Theorem 6 ([22]). *If G is a disconnected graph with not all components isomorphic then $vrn_{\exists}(G) = 3$. Moreover, if G is a disconnected graph with all components isomorphic then $vrn_{\exists}(G) \leq c + 2$ where c is the number of vertices in a component.*

In the next lemma, we give an example of a family of disconnected graphs G (parameterized by n, the number of vertices of the n'th graph in the family) for which $vrn_{\exists}(G) < vrn_{\forall}(G)$.

Lemma 2. *For all $n \geq 4$, there is a disconnected graph G_n such that $||V(G_n)|| = n$ and $vrn_{\exists}(G_n) < vrn_{\forall}(G_n)$.*

Proof Let $n \geq 4$. Define the ordered pair

$$(G_n, H_n) = \begin{cases} (K_{\frac{n}{2}+1} \cup K_{\frac{n}{2}-1}, 2K_{\frac{n}{2}}) & \text{if } n \text{ is even,} \\ (K_{\frac{n-1}{2}+1} \cup K_{\frac{n-1}{2}-1} \cup K_1, 2K_{\frac{n-1}{2}} \cup K_1) & \text{if } n \text{ is odd.} \end{cases}$$

By Theorem 6, $vrn_{\exists}(G_n) = 3$. It is clear that G_n and H_n are nonisomorphic graphs. For even n, both G_n and H_n have $\frac{n}{2}+1$ 1-vertex-cards that are isomorphic to $K_{\frac{n}{2}} \cup K_{\frac{n}{2}-1}$, and for odd n, both G_n and H_n have $\frac{n-1}{2}+1$ 1-vertex cards that are isomorphic to $K_{\frac{n-1}{2}} \cup K_{\frac{n-1}{2}-1} \cup K_1$. Thus, for even n, $vrn_{\forall}(G_n) \geq \frac{n}{2}+2 > vrn_{\exists}(G_n) = 3$, and for odd n, $vrn_{\forall}(G_n) \geq \frac{n-1}{2}+2 > vrn_{\exists}(G_n) = 3$. □

The Reconstruction Conjecture can be restated as follows: For each $n \geq 3$, given any collection $\mathcal{D}$ of n graphs with $n-1$ vertices in each, there can be at most one 1-vertex-preimage of $\mathcal{D}$. What can we say about the number of nonisomorphic 1-vertex-preimages of a collection $\mathcal{D}$ of graphs with $n-1$ vertices in each where the size of $\mathcal{D}$ is smaller than n? Is there a subset of vertex-deck$_1(G)$ that suffices to identify G up to isomorphism? Myrvold [23] showed that for any tree T, the number of nonisomorphic preimages of endvertex-deck(T) is exactly one; the unique preimage up to isomorphism is T itself. However, the

following theorem by Bryant [4] says that there are graphs G for which the endvertex-deck(G) has more than one nonisomorphic preimage.

Theorem 7 ([4]). *For any positive integer k, there exist nonisomorphic graphs G and H, with k endvertices in each, such that* endvertex-deck(G) = endvertex-deck(H).

Note that Theorem 7 only talks about the existence of at least two nonisomorphic 1-vertex-preimages of a certain collection consisting of k 1-vertex-cards, for every $k \geq 2$. In the next theorem, we show that there is a family of multisets of k graphs on $(2^{k-1}+1)n+k$ vertices with 2^n 1-vertex-preimages.

Theorem 8. *For all $k \geq 2$ and $n \geq 1$, there is a deck $\mathcal{D}$ of k vertex-cards on $(2^{k-1}+1)n+k$ vertices with at least 2^n 1-vertex-preimages.*

Proof Each of the k vertex-cards in $\mathcal{D}$ is identical, and defined as follows.

1. $x_0, \ldots, x_n$ are the vertices of the path graph P_{n+1}.
2. $y_1, \ldots, y_{k-1}$ are special selector vertices.
3. For $i := 1 \ldots n$,
  3.1 Let G_i be the complete graph $K_{2^{k-1}}$ and let $V(G_i) = \{z_{i,I} \mid I \subseteq \{1, \ldots, k-1\}\}$.
  3.2 Connect x_i to all the vertices of G_i.
  3.3 For each $z_{i,I} \in V(G_i)$, connect $z_{i,I}$ to each vertex y_j where $j \in I$.

Consider 1-vertex-preimages H of $\mathcal{D}$ of the following form.

1. $x_0, \ldots, x_n$ are the vertices of the path graph P_{n+1}.
2. $y_1, \ldots, y_{k-1}, y_k$ are special selector vertices.
3. For $i := 1 \ldots n$,
  3.1 Let G_i be the complete graph $K_{2^{k-1}}$ and let $V(G_i) = \{z_{i,j} \mid j \in \{1, \ldots, 2^{k-1}\}\}$.
  3.2 Connect x_i to all the vertices of G_i.
  3.3 The edges between the y-vertices and G_i are defined according to one of the following two cases.
    Case 1. Let $Y_1, \ldots, Y_{2^{k-1}}$ be an enumeration of subsets of $\{1, \ldots, k\}$ where each $||Y_j||$ is odd. For each $j \in \{1, \ldots, 2^{k-1}\}$, connect $z_{i,j}$ to each vertex y_ℓ where $\ell \in Y_j$.
    Case 2. Let $Y_1, \ldots, Y_{2^{k-1}}$ be an enumeration of subsets of $\{1, \ldots, k\}$ where each $||Y_j||$ is even. For each $j \in \{1, \ldots, 2^{k-1}\}$, connect $z_{i,j}$ to each vertex y_ℓ where $\ell \in Y_j$.

Note that H is a 1-vertex-preimage of $\mathcal{D}$, since $H - y_i$ is isomorphic to the 1-vertex-card in $\mathcal{D}$ for $1 \leq i \leq k$. As i varies from 1 to n, in step 3.3 each time we can apply either Case 1 or Case 2. And every two distinct sequences of such choices in the construction of H give rise to nonisomorphic graphs. Thus, the number of nonisomorphic 1-vertex-preimages is at least 2^n. □

5 Open Problems

In this section, we mention some open problems. Theorem 4(1) states that, for every $c \geq 1$, GI $\equiv^p_{iso}$ 2-LVD$_c$. However, for $k > 2$ and $c \geq 1$, we do not know whether k-LVD$_c$ is polynomial-time equivalent to GI or is NP-complete (or neither). Since for $k > 2$ and $c \geq 1$ it is not clear, even under the assumption that the Reconstruction Conjecture is true, whether k-LVD$_c$ is low for PP, it is at least possible that k-LVD$_c$ is NP-complete.

It is also interesting to investigate the complexity of problems related to the reconstruction numbers. For instance, we define the following list of problems:

a) EXIST-VRN$= \{\langle G, k\rangle \mid vrn_\exists(G) \leq k\}$. **b)** UNIV-VRN$= \{\langle G, k\rangle \mid vrn_\forall(G) \leq k\}$.
c) EXIST-ERN$= \{\langle G, k\rangle \mid ern_\exists(G) \leq k\}$. **d)** UNIV-ERN$= \{\langle G, k\rangle \mid ern_\forall(G) \leq k\}$.

It is easy to see that EXIST-VRN $\in \Sigma^p_2$ (since GI is low for Σ^p_2), UNIV-VRN $\in$ coNP$^{\text{GI}}$, EXIST-ERN $\in$ NP$^{\text{GI}}$, and UNIV-ERN $\in$ coNP$^{\text{GI}}$. It would be interesting to obtain tight (or tighter) bounds on the complexity of these problems. (For instance, is EXIST-VRN complete for Σ^p_2? Is UNIV-ERN coNP-hard?)

Acknowledgment. We thank the anonymous referees for helpful comments.

References

1. J. Bondy and R. Hemminger. Graph reconstruction—a survey. *Journal of Graph Theory,* 1:227–268, 1977.
2. J. Bondy. On Ulam's conjecture for separable graphs. *Pacific Journal of Mathematics,* 31:281–288, 1969.
3. J. Bondy. A graph reconstructor's manual. In *Surveys in Combinatorics,* London Mathematical Society Lecture Notes Series 66, pages 221–252. Cambridge University Press, 1991.
4. R. Bryant. On a conjecture concerning the reconstruction of graphs. *Journal of Combinatorial Theory,* 11:139–141, 1971.
5. H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics,* 48:1073–1082, 1988.
6. S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences,* 48(1):116–148, 1994.
7. F. Harary. On the reconstruction of a graph from a collection of subgraphs. In M. Fiedler, editor, *Theory of Graphs and its Applications,* pages 47–52. Czechoslovak Academy of Sciences, Prague, 1964.
8. L. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion.* Springer, 2002.
9. F. Harary and E. Palmer. The reconstruction of a tree from its maximal subtrees. *Canadian Journal of Mathematics,* 18:803–810, 1966.
10. F. Harary and M. Plantholt. The graph reconstruction number. *J. Graph Theory,* 9:451–454, 1985.
11. J. Kadin. *Restricted Turing Reducibilities and the Structure of the Polynomial Time Hierarchy.* PhD thesis, Cornell University, USA, 1988.

12. P. Kelly. *On Isometric Transformations.* PhD thesis, University of Wisconsin, USA, 1942.
13. P. Kelly. A congruence theorem for trees. *Pacific Journal of Mathematics,* 7:961–968, 1957.
14. D. Kratsch and L. Hemaspaandra. On the complexity of graph reconstruction. *Mathematical Systems Theory,* 27(3):257–273, 1994.
15. J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity.* Birkhauser, 1993.
16. J. Lauri. Proof of Harary's conjecture on the reconstruction of trees. *Discrete Mathematics,* 43:79–90, 1983.
17. J. Lauri and R. Scapellato. *Topics in Graph Automorphism and Reconstruction.* London Mathematical Society, Cambridge University Press, 2003.
18. A. Lozano and J. Torán. On the non-uniform complexity of the graph isomorphism problem. In *Proceedings of the 7th Structure in Complexity Theory Conference,* pages 118–129. IEEE Computer Society Press, June 1992.
19. B. Manvel. Some basic observations on Kelly's conjecture for graphs. *Discrete Mathematics,* 8:181–185, 1974.
20. A. Mansfield. The relationship between the computational complexities of the legitimate deck and isomorphism problems. *Quart. J. Math. Ser.,* 33(2):345–347, 1982.
21. B. Manvel. Reconstruction of graphs: Progress and prospects. *Congressus Numerantium,* 63:177–187, 1988.
22. W. Myrvold. The ally-reconstruction number of a disconnected graph. *Ars Combinat.,* 28:123–127, 1989.
23. W. Myrvold. The ally-reconstruction number of a tree with five or more vertices is three. *Journal of Graph Theory,* 14:149–166, 1990.
24. C. St. J. A. Nash-Williams. The reconstruction problem. In L. Beineke and R. Wilson, editors, *Selected Topics in Graph Theory,* pages 205–236. Academic Press, 1978.
25. V. Nýdl. Graph reconstruction from subgraphs. *Discrete Mathematics,* 235:335–341, 2001.
26. D. Ranjan and P. Rohatgi. On randomized reductions to sparse sets. In *Proceedings of the 7th Structure in Complexity Theory Conference,* pages 239–242. IEEE Computer Society Press, June 1992.
27. S. Ulam. *A Collection of Mathematical Problems.* Interscience Publishers, New York, 1960.

Generating Paths and Cuts in Multi-pole (Di)graphs*

Endre Boros[1], Khaled Elbassioni[1], Vladimir Gurvich[1], Leonid Khachiyan[2], and Kazuhisa Makino[3]

[1] RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway NJ 08854-8003, {boros,elbassio,gurvich}@rutcor.rutgers.edu

[2] Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway NJ 08854-8003, leonid@cs.rutgers.edu

[3] Division of Systems Science, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka, 560-8531, Japan, makino@sys.es.osaka-u.ac.jp

Abstract. Let $G = (V, E)$ be a (directed) graph with vertex set V and edge (arc) set E. Given a set $\mathcal{P}$ of (source-sink) pairs of vertices of G, an important problem that arises in the computation of network reliability is the enumeration of minimal subsets of edges (arcs) that connect/disconnect all/at least one of the given source-sink pairs of $\mathcal{P}$. For undirected graphs, we show that the enumeration problems for conjunctions of paths and disjunctions of cuts can be solved in incremental polynomial time. For directed graphs both of these problems are NP-hard. We also give a polynomial delay algorithm for enumerating minimal sets of arcs connecting respectively two given nodes s_1 and s_2 to a given vertex t_1, and each vertex of a given subset of vertices T_2.

1 Introduction

Let $G = (V, E)$ be a (directed) graph on vertex set V and edge (arc) set E. Let $\mathcal{P}$ be an arbitrary family of *(source-sink)* pairs of vertices of G: $\mathcal{P} = \{(s_i, t_i) \in V \times V \mid i \in [k] = \{1, \ldots, k\}\}$. We assume that $s_i \neq t_i$ for all $i \in [k]$, however sources and sinks may coincide (i.e. for $i \neq j$, we may have any of $s_i = s_j$, $t_i = t_j$ or $s_i = t_j$).

Consider the following two dual enumeration problems, stated as the generation of all minimal subsets of edges (arcs) $E' \subseteq E$ such that

Path-conjunction (PC): s_i is connected to t_i in the (di)graph $G' = (V, E')$ for all $i \in [k]$;

* This research was supported by the National Science Foundation (Grant IIS-0118635). The second and third authors are also grateful for the partial support by DIMACS, the National Science Foundation's Center for Discrete Mathematics and Theoretical Computer Science.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 298–309, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Cut-disjunction (CD): s_i is not connected to t_i in the (di)graph $G' = (V, E \setminus E')$ for some $i \in [k]$.

These problems play an important role in network reliability, where edges or arcs represent communication or transportation links, which may work or fail independently, and where the goal is to determine the probability that the network is working, based on the individual edge/arc failure probabilities. It turns out that such network reliability computations require in the general case the list of all multipolar (directed) paths and cuts, depending on the type of connectivity the network ought to maintain, see e.g. [1,4,5,11].

It is easy to see that for the path-conjunction and cut-disjunction problems on undirected graphs, we can assume without loss of generality that the given family of pairs $\mathcal{P}$ corresponds to all pairs in a given family of disjoint vertex sets $V_1, \ldots, V_k \subseteq V$,i.e., $\mathcal{P} = \frac{V_1}{2} \cup \ldots \cup \frac{V_k}{2}$. Such path-conjunctions are usually referred to in the literature as *generalized Steiner trees.* In other words, a generalized Steiner tree is a minimal set of edges $E' \subseteq E$ connecting all vertices within each set V_i, i.e. for each $i = 1, \ldots, k$, all vertices of V_i must belong to a single connected component of (V, E'). In particular, for $k = 1$ we obtain the usual definition of Steiner trees. When each set V_i consists of two vertices $\{u_i, v_i\}$, generalized Steiner trees are called *point-to-point* connections. By the same token, the general case of the cut-disjunction problem for undirected graphs calls for enumerating all minimal subsets of edges whose removal disconnects at least one of the given vertex sets $V_1, \ldots, V_k \subseteq V$.

Given a (di)graph $G = (V, E)$, we consider the problem of listing all subgraphs of G, or correspondingly, the family $\mathcal{F}_\pi \subseteq 2^E$ of all minimal subsets of E, satisfying a certain monotone connectivity property $\pi : 2^E \mapsto \{0, 1\}$:

GEN$(\mathcal{F}_\pi, \mathcal{X})$**:** *Given a monotone connectivity property π and a subfamily $\mathcal{X} \subseteq \mathcal{F}_\pi$ of subgraphs of a (di)graph G satisfying π, either find a new subgraph $X \in \mathcal{F}_\pi \setminus \mathcal{X}$, or prove that the given partial list is complete: $\mathcal{X} = \mathcal{F}_\pi$.*

For instance, if $\pi(X)$ is the property that the subgraph with edge set $X \subseteq E$ is connected, then $\mathcal{F}_\pi$ is the family of spanning trees of G. In this paper, we shall represent subgraphs of a given graph $G = (V, E)$ as subsets of edges. Enumeration algorithms for listing subgraphs satisfying a number of monotone properties are well known. For instance, it is known [13] that the problems of listing all minimal cuts or all spanning trees of an undirected graph $G = (V, E)$ can be solved with delay $O(|E|)$ per generated set. It is also known (see e.g., [6, 7,12]) that all minimal (s,t)-cuts or (s,t)-paths, can be listed with delay $O(|E|)$ per cut or path. Furthermore, polynomial delay algorithms also exist for listing directed spanning trees in a directed graph [9,14]. We consider the families of minimal path conjunctions and cut disjunctions for a given graph G and family of pairs $\mathcal{P}$, denoted respectively by $\mathcal{F}_{PC}(G, \mathcal{P})$, and $\mathcal{F}_{CD}(G, \mathcal{P})$.

Theorem 1. *For any undirected graph G and any family of pairs of vertices $\mathcal{P}$, the two generation problems $GEN(\mathcal{F}_{PC}(G, \mathcal{P}), \mathcal{X})$ and $GEN(\mathcal{F}_{CD}(G, \mathcal{P}), \mathcal{X})$ can be solved in polynomial time.*

Theorem 1 can be stated in the more general setting of generation problems in matroids. These more general results will be described in Section 2. More generally, the above two problems $\mathrm{GEN}(\mathcal{F}_{PC}(G,\mathcal{P}),\mathcal{X})$ and $\mathrm{GEN}(\mathcal{F}_{CD}(G,\mathcal{P}),\mathcal{X})$ remain tractable if we fix some edges, and ask for minimal extensions that form path conjunctions or cut disjunctions. The reduction is trivial: just contract these edges and consider the problem on the resulting graph.

Now let us consider the analogous problems for directed graphs. In the special case when $\mathcal{P}$ is the complete set of pairs of vertices of the graph G, problem $\mathrm{GEN}(\mathcal{F}_{PC}(G,\mathcal{P}),\mathcal{X})$ calls for enumerating minimal strongly connected subgraphs of G, the status of which was determined to be incrementally polynomially solvable in [3]. Problem $\mathrm{GEN}(\mathcal{F}_{CD}(G,\mathcal{P}),\mathcal{X})$, on the other hand, in this special case is known to be NP-hard [3]. We show here that the general path conjunction problem with arbitrary pairs is also NP-hard.

Theorem 2. *Given a digraph G, a family of k disjoint pairs $\mathcal{P}$ of vertices of G, and a subfamily $\mathcal{X} \subseteq \mathcal{F}_{PC}(G,\mathcal{P})$ of path conjunctions of G with respect to $\mathcal{P}$, it is NP-hard to decide if this subfamily is complete: $\mathcal{X} = \mathcal{F}_{PC}(G,\mathcal{P})$.*

As for the *disjunction* of directed paths, it is still open whether their enumeration can be done in incremental polynomial time. However, we prove in Section 5 that the following related extension problem is NP-hard.

Theorem 3. *Let $G = (V, E)$ be a digraph and $\mathcal{P}$ be a family of pairs of vertices of G. Fix a subset of arcs $Y \subseteq E$ and let $\mathcal{F}_{PD}(G,\mathcal{P},Y)$ be the family of minimal subsets of arcs $X \subseteq E$ such that the graph $(V, X \cup Y)$ contains a directed path between at least one of the pairs of vertices in $\mathcal{P}$. Given a sublist $\mathcal{X} \subseteq \mathcal{F}_{PD}(G,\mathcal{P},Y)$ it is NP-hard to check if the given sublist is complete: $\mathcal{X} = \mathcal{F}_{PD}(G,\mathcal{P},Y)$.*

The same statement holds for the extension problem for disjunction of cuts. These results should be contrasted with the corresponding results for undirected graphs, where the extension problems are polynomially solvable.

Finally, we shall also prove the following positive result which generalizes previously known results on generating directed spanning trees [9,14].

Theorem 4. *Let $G = (V, E)$ be a directed graph, $s_1, s_2 \in V$ be two arbitrary (not necessarily distinct) vertices, and $t_1, T_2 \subseteq V$ be arbitrary vertex and subset of vertices of V. Let $\mathcal{P} = \{(s_1, t_1)\} \bigcup \{(s_2, t) : t \in T_2\}$. Then problem $GEN(\mathcal{F}_{PC}(G,\mathcal{P}),\mathcal{X})$ can be solved with polynomial delay.*

To prove Theorem 4, we use the *backtracking* method for enumeration (see [13]). This method can generally be explained as follows. Suppose that we want to enumerate all elements of a family $\mathcal{F} \subseteq 2^E$, where $E = \{1, 2, \ldots, |E|\}$. The algorithm works by building a search tree of depth $|E|$ whose leaves contain the elements of the family $\mathcal{F}$. Each node of the tree is identified with two disjoint subsets $S_1, S_2 \subseteq E$, and have at most two children. At the root of the tree, we have $S_1 = S_2 = \emptyset$. The left child of any node (S_1, S_2) of the tree at level i is $(S_1 \cup \{i\}, S_2)$ provided that there is an $X \in \mathcal{F}$, such that $X \supseteq S_1 \cup \{i\}$ and $X \cap S_2 = \emptyset$. The right child of any node (S_1, S_2) of the tree at level i

is $(S_1, S_2 \cup \{i\})$ provided that there is an $X \in \mathcal{F}$, such that $X \supseteq S_1$ and $X \cap (S_2 \cup \{i\}) = \emptyset$. When we extend the set S_1 by a new element $i \in E$ to get a new set S_1', we may restrict our attention to subsets S_1' satisfying certain properties. More precisely, let $\mathcal{F}' \subseteq 2^E$ be a family of sets such that we can test in polynomial time if a set $X \in \mathcal{F}'$, and such that for every $X \in \mathcal{F}$, there is a set $X' \in \mathcal{F}'$ contained in X. The following is a sufficient condition for this method to work in polynomial time:

(F) For any two disjoint subsets $S_1 \in \mathcal{F}'$ and $S_2 \subseteq E$, and for any $i \in E \setminus (S_1 \cup S_2)$ such that $S_1 \cup \{i\} \in \mathcal{F}'$, we can check in polynomial time if there is an element $X \in \mathcal{F}$, such that $X \supseteq S_1 \cup \{i\}$ and $X \cap S_2 = \emptyset$.

This way, under assumption (F), we obtain a polynomial delay, polynomial space algorithm for enumerating the elements of $\mathcal{F}$ in lexicographic order, by performing a *depth-first search* traversal on the nodes of the backtracking tree constructed as above.

The remainder of the paper consists of the proof of a more general result for matroids in Section 2 instrumental in the proof of Theorem 1 in Section 3. The proofs of Theorems 2, 3 and 4 are presented in Sections 4, 5 and 6, respectively.

2 Generation Problems in Matroids

Many generation problems for undirected graphs can be viewed more generally as generation problems in graphical or co-graphical matroids, see [10,16] for general background on Matroid Theory.

Let M be a matroid on ground set S of cardinality $|S| = n$. In general, we assume that M is defined by an *independence oracle,* i.e. an algorithm $\mathcal{O}$ which, given a subset X of S, can determine in unit time whether or not X is independent in M. This assumption implies that the rank of any set $X \subseteq S$, $r(X) = \max\{|I| \mid I \text{ independent subset of } X\}$, and in particular, the rank of the matroid $r(M) \stackrel{\text{def}}{=} r(S)$ can be determined in $O(n)$ time by the well-known greedy algorithm. A set $U \subseteq S$ is said to *span* an element $s \in S$ if $r(U \cup \{s\}) = r(U)$, to span a set $W \subseteq S$ if $r(U \cup W) = r(U)$, and to span the whole matroid M if $r(U) = r(S) = r(M)$. In the last case S is called a *spanning set.* Minimal spanning sets for S are called *bases.* Equivalently, a base B is a maximal independent set. Further, $|B| = r(M)$ for every base B in M. Minimal dependent sets in M are called *circuits.*

Denote the hypergraphs of all bases and circuits of M by $\mathcal{B}(M)$ and $\mathcal{C}(M)$ respectively. Given a matroid M, it is known that the complementary set $\mathcal{B}^c = \{S \setminus B \mid B \in \mathcal{B}(M)\}$ is the set of bases of another matroid M^* on the same ground set S. The matroid M^* is called the *dual matroid* of M. The bases of M^* are called the *co-bases* of M and the circuits of M^* are called the *co-circuits* of M. Note that the rank of a set $X \subseteq S$ in the dual matroid M^*, $r^*(X) = r(S \setminus X) + |X| - r(M)$, can also be computed in $O(n)$ oracle time, and thus, $\mathcal{M}$ can be used as an independence oracle for the dual matroid.

In reliability theory we apply the above definitions to graphical and co-graphical matroids. Given a connected multi-graph $G = (V, E)$, we denote by M_G the graphical matroid for G. The elements of M_G are the edges of G, and a set $E' \subseteq E$ spans $e = (v', v'')$ iff E' contains a path between v' and v''. In particular, E' is independent iff it is a forest, that is if (V, E') contains no cycle. This means that the bases M_G are the spanning trees of G, and the circuits of M_G are the simple cycles of G. Thus the paths between vertices v' and v'' in G are identified with the circuits through the edge $e = (v', v'')$ in M_G. (If $e = (v', v'') \notin E$, we can add e to M_G.) The bases of the dual (co-graphical) matroid M_G^* are the complements to the spanning trees, i.e. the minimal transversals to the simple cycles, so called *feedbacks*. The circuits are the minimal transversals to the spanning trees, that is minimal *cuts* for G.

We shall need the following Theorem from [2]

Theorem 5. *Let M be a matroid with ground set S, let $a \in S$, and let $\mathcal{C}(M, a)$ be the set of circuits C of M such that $a \in C$. Assuming that M is defined by an independence oracle, all elements of $\mathcal{C}(M, a)$ can be enumerated in incremental polynomial time.*

The above theorem can be extended as follows.

Theorem 6. *Given a matroid M with ground set S and two non-empty sets D, $A \subseteq V$, all minimal subsets of $\mathcal{D}$ which span A can be enumerated in incremental polynomial time. All maximal subsets of D which do not span A can also be enumerated in incremental polynomial time.*

Proof. Let α be a new element representing A, and let M_α be the matroid on $D \cup \alpha$ with the following rank function:

$$\rho(X) = \begin{cases} r(X), & \text{if} \quad \alpha \notin X \\ \max\{r((X \setminus \alpha) \cup a) \mid a \in A\}, & \text{otherwise.} \end{cases} \tag{1}$$

It is easy to check that M_α is indeed a matroid. When M is a vectorial matroid over a large field, α can be interpreted as the "general linear combination" of all elements of A; in general, $\rho(X)$ is the so-called *principal extension of* $r(X)$ *on* A *with value* 1 (see e.g. [8]). Let $SPAN(D, A)$ be the collection of all minimal subsets of $\mathcal{D}$ which span A. When $I \in \mathcal{SPAN}(D, A)$ then $I \cup \alpha$ is a circuit in M_α and conversely, for any circuit C in M_α containing α, the set $C \setminus \alpha$ belongs to $\mathcal{SPAN}(D, A)$. Hence the enumeration problem for $\mathcal{SPAN}(D, A)$ is equivalent with that for the set of all circuits through α in M_α. Given an independence oracle for M, the rank function (1) of the extended matroid can be trivially evaluated in oracle-polynomial time. Therefore the first claim of Theorem 6 directly follows from Theorem 5. To see the second claim note that the maximal subsets of D which do not span A are in one-to-one correspondence with the circuits of the dual matroid M_α^* which contain α. □

3 Proof of Theorem 1

Let us consider first path-conjunctions (PC): In order to generate all minimal subsets X of edges of a given (multi) graph $G = (V, E)$ such that $G(X) = (V, X)$ contains paths between given pairs of vertices s_i, t_i, for $i \in [k]$, we add the set A of k of "new" edges $e_i = (s_i, t_i)$ to E, and let $D = E$. Then each minimal subset of D which spans A in the graphical matroid of $G' = (V, D \cup A)$ is a minimal set of edges of G connecting all pairs of vertices s_i, t_i, and vice versa. Consequently, all such minimal path conjunctions can be enumerated in incremental polynomial time by the first part of Theorem 6.

Let us consider next cut-disjunctions (CD): Generating all minimal subsets of edges of $G = (V, E)$ which disconnect at least one pair of vertices s_i, t_i for $k \in [k]$ is equivalent to the generation of all maximal subsets of E which do not connect all the pairs. Letting as before $D = E$ and $A = \{(s_i, t_i) : i \in [k]\}$, we conclude that all such maximal cut disjunction can be enumerated in incremental polynomial time by the second part of Theorem 6. □

Remark. Suppose we are given a graph $G = (V, E)$, a set of k terminals $\{s_1, \ldots, s_k\} \subseteq V$, and it is sought to enumerate all minimal sets of edges whose removal disconnects the terminals $\{s_1, \ldots, s_k\}$ from each other. Such a minimal set of edges is commonly referred to as a *multiway cut,* see [15].

Interestingly, the special case of enumerating cut conjunctions can be solved in incremental polynomial time as follows. First of all, we can assume without loss of generality that the input graph $G = (V, E)$ contains no edges between the given terminals $s_1, \ldots, s_k$: if such edges exist, then any multiway cut must remove all of them. Let A be the set of $k-1$ "new" edges forming a spanning tree on the terminals, say $A = \{a_1 = (s_1, s_2), a_2 = (s_2, s_3), \ldots, a_{k-1} = (s_{k-1}, s_k)\}$. Let $G' = (V, E')$, where $E' = E \cup A$, and let M^* be the co-graphical matroid of G'. An edge set $X \subseteq E$ spans A in M^* if and only if

$$r^*(X \cup A) = r^*(X), \tag{2}$$

where $r^*(\cdot)$ is the rank function of M^*. By definition, $r^*(Z) = r(E' \setminus Z) + |Z| - r(M)$, where M is the graphical matroid of G' and $r(\cdot)$ is the rank function of M. So condition (2) can be written as follows: $r(E \setminus X) + |A| = r(E' \setminus X)$, or equivalently, $r(Y) + |A| = r(Y \cup A)$ where $Y = E \setminus X$ is the complement of X in $G = (V, E)$. So if we remove X from G, and then start adding the edges of A to the resulting graph (V, Y), then each new edge from A should be decreasing the number of connected components. But this is the same as saying that the terminals $s_1, \ldots, s_k$ are all in distinct connected components of (V, Y), i.e. that X is a multiway cut. So the enumeration problem for multiway cuts, in this case, is equivalent to the enumeration of all minimal subsets X of E such that X spans A in M^*, which, by Theorem 6, can be done in incremental polynomial time. □

Analogously, the special case of path disjunction, when the goal is to enumerate all minimal subsets of edges which connect at least one pair of vertices in $\{s_1, s_2, \ldots, s_k\}$, can also be solved in incremental polynomial time.

4 Proof of Theorem 2

We use a polynomial-time transformation from the satisfiability problem. Let $\phi = C_1 \wedge \ldots \wedge C_m$ be a conjunctive normal form on $2n$ literals $x_1, \overline{x}_1, \ldots, x_n, \overline{x_n}$. We construct a digraph $G = (V,E)$ on $|V| = 6m + n + 2$ vertices and $|E| = 6m + 2n + \sum_{j=1}^{m} |C_j|$ arcs. An example of our construction is given in Figure 1.
Vertices: We associate with each clause C_j, for $j = 1, \ldots, m$, 6 vertices of G: s_j, s'_j, s''_j and t_j, t'_j, and t''_j. We associate with each variable x_i, for $i = 1, \ldots, n$ a vertex x_i of V. In addition there are two other vertices u and v.
Arcs: For each clause C_j, for $j = 1, \ldots, m$, we have 6 arcs $(s'_j, s_j), (s_j, u), (u, s''_j)$ and $(t'_j, v), (v, t_j), (v, t''_j)$. For each $i = 1, \ldots, n$, we also have two arcs (u, x_i) and (x_i, v) representing positive and negative literals x_i and $\overline{x}_i$ respectively. Finally, if x_i appears in C_j we add an arc between s_j and x_i and if $\overline{x}_i$ appears in C_j we add an arc between x_i and t_j.
Pairs: We use the following set $\mathcal{P}$ of $3m$ disjoint pairs:

$$\mathcal{P} = \{(s_j, t_j) : j \in [m] = \{1, \ldots, m\}\} \bigcup \{(s'_j, s''_j) : j \in [m]\} \bigcup \{(t'_j, t''_j) : j \in [m]\}.$$

Trivial path conjunctions: Note that any path conjunction must include the sets of arcs $E_j = \{(s'_j, s_j), (s_j, u), (u, s''_j), (t'_j, v), (v, t_j), (t_j, t''_j)\}$, for $j = 1, \ldots, m$. Thus, any path conjunction that connects the family of pairs specified by $\mathcal{P}$ must be an extension of this set of arcs. We call such an extension trivial if it includes both arcs (u, x_i) and (x_i, v) for some $i \in [n]$. This way, we define a set of n trivial path conjunctions.
Non-trivial minimal path conjunctions: Let us now show that any non-trivial minimal path conjunction yields a satisfying assignment for C and conversely, any satisfying assignment for ϕ gives a non-trivial minimal path conjunction for G with respect to $\mathcal{P}$. This will prove Theorem 2. Note that any such non-trivial path conjunction must avoid one of the arcs $(u, x_i), (x_i, v)$ for each $i = 1, \ldots, n$.

Let $\sigma = (\ell_1, \ell_2, \ldots, \ell_n)$ be the set of literals assigned the value *True* in a satisfying truth assignment for C. We define a non-trivial minimal path conjunction

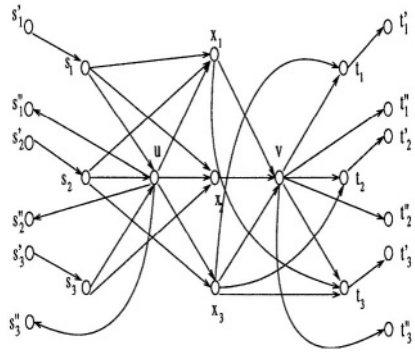


Fig. 1. An example of the NP-hard construction proof for Theorem 2 with CNF $\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$.

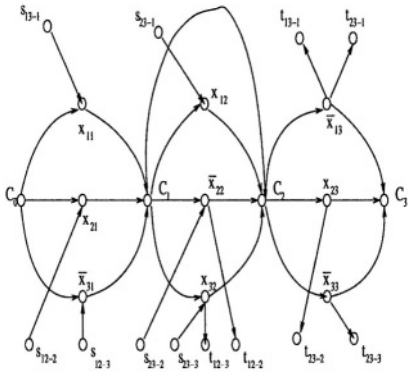


Fig. 2. An example of the NP-hard construction proof for Theorem 3 with CNF $\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$.

X of G corresponding to σ as follows. First we include in X all sets of arcs belonging to E_j, for $j = 1, \ldots, m$. Next, we append to X all arcs corresponding to literals *not* appearing in σ. Then for each clause C_j, for $j = 1, \ldots, m$, we include one of the arcs (s_j, x_i) or (x_i, t_j) depending on which of the literals x_i or $\bar{x}_i$ satisfies C_j, respectively. Now, the fact that σ is satisfying implies that there is a path from s_j to t_j, for each $j = 1, \ldots, m$.

Conversely, let X be a non-trivial minimal path conjunction connecting the pairs of $\mathcal{P}$. The non-triviality of X implies that, for each $i = 1, \ldots, n$, one of the arcs (u, x_i) or (x_i, v) is not included in X. Let us define a satisfying truth assignment σ that assigns True to the literals corresponding to arcs *not* included in X. Since X contains a path from s_j to t_j for each $j \in [m]$, one of the arcs (s_j, x_i) or (x_i, t_j) must be included in X, implying that ϕ is satisfiable.

5 Proof of Theorem 3

Again, we use a polynomial-time transformation from the satisfiability problem. Let $\phi = C_1 \wedge \ldots \wedge C_m$ be a conjunctive normal form (CNF) on $2n$ literals $x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n$. Assume that no clause contains a literal and its negation, and that each of the $2n$ literals appears in at least one of the clauses. For a literal ℓ_i, let $d(\ell_i)$ be the number of clauses in which the literal appears. We construct a digraph $G = (V, E)$ on $|V| = m + 1 + \sum_{j=1}^{m} |C_j| + 2\sum_{i=1}^{n} d(x_i)d(\overline{x}_i)$ vertices and $|E| \leq 2\sum_{j=1}^{m} |C_j| + 3\sum_{i=1}^{n} d(x_i)d(\overline{x}_i)$ arcs. An example of our construction is given in Figure 2.

Vertices: In addition to vertex C_0, we associate a vertex $C_j \in V$ with each clause C_j, for $j = 1, \ldots, m$. For each pair of clauses (C_j, C_k), $j < k$, such that literal ℓ_i appears in C_j and $\overline{\ell}_i$ appears in C_k, we define two vertices s_{jk-i}, t_{jk-i}. Finally, for each clause C_j, for $j = 1, \ldots, m$, and for each literal ℓ_i appearing in C_j, we define a vertex ℓ_{ij} of V.

Arcs: For each clause C_j and each literal ℓ_i appearing in C_j we define two arcs (C_{j-1}, ℓ_{ij}) and (ℓ_{ij}, C_j). In addition, for each pair of clauses (C_j, C_k), $j < k$, such that literal ℓ_i appears in C_j and $\bar{\ell}_i$ appears in C_k, we include the arcs $(s_{jk-i}, \ell_{ij}), (C_j, C_{k-1})$ and $(\bar{\ell}_{ik}, t_{jk-i})$.
Pairs: We use a set $\mathcal{P}$ of $\sum_{i=1}^{n} d(x_i)d(\bar{x}_i)+1$ disjoint pairs. $\mathcal{P}$ contains (C_0, C_m) and (s_{jk-i}, t_{jk-i}) for every pair of clauses (C_j, C_k), $j < k$, such that literal ℓ_i appears in C_j and $\bar{\ell}_i$ appears in C_k.
Fixed set of arcs Y: We include in Y all arcs of the form $(s_{jk-i}, \ell_{ij}), (\ell_{ij}, t_{jk-i})$ and (C_j, C_{k-1}) for all $j, k \in [m]$ and all literals ℓ_i appearing in C_j or C_k.

Clearly, for every pair of clauses (C_j, C_k), $j < k$, such that literal ℓ_i appears in C_j and $\bar{\ell}_i$ appears in C_k, the two arcs $(\ell_j i, C_j)$ and $(\bar{\ell}_{ik}, C_k)$ extend Y to a dipath connecting s_{jk-i} to t_{jk-i}. Thus apart form this set of $\sum_{i=1}^{n} d(x_i)d(\bar{x}_i)$ paths, any other path disjunction must avoid one of the arcs (ℓ_{ij}, C_j) or $(\bar{\ell}_{ik}, C_k)$ for every $j < k$, such that literal ℓ_i appears in C_j and $\bar{\ell}_i$ appears in C_k. The only path that remains to be connected is the one from C_0 to C_m and it is easy to see that such path corresponds to a satisfying truth assignment for ϕ.

6 Proof of Theorem 4

Let $G = (V, E)$ be a digraph. Given a node $s \in V$ and a subset of nodes $T \subseteq V$, let us call an (s,T)-*directed Steiner tree* any minimal subset of arcs $X \subseteq E$ such that there is a dipath in (V, X) from s to every node in T. Note that the minimality of X implies indeed that the underlying graph of (V, X) must be a tree, whose leaves belong to T.

Given $s \in V$, [9,14] give algorithms for enumerating all directed trees connecting s to all other nodes in the graph. We begin first by showing how to use the backtracking approach to enumerate all (s,T)-directed Steiner trees connecting a given node $s \in V$ to a given subset of nodes $T \subseteq V$. For this we need to verify that (F) is satisfied. Let $\mathcal{F}$ be the family of directed (s,T)-Steiner trees, and $\mathcal{F}'$ be the family of directed Steiner subtrees $\mathbf{T}'$ connecting s to some subset of vertices $T' \subseteq T$, and such that every leaf of $\mathbf{T}'$ belongs to T', except possibly one leaf u which is not in T, but there is a path from u to some $t \in T \setminus T'$ that does not use any vertex of $\mathbf{T}'$. Consider any two disjoint subsets $S_1 \in \mathcal{F}'$ and $S_2 \subseteq E$. Let T' be the subset of T reachable from s by using only arcs from S_1. If the tree (V, S_1) has a leaf vertex $u \notin T$, then any arc $a \in E \setminus (S_1 \cup S_2)$ satisfying $S_1 \cup \{a\} \in \mathcal{F}'$ must be of the form (u, v). Otherwise, any arc $a = (u, v) \in E$ can be used to extend S_1 as long as u is reachable from s by a dipath in the graph (V, S_1). In both cases, the check whether $S_1 \cup \{a\}$ can be extended to an (s,T)-directed Steiner tree that avoids S_2 can be done by simply deleting all the vertices appearing in S_1 and all the arcs appearing in S_2 from G and checking the reachability of some vertex in $T \setminus T'$ from v.

Now we turn our attention to the case with two source vertices s_1, s_2 and sinks $t_1 \in V$ and $T_2 \subseteq V$. We assume that the graph (V, E) contains a dipath connecting s_1 to t_1 and a directed Steiner tree connecting s_2 to every vertex in T_2. Otherwise there is nothing to generate. Such a condition is easy to verify.

Let us call a *minimal $\mathcal{P}$-linked subgraph,* MPLS in short, any minimal collection of arcs X such that there is a directed path in (V, X) from s_1 to t_1, and from s_2 to any node in T_2. Given a digraph G and an (s, T)-Steiner tree $\mathbf{T}$ in G, we call a *cross path* any (v_1, v_2)-dipath in $(V, E \backslash E(\mathbf{T}))$, where v_1 and v_2 are two vertices such that there is no dipath from v_2 to v_1 in $\mathbf{T}$. The following property of MPLS holds.

Lemma 1. *Let $X \subseteq E$ be the union of an (s_1, t_1)-dipath $\mathbf{P}_1$ and an (s_2, T_2)-Steiner tree $\mathbf{T}_2$. Then X is an MPLS if and only if X has no cross paths.*

Proof. Suppose that X is an MPLS, but has a cross path from v_1 to v_2. Consider the unique dipath $s_2 = x_0, x_1, \ldots, x_t = v_2$, in $\mathbf{T}_2$ from s_2 to v_2 and let x_r be the last vertex on this path that belongs to $T_2 \cup \{v_1\}$, or lies on a path from s_2 to some vertex $t \in T_2$ that does not pass through v_2. Then, if we delete from X all the arcs $(x_r, x_{r+1}), \ldots, (x_{t-1}, x_t)$, we would still have a path between s_1 and t_1 and between s_2 and every vertex in T_2. This contradicts the minimality of X.

Conversely, suppose that we drop an arc $a \in X = E(\mathbf{P}_1) \cup E(\mathbf{T}_2)$, and the graph $(V, X \backslash \{a\})$ still contains an (s_1, t_1)-dipath and an (s_2, T_2)-directed Steiner tree. If a belongs to $\mathbf{P}_1$, then X contains two distinct dipaths between s_1 and t_1, and if a belongs to $\mathbf{T}_2$, then X contains two distinct dipaths between s_2 and some $t \in T_2$. In both cases, it easy to see that X must have a cross path. □

The two following corollaries are immediate from Lemma 1.

Corollary 1. *Let X be an MPLS that is composed of the union of a directed (s_1, t_1)-path $\mathbf{P}_1$ and a directed Steiner tree $\mathbf{T}_2$ connecting s_2 to every node in T_2. Suppose that $\mathbf{P}_1$ and $\mathbf{T}_2$ contain common vertices v_1 and v_2 such that v_1 lies on the path between s_1 and v_2 in $\mathbf{P}_1$, and on the path between s_2 and v_2 in $\mathbf{T}_2$, then $\mathbf{P}_1$ and $\mathbf{T}_2$ must contain exactly the same set of vertices between v_1 and v_2.*

Corollary 2. *In every MPLS X, the dipath from s_1 to t_1 is unique, and the dipath from s_2 to any $t \in T_2$ is unique.*

Yet, for more than two sources, Corollary 2 may not hold, as illustrated by the following example with three sources $\{s_1, s_2, s_3\}$ and three sinks $\{t_1, t_2, t_3\}$ (see Figure 3). Let G be a digraph on 10 vertices $\{s_1, s_2, s_3, a, b, c, d, t_1, t_2, t_3\}$ and with arcs $\{(s_1, a), (s_2, b), (s_3, d), (b, t_3), (c, t_1), (d, t_2), (a, b), (b, c), (a, d), (d, c), (c, a)\}$. Clearly, in this example, there exists two dipaths between s_1 and t_1 namely, s_1, a, b, c, t_1 and s_1, a, d, c, t_1. Yet, all the arcs are needed to maintain the connectivity between s_2 and t_2 or between s_3 and t_3.

Lemma 2. *Every (s_1, t_1)-dipath can be extended to an MPLS, and every MPLS is an extension of some (s_1, t_1)-dipath.*

Proof. The first part of the lemma is a consequence of Lemma 1. Indeed, fix an (s_1, t_1)-dipath $\mathbf{P}_1$ and let $\mathbf{T}_2$ be an arbitrary directed (s_2, T_2)-Steiner tree. Let us modify the union $F = E(\mathbf{P}_1) \cup E(\mathbf{T}_2)$ by deleting arcs from $\mathbf{T}_2$ to obtain an MPLS $X \subseteq F$ as follows. For every two vertices v_1 and v_2 in F such that there is

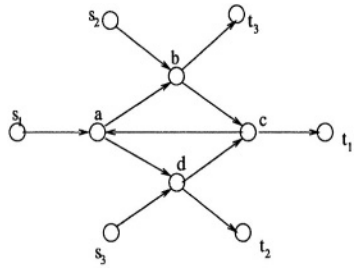


Fig. 3. An example showing that Corollary 2 does not hold with 3 sources.

a cross path from v_1 to v_2, consider the unique dipath $s_2 = x_1, x_2, \ldots, x_t = v_2$, in $\mathbf{T}_2$ from s_2 to v_2 and let $x_r \neq v_2$ be the last vertex on this path that belongs to $T_2 \cup \{v_1\}$, or lies on a path, in $\mathbf{T}_2$, from s_2 to some vertex $u \in T_2$ that does not pass through v_2. Then, we delete from F all the arcs $(x_r, x_{r+1}), \ldots, (x_{t-1}, x_t)$. Clearly the resulting graph still contains an MPLS. We repeat this operation until there are no cross paths left. The resulting graph will be an MPLS by Lemma 1. To see the second part, note by Corollary 2 that an MPLS X contains a unique path from s_1 to every $t \in T_2$. This readily defines an (s_1, t_1)-dipath of which X is an extension. □

To prove Theorem 4, we apply backtracking as follows. First, we generate an (s_1, t_1)-dipath $\mathbf{P}_1$ connecting s_1 to t_1. We can generate all such paths in lexicographic order as explained above, and by Lemma 2 we know that each of them can be extended to an MPLS, and that all such possible extensions are exactly the set of MPLS's. Furthermore, the MPLS extensions X and X' of any two distinct (s_1, t_1)-dipaths $\mathbf{P}_1$ and $\mathbf{P}'_1$ are distinct. This follows from the fact that if $X = X'$ then both of them contain $\mathbf{P}_1 \cup \mathbf{P}'_1$, and hence they contain at least two different dipaths between s_1 and t_1, in contradiction to Corollary 2.

For each generated (s_1, t_1)-dipath $\mathbf{P}_1$, let us generate all possible extensions to an MPLS using backtracking. Let $\mathcal{F}$ be the family of all such possible extensions, and $\mathcal{F}'$ be the family of all subsets $X \subseteq E \setminus E(\mathbf{P}_1)$ of arcs such that (i) X forms with some subset of arcs $Y \subseteq E(\mathbf{P}_1)$ a directed Steiner subtree $\mathbf{T}'_2$ connecting s_2 to some subset of vertices $T'_2 \subseteq T_2$, (ii) every leaf of $\mathbf{T}'_2$ belongs to T'_2, except possibly one leaf u which is not in T_2, but there is a dipath in G from u to some $t \in T_2 \setminus T'_2$ that does not use any vertex of $\mathbf{T}'_2$, and (iii) there are no cross paths in $\mathbf{P}_1 \cup \mathbf{T}'_2$. By Lemma 1, any set $X \subseteq E$ satisfying (i), (ii) and (iii) can be extended to an MPLS, and therefore, all we need to check is that condition (F) is satisfied. Consider any two disjoint subsets $S_1 \in \mathcal{F}'$ and $S_2 \subseteq E$. Let T'_2 be the subset of T_2 reachable from s_2 by using only arcs from S_1. If the subgraph $(V, S_1 \cup E(\mathbf{P}_1))$ has a leaf vertex (that is a vertex with out-degree zero) $u \notin T_2 \cup \{t_1\}$, then any arc $a \in E \setminus (S_1 \cup S_2)$ satisfying $S_1 \cup \{a\} \in \mathcal{F}'$ must be of the form (u, v). Otherwise, any arc $a = (u, v) \in E$ can be used to extend S_1 as long as u is reachable from s_2 by a dipath in the

graph $(V, S_1 \cup E(\mathbf{P_1}))$. In both cases, the check whether $S_1 \cup \{a\}$ can be extended to an MPLS that avoids S_2 can be done as follows. First, assume the path $\mathbf{P_1}$ consists of the vertices $s_1 = x_1, x_2, \ldots, x_r = t_1$ in that order, and let $\mathbf{P}$ be the path $s_2 = y_1, y_2, \ldots, y_k = v$ from s_2 to v in $(V, S_1 \cup E(\mathbf{P_1}))$. If the paths $\mathbf{P_1}$ and $\mathbf{P}$ intersect and $\mathbf{P}$ is the *only* path, from s_2 to a node in $\mathbf{T}'_2$, that intersects $\mathbf{P_1}$, let $l \in \{1, \ldots, k\}$ be the largest index such that $y_l = x_t$ for some $t \in \{1, \ldots, r\}$. Then all vertices on the sub-path $x_{t+1}, \ldots, x_r = t_1$ must be deleted from G to avoid cross paths. Otherwise, if there is a path in $\mathbf{T}'_2$, from s_2 to some $v' \neq v$, that intersects $\mathbf{P_1}$, then all the nodes on $\mathbf{P_1}$ must be deleted from G to avoid cross paths. Next, we delete also from G all the vertices appearing in S_1 and all the arcs in S_2. Finally, checking (F) reduces now to checking the reachability, from v, of some vertex in $T_2 \setminus T'_2$ in the remaining graph.

References

1. U. Abel and R. Bicker, Determination of All Cutsets Between a Node Pair in an Undirected Graph, *IEEE Transactions on Reliability* **31** (1986) pp. 167–171.
2. E. Boros, K. Elbassioni, V. Gurvich and L, Khachiyan. Algorithms for Enumerating Circuits in Matroids, in *the Proceedings of the 14th Annual International Symposium on Algorithms and Computation (ISAAC 2003), LNCS* 2906, pp. 485–494, Kyoto, Japan, December 2003.
3. E. Boros, K. Elbassioni, V. Gurvich and L. Khachiyan, On enumerating minimal dicuts and strongly connected subgraphs, to appear in *Proc. 10th Conf. on Integer Programming and Combinatorial Optimization (IPCO X),* June 2004.
4. V.K. Bansal, K.B. Misra and M.P. Jain, Minimal Pathset and Minimal Cutset Using Search Technique, *Microelectr. Reliability* **22** (1982), pp. 1067–1075.
5. C. J. Colburn, The Combinatorics of Network Reliability, Oxford Univ. Press, New York, 1987.
6. N. D. Curet, J. DeVinney and M. E. Gaston, An efficient network flow code for finding all minimum cost s-t cutsets, *Comp. and Oper. Res.* **29** (2002), pp. 205–219.
7. D. Gusfield and D. Naor, Extracting maximum information about sets of minimum cuts, *Algorithmica* **10** (1993), pp. 64–89.
8. L. Lovász, Submodular functions and convexity, in *Mathematical Programming: The State of the Art,* Bonn 1982, pp. 235–257, (Springer Verlag, 1983).
9. S. Kapoor and H. Ramesh, An Algorithm for Enumerating All Spanning Trees of a Directed Graph, *Algorithmica* 27(2), pp. 120–130, 2000.
10. J. Oxley, Matroid Theory, Oxford University Press, Oxford, 1992.
11. J. S. Provan and M. O. Ball, Computing Network Reliability in Time Polynomial in the Number of Cuts, *Operations Research* **32** (1984), pp. 516–526.
12. J. S. Provan and D. R. Shier, A paradigm for listing (s,t) cuts in graphs, *Algorithmica* **15**, (1996), pp. 351–372.
13. R. C. Read and R. E. Tarjan, Bounds on backtrack algorithms for listing cycles, paths, and spanning trees, *Networks,* 5 (1975) pp. 237–252.
14. T. Uno, An Algorithm for Enumerating all Directed Spanning Trees in a Directed Graph, in *the Proceedings of the 7th Annual International Symposium on Algorithms and Computation ISAAC,* 1996, pp. 166–173
15. V. Vazirani, Approximation Algorithms, Springer-Verlag, Berlin, 2001.
16. D.J.A. Welsh, Matroid Theory, Academic Press, London, New York, San Francisco, 1976.

Packing Directed Cycles Efficiently

Zeev Nutov[1] and Raphael Yuster[2]

[1] Department of Computer Science, The Open University of Israel
Tel Aviv, Israel
nutov@openu.ac.il
[2] Department of Mathematics, University of Haifa at Oranim
Tivon 36006, Israel
raphy@research.haifa.ac.il

Abstract. Let G be a simple digraph. The *dicycle packing number* of G, denoted $\nu_c(G)$, is the maximum size of a set of arc-disjoint directed cycles in G. Let G be a digraph with a nonnegative arc-weight function w. A function ψ from the set $\mathcal{C}$ of directed cycles in G to R_+ is a *fractional dicycle packing* of G if $\sum_{e \in C \in \mathcal{C}} \psi(C) \leq w(e)$ for each $e \in E(G)$. The *fractional dicycle packing number*, denoted $\nu_c^*(G,w)$, is the maximum value of $\sum_{C \in \mathcal{C}} \psi(C)$ taken over all fractional dicycle packings ψ. In case $w \equiv 1$ we denote the latter parameter by $\nu_c^*(G)$.
Our main result is that $\nu_c^*(G) - \nu_c(G) = o(n^2)$ where $n = |V(G)|$. Our proof is algorithmic and generates a set of arc-disjoint directed cycles whose size is at least $\nu_c(G) - o(n^2)$ in randomized polynomial time. Since computing $\nu_c(G)$ is an NP-Hard problem, and since almost all digraphs have $\nu_c(G) = \Theta(n^2)$ our result is a FPTAS for computing $\nu_c(G)$ for almost all digraphs.
The latter result uses as its main lemma a much more general result. Let $\mathcal{F}$ be any fixed family of oriented graphs. For an oriented graph G, let $\nu_{\mathcal{F}}(G)$ denote the maximum number of arc-disjoint copies of elements of $\mathcal{F}$ that can be found in G, and let $\nu_{\mathcal{F}}^*(G)$ denote the fractional relaxation. Then, $\nu_{\mathcal{F}}^*(G) - \nu_{\mathcal{F}}(G) = o(n^2)$. This lemma uses the recently discovered *directed regularity lemma* as its main tool.
It is well known that $\nu_c^*(G,w)$ can be computed in polynomial time by considering the dual problem. However, it was an open problem whether an optimal fractional dicycle packing ψ *yielding* $\nu_c^*(G,w)$ can be generated in polynomial time. We prove that a maximum fractional dicycle packing yielding $\nu_c^*(G,w)$ with at most $O(n^2)$ dicycles receiving nonzero weight can be found in polynomial time.

1 Introduction

All graphs and digraphs considered here are finite and have no loops, parallel arcs or isolated vertices. For the standard terminology used the reader is referred to [5]. We use the terms *digraph* and *dicycle* to refer to a directed graph and a directed cycle, respectively.

We consider the following fundamental problem in algorithmic graph-theory. Given a digraph G, how many arc-disjoint cycles can be packed into G? Define

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 310–321, 2004.
© Springer-Verlag Berlin Heidelberg 2004

the *dicycle packing number* of G, denoted $\nu_c(G)$, to be the maximum size of a set of arc-disjoint dicycles in G. We also consider the fractional relaxation of this problem. Let R_+ denote the set of nonnegative reals. A *fractional dicycle packing* of G is a function ψ from the set $\mathcal{C}$ of dicycles in G to R_+, satisfying $\sum_{e \in C \in \mathcal{C}} \psi(C) \leq 1$ for each $e \in E(G)$. Letting $|\psi| = \sum_{C \in \mathcal{C}} \psi(C)$, the *fractional dicycle packing number*, denoted $\nu_c^*(G)$, is defined to be the maximum of $|\psi|$ taken over all fractional dicycle packings ψ. Since a dicycle packing is also a fractional dicycle packing, we always have $\nu_c^*(G) \geq \nu_c(G)$. The notion of a fractional dicycle packing can be extended to digraphs with nonnegative arc weights. In this case we require that $\sum_{e \in C \in \mathcal{C}} \psi(C) \leq w(e)$ for each $e \in E(G)$ where $w(e)$ is the weight of e. We denote by $\nu_c^*(G, w)$ the corresponding *fractional dicycle packing number* where $w : E \to R_+$ is the weight function.

Problems concerning packing arc-disjoint or vertex-disjoint dicycles in digraphs have been studied extensively (see, e.g., [4,14]). It is well known that computing $\nu_c(G)$ (and hence finding a maximum dicycle packing) is an NP-Hard problem. Even the very special case of deciding whether a digraph has a triangle decomposition is known to be NP-Complete (see, e.g. [6] for a more general theorem on the NP-Completeness of such decomposition problems). Currently, the best approximation algorithm for this problem [12] has an approximation ratio of $O(n^{1/2})$ which is also an upper bound for the integrality gap. Thus, it is interesting to find out when $\nu_c(G)$ and $\nu_c^*(G)$ are "close" as this immediately yields an efficient approximation algorithm for this NP-Hard problem. Our main result shows that the two parameters differ by at most $o(n^2)$, thus giving an approximation algorithm with an $o(n^2)$ additive error term.

Theorem 1. *If G is an n-vertex digraph then $\nu_c^*(G) - \nu_c(G) = o(n^2)$. Furthermore, a set of at least $\nu_c(G) - o(n^2)$ arc-disjoint dicycles can be generated in randomized polynomial time.*

The $o(n^2)$ additive error term is only interesting if the graph G is dense and $\nu_c(G) = \Theta(n^2)$. This, however, is the case for almost all digraphs, as it is known (and easy) that the directed random graph $G(n,p)$ has $\nu_c(G) = \Theta(n^2)$ for any constant p, $0 < p < 1$ (in this model each of the $n(n-1)$ arcs has probability p of being selected). There are also many other explicit constructions of digraphs with $\nu_c(G) = \Theta(n^2)$ which do not resemble a typical element of $G(n,p)$. A construction showing that the $o(n^2)$ error term in Theorem 1 **cannot** be replaced with $o(n^{1.5})$ will be given in the journal version of this paper.

Theorem 1 uses as its main lemma a much more general result concerning packings of oriented graphs. Recall that an oriented graph is a directed graph without 2-cycles. Let $\mathcal{F}$ be any given (finite or infinite) family of oriented graphs. For an oriented graph G, let $\nu_{\mathcal{F}}(G)$ denote the maximum number of arc-disjoint copies of elements of $\mathcal{F}$ that can be found in G, and let $\nu_{\mathcal{F}}^*(G)$ denote the respective fractional relaxation. We prove the following.

Theorem 2. *For any given family of oriented graphs, if G is an n-vertex oriented graph then $\nu_{\mathcal{F}}^*(G) - \nu_{\mathcal{F}}(G) = o(n^2)$. Furthermore, a set of at least $\nu_{\mathcal{F}}(G) - o(n^2)$ arc-disjoint elements of $\mathcal{F}$ can be generated in randomized polynomial time.*

Theorem 1 is a consequence of Theorem 2 by considering the family $\mathcal{F}$ of all directed cycles of length at least 3. An initial preprocessing step allows us to get rid of the 2-cycles of G.

We note that an *undirected* version of Theorem 2 has been recently proved by the second author [17] extending and simplifying an earlier result of Haxell and Rödl [11] dealing with single element families. The proof of Theorem 2 makes use of the recently discovered *directed regularity lemma* which has been proved by Alon and Shapira in [2], and which enables us to overcome several difficulties that do not occur in the undirected case.

It is well known that $\nu_c^*(G,w)$ can be computed in polynomial time by considering the dual problem whose solution is known to be computable in polynomial time [13]. This follows from the strong duality theorem. However notice that this does not imply that an optimal fractional dicycle packing ψ yielding $\nu_c^*(G,w)$ can also be generated in polynomial time and this was an open problem (see, e.g. [10, p. 294]). We prove the following result.

Theorem 3. *If G is an n-vertex digraph associated with a nonnegative arc-weight function w, then a maximum fractional dicycle packing yielding $\nu_c^*(G,w)$ can be computed in polynomial time. Furthermore, a maximum fractional dicycle packing with at most $O(n^2)$ (resp. $O(n^3)$) dicycles receiving nonzero weight can be found in (resp. strongly) polynomial time.*

In the next two sections we prove our results.

2 Proofs of Theorem 1 and Theorem 2

2.1 Reducing Theorem 1 to a Special Case of Theorem 2

The following simple lemma shows that the problem of finding a maximum dicycle packing in a digraph G is equivalent to the problem of finding a maximum dicycle packing in the spanning subgraph G' of G obtained from G by deleting all 2-cycles. In particular, it shows that if $\nu_c^*(G') - \nu_c(G') = o(n^2)$ then $\nu_c^*(G) - \nu_c(G) = o(n^2)$.

Lemma 4 *If G is a digraph then there is always a maximum dicycle packing of G that contains all the 2-cycles.* □

The proof of Lemma 4 is an easy exercise. Notice that Lemma 4 together with Theorem 2 applied to the family of all directed cycles of length at least 3 yields Theorem 1.

2.2 Tools Used in the Proof of Theorem 2

An important tool used in the proof of Theorem 2 is the following directed version of Szemerédi's regularity lemma. The proof, which is a modified version of the proof of the standard regularity lemma given in [15], can be found in [2].

We now give the definitions necessary in order to state the directed regularity lemma.

Let $G = (V, E)$ be a digraph, and let A and B be two disjoint subsets of $V(G)$. If A and B are non-empty and $e(A, B)$ is the number of arcs from A to B, the *density of arcs from A to B* is

$$d(A, B) = \frac{e(A, B)}{|A||B|}.$$

For $\gamma > 0$ the pair (A, B) is called γ-*regular* if for every $X \subset A$ and $Y \subset B$ satisfying $|X| > \gamma|A|$ and $|Y| > \gamma|B|$ we have

$$|d(X, Y) - d(A, B)| < \gamma \qquad |d(Y, X) - d(B, A)| < \gamma.$$

An *equitable partition* of a set V is a partition of V into pairwise disjoint classes $V_1, \ldots, V_m$ whose sizes are as equal as possible. An equitable partition of the set of vertices V of a digraph G into the classes $V_1, \ldots, V_m$ is called γ-*regular* if $|V_i| \leq \gamma|V|$ for every i and all but at most $\gamma\binom{m}{2}$ of the pairs (V_i, V_j) are γ-regular. The directed regularity lemma states the following:

Lemma 5 *For every $\gamma > 0$, there is an integer $M(\gamma) > 0$ such that every digraph G with $n > M$ vertices has a γ-regular partition of the vertex set into m classes, for some $1/\gamma \leq m \leq M$.* □

Let H_0 be a fixed oriented graph with the vertices $\{1, \ldots, k\}$, $k \geq 3$. Let W be a k-partite oriented graph with vertex classes $V_1, \ldots, V_k$. A subgraph J of W with ordered vertex set $v_1, \ldots, v_k$ is *partite-isomorphic* to H_0 if $v_i \in V_i$ and the map $v_i \to i$ is an isomorphism from J to H_0. The following lemma is almost identical to the proof of Lemma 15 in [11] and hence the proof is omitted.

Lemma 6 *Let δ and ζ be positive reals. There exist $\gamma = \gamma(\delta, \zeta, k)$ and $T = T(\delta, \zeta, k)$ such that the following holds. Let W be a k-partite oriented graph with vertex classes $V_1, \ldots, V_k$ and $|V_i| = t > T$ for $i = 1, \ldots, k$. Furthermore, for each arc $(i, j) \in E(H_0)$, (V_i, V_j) is a γ-regular pair with density $d(i, j) \geq \delta$ and for each arc $(i, j) \notin E(H_0)$, $E(V_i, V_j) = \emptyset$. Then, there exists a spanning subgraph W' of W, consisting of at least $(1-\zeta)|E(W)|$ arcs such that the following holds. For an arc $e \in E(W')$, let $c(e)$ denote the number of subgraphs of W' that are partite isomorphic to H_0 and that contain e. Then, for all $e \in E(W')$, if $e \in E(V_i, V_j)$ then*

$$\left| c(e) - t^{k-2} \frac{\prod_{(s,p) \in E(H_0)} d(s, p)}{d(i, j)} \right| < \zeta t^{k-2}.$$

□

Finally, we need to state the seminal result of Frankl and Rödl [7] on near perfect coverings and matchings of uniform hypergraphs. Recall that if x, y are two vertices of a hypergraph then $deg(x)$ denotes the degree of x and $deg(x,y)$ denotes the number of hyperedges that contain both x and y (their *co-degree*). We use the stronger version of the Frankl and Rödl Theorem due to Pippenger (see, e.g., [8]).

Lemma 7 *For an integer $r \geq 2$ and a real $\beta > 0$ there exists a real $\mu > 0$ so that: If the r-uniform hypergraph L on q vertices has the following properties for some d:*
(i) $(1-\mu)d < deg(x) < (1+\mu)d$ holds for all vertices,
(ii) $deg(x,y) < \mu d$ for all distinct x and y,
then L has a matching of size at least $(q/r)(1-\beta)$. □

2.3 Proof of Theorem 2

Let $\mathcal{F}$ be a family of oriented graphs, and let $\epsilon > 0$. To avoid the trivial case we assume that each element of $\mathcal{F}$ has at least three vertices. We shall prove there exists $N = N(\mathcal{F}, \epsilon)$ such that for all $n > N$, if G is an n-vertex oriented graph then $\nu^*_{\mathcal{F}}(G) - \nu_{\mathcal{F}}(G) < \epsilon n^2$.

The idea of the proof is as follows. Given an n-vertex graph G and a maximum fractional $\mathcal{F}$-packing ψ of G, we apply Lemma 5 to G and define a fractional $\mathcal{F}$-packing ψ' on the resulting m-vertex cluster graph of the partition (the graph whose vertices are the vertex classes of the partition and whose arcs connect appropriately defined dense pairs of vertex classes). We show that $|\psi'|$ is very close to $|\psi|m^2/n^2$. We then show how each fractional copy H of ψ' can be translated back into an appropritate number of (integral) edge-disjoint copies of H in G, such that the total number of copies constructed in G in this way is very close to $|\psi'|n^2/m^2$, and hence to $|\psi|$.

Let k_∞ be the maximal order of a graph in $\mathcal{F}$ (possibly $k_\infty = \infty$). Let $k_0 = \min\{k_\infty, \lceil 20/\epsilon \rceil\}$. Let $\delta = \beta = \epsilon/4$. For all $r = 2, \ldots, {k_0}^2$, let $\mu_r = \mu(\beta, r)$ be as in Lemma 7, and put $\mu = \min_{r=2}^{{k_0}^2}\{\mu_r\}$. Let $\zeta = \mu\delta^{{k_0}^2}/2$. For $k = 3, \ldots, k_0$, let $\gamma_k = \gamma(\delta, \zeta, k)$ and $T_k = T(\delta, \zeta, k)$ be as in Lemma 6. Let $\gamma = \min_{k=3}^{k_0}\{\gamma_k\}$. Let $M = M(\gamma\epsilon/(25{k_0}^2))$ be as in Lemma 5. Finally, we shall define N to be a sufficiently large constant, depending on the above chosen parameters, and for which various conditions stated in the proof below hold. Thus, indeed, $N = N(\mathcal{F}, \epsilon)$.

Fix an n-vertex oriented graph G with $n > N$ vertices. Fix a fractional $\mathcal{F}$-packing ψ with $|\psi| = \nu^*_{\mathcal{F}}(G)$. We may assume that ψ assigns a value to each *labeled* copy of an element of $\mathcal{F}$ simply by dividing the value of ψ on each nonlabeled copy by the size of the automorphism group of that element. If $\nu^*_{\mathcal{F}}(G) < \epsilon n^2$ we are done. Hence, we assume $\nu^*_{\mathcal{F}}(G) = \alpha n^2 \geq \epsilon n^2$.

We apply Lemma 5 to G and obtain a γ'-regular partition with m' parts, where $\gamma' = \gamma\epsilon/(25{k_0}^2)$ and $1/\gamma' < m' < M(\gamma')$. Denote the parts by $U_1, \ldots, U_{m'}$. Notice that the size of each part is either $\lfloor n/m' \rfloor$ or $\lceil n/m' \rceil$. For simplicity we

may and will assume that n/m' is an integer, as this assumption does not affect the asymptotic nature of our result. For the same reason we may and will assume that $n/(25m'k_0^2/\epsilon)$ is an integer.

We randomly partition each U_i into $25k_0^2/\epsilon$ equal parts of size $n/(25m'k_0^2/\epsilon)$ each. All m' partitions are independent. We now have $m = 25m'k_0^2/\epsilon$ *refined* vertex classes, denoted $V_1, \ldots, V_m$. Suppose $V_i \subset U_s$ and $V_j \subset U_t$ where $s \neq t$. We claim that if (U_s, U_t) is a γ'-regular pair then (V_i, V_j) is a γ-regular pair. Indeed, if $X \subset V_i$ and $Y \subset V_j$ have $|X|, |Y| > \gamma n/(25m'k_0^2/\epsilon)$ then $|X|, |Y| > \gamma' n/m'$ and so $|d(X,Y) - d(U_s,U_t)| < \gamma'$ and $|d(Y,X) - d(U_t,U_s)| < \gamma'$. Also $|d(V_i,V_j) - d(U_s,U_t)| < \gamma'$ and $|d(V_j,V_i) - d(U_t,U_s)| < \gamma'$. Thus, $|d(X,Y) - d(V_i,V_j)| < 2\gamma' < \gamma$ and $|d(Y,X) - d(V_j,V_i)| < 2\gamma' < \gamma$.

Let H be a labeled copy of some $H_0 \in \mathcal{F}$ in G. If H has k vertices and $k \leq k_0$ then the expectation of the number of pairs of vertices of H that belong to the same vertex class in the refined partition is clearly at most $\binom{k}{2}\epsilon/(25k_0^2) < \epsilon/50$. Thus, the probability that H has two vertices in the same vertex class is also at most $\epsilon/50$. We call H *good* if it has $k \leq k_0$ vertices and its k vertices belong to k distinct vertex classes of the refined partition. By the definition of k_0, if H has $k > k_0$ vertices and $\psi(H) > 0$ then we must have $k > 20/\epsilon$. Since oriented graphs with k vertices have at least $k/2$ arcs, the contribution of graphs with $k > k_0$ vertices to $\nu^*_{\mathcal{F}}(G)$ is at most $\binom{n}{2}/(10/\epsilon) < \epsilon n^2/20$. Hence, if ψ^{**} is the restriction of ψ to good copies (the bad copies having $\psi^{**}(H) = 0$) then the expectation of $|\psi^{**}|$ is at least $(\alpha - \epsilon/50 - \epsilon/20)n^2$. We therefore *fix* a partition $V_1, \ldots, V_m$ for which $|\psi^{**}| \geq (\alpha - 0.07\epsilon)n^2$.

We say that the set of arcs $E(V_i, V_j)$ is *good* if (V_i, V_j) is a γ-regular pair and also $d(V_i, V_j) \geq \delta$. Notice that it is possible that $E(V_i, V_j)$ is good while $E(V_j, V_i)$ is not good (because of sparseness). Let G^* be the spanning subgraph of G consisting of the union of the good sets of arcs (thus, we discard arcs inside classes, between non regular pairs, between sparse pairs, or one-sided sparse pairs). Let ψ^* be the restriction of ψ^{**} to the labeled copies of elements of $\mathcal{F}$ in G^*. We claim that $\nu^*_{\mathcal{F}}(G^*) \geq |\psi^*| > |\psi^{**}| - 1.02\delta n^2 \geq (\alpha - 0.07\epsilon - 1.02\delta)n^2 = (\alpha - 1.3\delta)n^2$. Indeed, by considering the number of discarded arcs we get (using $m' > 1/\gamma'$ and $\delta >> \gamma'$)

$$|\psi^{**}| - |\psi^*| \leq |E(G) - E(G^*)| <$$

$$\gamma' \binom{m'}{2} \frac{2n^2}{m'^2} + m'(m'-1)(\delta + \gamma')\frac{n^2}{m'^2} + m'\frac{n^2}{m'^2} < 1.02\delta n^2.$$

Let R denote the m-vertex digraph whose vertices are $\{1, \ldots, m\}$ and $(i,j) \in E(R)$ if and only if $E(V_i, V_j)$ is good. We define a (labeled) fractional dicycle packing ψ' of R as follows. Let H be a labeled copy of some $H_0 \in \mathcal{F}$ in R and assume that the vertices of H are $\{u_1, \ldots, u_k\}$ where u_i plays the role of vertex i in H_0. We define $\psi'(H)$ to be the sum of the values of ψ^* taken over all subgraphs of $G^*[V_{u_1}, \ldots, V_{u_k}]$ which are partite isomorphic to H_0, divided by n^2/m^2. Notice that by normalizing with n^2/m^2 we guarantee that ψ' is a proper fractional $\mathcal{F}$-packing of R and that $\nu^*_{\mathcal{F}}(R) \geq |\psi'| = m^2|\psi^*|/n^2 \geq m^2(\alpha - 1.3\delta)$.

Notice also that although R may contain 2-cycles, they receive no weight in ψ' as G has no 2-cycles.

We use ψ' to define a random coloring of the arcs of G^*. Our "colors" are the labeled copies of elements of $\mathcal{F}$ in R. Let $d(i,j)$ denote the density from V_i to V_j and notice that $|E_{G^*}(V_i,V_j)| = d(i,j)n^2/m^2$. Let H be a labeled copy of some $H_0 \in \mathcal{F}$ in R, and assume that H contains the arc (i,j). Each $e \in E(V_i,V_j)$ is chosen to have the "color" H with probability $\psi'(H)/d(i,j)$. The choices made by distinct arcs of G^* are independent. Notice that this random coloring is legal (in the sense that the sum of probabilities is at most one) since the sum of $\psi'(H)$ taken over all labeled copies of elements of $\mathcal{F}$ containing (i,j) is at most $d(i,j)$. Notice also that some arcs might stay uncolored.

Let H be a labeled copy of some $H_0 \in \mathcal{F}$ in R, and assume that $\psi'(H) > m^{1-k_0}$. Without loss of generality, assume that the vertices of H are $\{1,\ldots,k\}$ where $i \in V(H)$ plays the role of $i \in V(H_0)$. Let r denote the number of arcs of H. Notice that $r < k_0^2$. Let W_H be the k-partite subgraph of G^* with vertex classes $V_1,\ldots,V_k$ and with the arcs $\cup_{(i,j)\in H}E(V_i,V_j)$. Notice that W_H satisfies the conditions in Lemma 6, since $t = n/m > N\epsilon/(25{k_0}^2M) > T_k$ (here we assume $N > 25{k_0}^2MT_k/\epsilon$). Let W'_H be the spanning subgraph of W_H whose existence is guaranteed in Lemma 6. Let X_H denote the spanning subgraph of W'_H consisting only of the arcs whose color is H. Notice that X_H is a random subgraph of W'_H. For an arc $e \in E(X_H)$, let $C_H(e)$ denote the set of subgraphs of X_H that contain e and that are partite isomorphic to H_0. Put $c_H(e) = |C_H(e)|$. The following lemma, shows that for all $e \in E(X_C)$, $c_H(e)$ can be tightly approximated with high probability. As the proof is quite long, it is omitted from this extended abstract and will be given in the final version of this paper.

Lemma 8 *With probability at least* $1 - m^3/n$, *for all* $e \in E(X_H)$,

$$\left|c_H(e) - t^{k-2}\psi'(H)^{r-1}\right| < \mu\psi'(H)^{r-1}t^{k-2}. \tag{1}$$

We also need the following simple lemma that gives a lower bound for the number of arcs of X_H.

Lemma 9 *With probability at least* $1 - 1/n$,

$$|E(X_H)| > (1-2\zeta)r\frac{n^2}{m^2}\psi'(H).$$

Proof: For $(i,j) \in E(H_0)$, the expected number of arcs of $E(V_i,V_j)$ that received the color H is precisely $d(i,j)\frac{n^2}{m^2}\frac{\psi'(H)}{d(i,j)} = \frac{n^2}{m^2}\psi'(H)$. Summing over all r arcs of H_0, the expected number of arcs of W_H that received the color H is precisely $r\frac{n^2}{m^2}\psi'(H)$. As at most $\zeta|E(W_H)|$ arcs belong to W_H and do not belong to W'_H we have that the expectation of $|E(X_H)|$ is at least $(1-\zeta)r\frac{n^2}{m^2}\psi'(H)$. As ζ, r, m are constants and as $\psi'(H)$ is bounded from below by the constant m^{1-k_0}, we have, by the common large deviation inequality of Chernoff (cf. [3] Appendix A), that for $n > N$ sufficiently large, the probability that $|E(X_H)|$

deviates from its mean by more than $\zeta r\frac{n^2}{m^2}\psi'(H)$ is exponentially small in n. In particular, the lemma follows. □

Since R contains at most $O(m^{k_0})$ labeled copies of elements of $\mathcal{F}$ with at most k_0 vertices, we have that with probability at least $1-O(m^{k_0}/n)-O(m^{k_0+3}/n) > 0$ (here we assume again that N is sufficiently large) *all* labeled copies H of elements of $\mathcal{F}$ in R with $\psi'(H) > m^{1-k_0}$ satisfy the statements of Lemma 8 and Lemma 9. We therefore fix a coloring for which Lemma 8 and Lemma 9 hold for all labeled copies H of elements of $\mathcal{F}$ in R having $\psi'(H) > m^{1-k_0}$.

Let H be a labeled copy of some $H_0 \in \mathcal{F}$ in R with $\psi'(H) > m^{1-k_0}$, and let r denote the number of arcs of H. We construct an r-uniform hypergraph L_H as follows. The vertices of L_H are the arcs of the corresponding X_H from Lemma 8. The hyperedges of L_H correspond to the arc sets of the subgraphs of X_H that are partite isomorphic to H_0. We claim that our hypergraph satisfies the conditions of Lemma 7. Indeed, let q denote he number of vertices of L_H. Notice that Lemma 9 provides a lower bound for q. Let $d = t^{k-2}\psi'(H)^{r-1}$. Notice that by Lemma 8 *all* vertices of L_H have their degrees between $(1-\mu)d$ and $(1+\mu)d$. Also notice that the co-degree of any two vertices of L_H is at most t^{k-3} as two arcs cannot belong, together, to more than t^{k-3} subgraphs of X_H that are partite isomorphic to H_0. In particular, for N sufficiently large, $\mu d > t^{k-3}$. By Lemma 7 we have at least $(q/r)(1-\beta)$ arc-disjoint copies of H_0 in X_H. In particular, we have at least

$$(1-\beta)(1-2\zeta)\frac{n^2}{m^2}\psi'(H) > (1-2\beta)\psi'(H)\frac{n^2}{m^2}$$

such copies. Recall that $|\psi'| \geq m^2(\alpha - 1.3\delta)$. Since there are $O(m^{k_0})$ labeled copies H of elements of $\mathcal{F}$ in R with $0 < \psi'(H) \leq m^{1-k_0}$, their total contribution to $|\psi'|$ is $O(m)$. Hence, summing the last inequality over all H with $\psi'(H) > m^{1-k_0}$ we have at least

$$(1-2\beta)m^2(\alpha - 1.3\delta - O(\frac{1}{m}))\frac{n^2}{m^2} > n^2(\alpha-\epsilon)$$

arc-disjoint copies of elements of $\mathcal{F}$ in G. It follows that $\nu_{\mathcal{F}}(G) \geq n^2(\alpha-\epsilon)$. As $\nu^*_{\mathcal{F}}(G) = \alpha n^2$, Theorem 3 follows. □

The proof of Theorem 2 implies a polynomial (in n) time algorithm that produces a set of $n^2(\alpha-\epsilon)$ arc-disjoint elements of $\mathcal{F}$ in G with probability at least, say, 0.99. Indeed, Lemma 5 can be implemented in $o(n^3)$ time using the algorithm of Alon et. al. [1] applied to the directed regularity lemma. Lemma 6 can be implemented using a simple greedy algorithm following the proof in [11]. Lemma 7 has a polynomial running time implementation due to Grable [9]. By Theorem 3, computing ψ and ψ^{**} can be done in $O(n^3)$ time. The other ingredients of the proof, namely, computing ψ' and the random coloring are easily implemented in polynomial time.

3 Computing a Fractional Dicycle Packing in Polynomial Time

The main difficulty in proving Theorem 3 stems from the fact that the LP-formulation of the problem might have an exponential number of variables since, in general, the number of dicycles in a digraph might be exponential in the size of the digraph.

Let ψ be a fractional dicycle packing of a weighted digraph (G, w). Notice that it is sufficient to specify the values of ψ only for dicycles with $\psi(C) > 0$. Thus, if $\mathcal{C}(G)$ denotes the set of dicycles in G, let $\chi(\psi) = \{C \in \mathcal{C}(G) : \psi(C) > 0\}$ denote the *characteristic set* of ψ. An algorithm for the maximum dicycle packing problem is *polynomial* if it runs in time polynomial in the size of the input digraph (G, w), and delivers a polynomial size set of dicycles $\mathcal{C} \subseteq \mathcal{C}(G)$ and nonnegative numbers $\psi(C)$ for $C \in \mathcal{C}$, such that by assigning $\psi(C) = 0$ to all $C \in \mathcal{C}(G) - \mathcal{C}$ we obtain an optimal fractional dicycle packing of (G, w), namely $|\psi| = \nu_c^*(G, w)$.

A *fractional dicycle arc cover* of (G, w) is a function $x : E(G) \to [0, 1]$, such that $\sum_{e \in C} x(e) \geq 1$ for every $C \in \mathcal{C}(G)$. The *value* of a fractional dicycle arc cover x is $|x| = \sum_{e \in E(G)} w(e)x(e)$. Let $\tau_c^*(G, w)$ denote the value of a minimum fractional dicycle arc cover of (G, w). The *minimum fractional dicycle arc cover problem* is a dual to the maximum fractional dicycle packing problem.

For the rest of this section we will assume that G is the complete digraph (the digraph in which any pair of distinct vertices are connected with arcs in both directions). We may assume this since if G is not complete, we may add the nonexistent arcs and assign them zero weight. Notice that $\nu_c^*(G, w)$ and $\tau_c^*(G, w)$ remain intact after this modification.

It was shown in [13] that the minimum fractional dicycle arc cover problem is reduced to solving a linear program with a polynomial number of constraints. Let $\mathcal{C}_k(G)$ denote the set of all the dicycles in G of length k. The result in [13] shows that in order to solve the fractional dicycle arc cover problem in a complete digraph, it is sufficient to consider only dicycles of length 2 and 3.

Theorem 10 ([13],Theorem 4.1). *If (G, w) is a complete weighted digraph then*

$$\begin{aligned} \tau_c^*(G, w) = \min & \sum_{e \in E(G)} w(e)x(e) & (2) \\ s.t. \quad & \sum_{e \in C} x(e) \geq 1 \quad \text{for all } C \in \mathcal{C}_3(G) \\ & \sum_{e \in C} x(e) = 1 \quad \text{for all } C \in \mathcal{C}_2(G) \\ & x(e) \geq 0 \quad \text{for all } e \in E(G). \end{aligned}$$

Proof of Theorem 3: Using Theorem 10 we will show that a maximum fractional dicycle packing with $O(n^2)$ (resp. $O(n^3)$) dicycles in its characteristic set, can be found in (resp., strongly) polynomial time. Consider the dual linear program to (2). By the strong duality theorem:

$$\begin{aligned} \nu_c^*(G,w) = \quad & \max \textstyle\sum_{C\in\mathcal{C}_2(G)\cup\mathcal{C}_3(G)} \psi(C) \\ & s.t. \textstyle\sum_{e\in C\in\mathcal{C}_2(G)\cup\mathcal{C}_3(G)} \psi(C) \le w(e) \text{ for all } e \in E(G) \\ & \psi(C) \ge 0 \qquad\qquad \text{for all } C \in \mathcal{C}_3(G). \end{aligned} \tag{3}$$

Note that a feasible solution ψ to (3) might not be a fractional dicycle packing, since there might be a dicycle $C \in \mathcal{C}_2(G)$ for which $\psi(C) < 0$ holds. A *fractional dicycle pseudopacking* is a fractional dicycle packing in which 2-cycles may receive negative values. In particular, a feasible solution of (3) is a fractional dicycle pseudopacking. We will show a polynomial time iterative procedure that converts an *optimal* solution ψ of (3) into a fractional dicycle packing of value $|\psi| = \nu_c^*(G,w)$, and such that the cardinality of the characteristic set is at most $|\chi(\psi)|$.

For a pair of vertices u, v, let $C_{uv} = C_{vu}$ denote the 2-cycle consisting of the two arcs $(u,v),(v,u)$. For a fractional dicycle pseudopacking ψ, let $\Delta_\psi(u,v) = w(u,v) - \sum_{(u,v)\in C} \psi(C)$. Notice that $\Delta_\psi(u,v) \ge 0$ for each $(u,v) \in E(G)$. Notice also that if ψ is an optimal solution of (3) then $\min\{\Delta_\psi(u,v)\ ,\ \Delta_\psi(v,u)\} = 0$. Our algorithm consists of repeated applications of one of the following two basic operations.

Operation *A*: Let ψ be a fractional dicycle pseudopacking with $|\psi| = \nu_c^*(G,w)$. Suppose that for $s,t \in V(G)$ holds: $\psi(C_{st}) < 0$, $\Delta_\psi(t,s) = 0$ and $\Delta_\psi(s,t) > 0$. Let C_1 be a dicycle such that $(t,s) \in C_1$ and $\psi(C_1) > 0$. Notice that C_1 must exist. Let $\delta = \min\{\Delta_\psi(s,t)\ ,\ \psi(C_1)\}$. Let $\hat{\psi}$ be the same as ψ except for $\hat{\psi}(C_1) = \psi(C_1) - \delta$ and $\hat{\psi}(C_{st}) = \psi(C_{st}) + \delta$. Notice that $\hat{\psi}$ is a fractional dicycle pseudopacking with $|\hat{\psi}| = |\psi| = \nu_c^*(G,w)$. Furthermore $|\chi(\hat{\psi})| \le |\chi(\psi)|$ and if $\Delta_{\hat{\psi}}(s,t) > 0$, then $|\chi(\hat{\psi})| \le |\chi(\psi)| - 1$.

Operation *B*: Let ψ be a fractional dicycle pseudopacking with $|\psi| = \nu_c^*(G,w)$. Suppose that for $u,v \in V(G)$ holds: $\psi(C_{uv}) < 0$ and $\Delta_\psi(u,v) = \Delta_\psi(v,u) = 0$. Let C_1, C_2 be dicycles such that $(u,v) \in C_1$, $(v,u) \in C_2$, and let $\mu = \min\{\psi(C_1)\ ,\ \psi(C_2)\} > 0$. Notice that C_1 and C_2 exist. Let $\hat{C}$ be a dicycle in $(C_1 \cup C_2) - C_{uv}$. Let $\hat{\psi}$ be the same as ψ except for $\hat{\psi}(C_i) = \psi(C_i) - \mu$ for $i = 1,2$, $\hat{\psi}(\hat{C}) = \psi(\hat{C}) + \mu$, $\hat{\psi}(C_{uv}) = \psi(C_{uv}) + \mu$. Notice that $\hat{\psi}$ is a fractional dicycle pseudopacking with $|\hat{\psi}| = |\psi| = \nu_c^*(G,w)$. Furthermore, $\Delta_{\hat{\psi}}(u,v) = \Delta_{\hat{\psi}}(v,u) = 0$, $|\chi(\hat{\psi})| \le |\chi(\psi)|$, and $|\{C \in \chi(\hat{\psi}) \ :\ (u,v) \in C \text{ or } (v,u) \in C\}| \le |\{C \in \chi(\psi) \ :\ (u,v) \in C \text{ or } (v,u) \in C\}| - 1$.

Our algorithm proceeds as follows. We begin with a fractional dicycle pseudopacking ψ which is an optimal solution of (3). At each stage we apply either Operation *A* or Operation *B*. Thus, at any stage we have a fractional dicycle pseudopacking $\hat{\psi}$ with $|\chi(\hat{\psi})| \le |\chi(\psi)|$ and with $|\hat{\psi}| = |\psi| = \nu_c^*(G,w)$. At any stage, let $\mathcal{S}'$ denote the set of dicycles C with $\hat{\psi}(C) < 0$ and let $\mathcal{S}$ denote the set of cycles with $\hat{\psi}(C) > 0$. Notice that operations *A* and *B* guarantee that $\mathcal{S}'$ contains only 2-cycles. In the beginning, $\mathcal{S} \subset \mathcal{C}_2(G) \cup \mathcal{C}_3(G)$ but Operation *B* may add longer cycles to $\mathcal{S}$. Notice that in operations *A* and *B* we can increase the weight of a 2-cycle without decreasing the weight of any other 2-cycle. In the main loop, as long as $\mathcal{S}' \neq \emptyset$, the algorithm chooses $C_{uv} \in \mathcal{S}'$ and increases $\hat{\psi}(C_{uv})$ by $\min\{\Delta_{\hat{\psi}}(u,v)\ ,\ \Delta_{\hat{\psi}}(v,u)\}$. In case $\Delta_{\hat{\psi}}(u,v) = \Delta_{\hat{\psi}}(v,u) = 0$ holds,

the algorithm iteratively applies Operation B until $\hat{\psi}(C_{uv}) \geq 0$ holds. Otherwise, for $(s,t) = (u,v)$ or for $(s,t) = (v,u)$, $\Delta_{\hat{\psi}}(s,t) > 0$ and $\Delta_{\hat{\psi}}(t,s) = 0$ holds. Then, the algorithm iteratively applies Operation A for an appropriate choice among $(s,t) = (u,v)$ or $(s,t) = (v,u)$, until $\hat{\psi}(C_{uv}) \geq 0$, or until $\Delta_{\hat{\psi}}(u,v) = \Delta_{\hat{\psi}}(v,u) = 0$ holds. During the operations, the algorithm updates the sets $\mathcal{S}'$ and $\mathcal{S}$. It is not hard to see, that every iteration of the main loop decreases $|\mathcal{S}'|$ by at least 1. Thus, at the end of the algorithm, $\hat{\psi}$ is an optimal fractional dicycle packing. As each inner loop is repeated at most $|\mathcal{S}|$ times, the algorithm is polynomial in the initial values of $|\mathcal{S}'|$, $|\mathcal{S}|$, and in $n = |V(G)|$.

It is known that any linear program of the form $\max\{y \in R^m : Qy \leq q\}$ which has an optimal solution, has an optimal solution $\tilde{y}$ which is *basic*. That is, there exists a set of m equations and tight inequalities, such that $\tilde{y}$ is the unique solution to the corresponding equation system. Let $\tilde{\psi}$ be a basic solution to (3). The linear program (3) has $O(n^3)$ variables and $O(n^3)$ constraints. Note, however, that only $O(n^2)$ of the variables do *not* have constraints of the form $\psi(C) \geq 0$. Thus, $\tilde{\psi}$ has only $O(n^2)$ nonzero entries, i.e., $|\chi(\tilde{\psi})| = O(n^2)$. One can compute an optimal basic solution to (3) in polynomial time using the interior point method.

Tardos [16] showed that there exists a strongly polynomial time algorithm for solving linear programs with $\{0, \pm 1\}$ constraint matrices, assuming all the other input numbers are rational. Thus, the linear program (3) can be solved in strongly polynomial time. However, the algorithm in [16] might produce an optimal solution $\tilde{\psi}$ which is not basic. In this case, we can guarantee only that $|\chi(\tilde{\psi})| = O(n^3)$. □

Similar techniques can be used to show that there always exists a compact maximum *integral* dicycle packing. Let $\nu_c(G,w)$ denote the value of a maximum integral dicycle packing in (G,w).

Lemma 11 *Let (G,w) be a complete weighted digraph, and let $\tilde{\nu}_c(G,w)$ be the value of an optimal integral solution to (3). Then $\tilde{\nu}_c(G,w) = \nu_c(G,w)$.*

Proof: We can assume, w.l.o.g., that the weights are integral. Then the algorithm described above preserves integrality. Thus $\tilde{\nu}_c(G,w) \leq \nu_c(G,w)$. We now prove the reverse inequality. Let ψ be an optimal integral solution to (3). Among all integral dicycle pseudopackings in (G,w), let $\tilde{\psi}$ be one for which $\sum_{C \in \mathcal{C}_3(G) \cap \chi(\psi)} \psi(C)$ is maximal. We claim that $\chi(\tilde{\psi}) \subseteq \mathcal{C}_2(G) \cup \mathcal{C}_3(G)$. If not, then there is a dicycle $\hat{C} = \{(v_1,v_2),(v_2,v_3),\ldots,(v_{k-1},v_k),(v_k,v_1)\}$, such that $k \geq 4$, and $\tilde{\psi}(\hat{C}) \geq 1$. Let $C_1 = \{(v_1,v_2),(v_2,v_3),(v_3,v_1)\}$ and let $C_2 = \{(v_3,v_4),\ldots,(v_k,v_1),(v_1,v_3)\}$. Define $\hat{\psi}$ on $\chi(\tilde{\psi}) \cup \{C_1, C_2, C_{v_1v_3}\}$ as follows: $\hat{\psi}(C) = \tilde{\psi}(C) - 1$ if $C = C_{v_1v_3}$ or if $C = \hat{C}$, $\hat{\psi}(C) = \tilde{\psi}(C) + 1$ if $C = C_1$ or if $C = C_2$ and $\hat{\psi}(C) = \tilde{\psi}(C)$ otherwise. It is easy to see that $\hat{\psi}$ is a dicycle pseudopacking such that $\sum_{C \in \mathcal{C}_3(G) \cap \chi(\hat{\psi})} \psi(C) \geq 1 + \sum_{C \in \mathcal{C}_3(G) \cap \chi(\tilde{\psi})} \psi(C)$ contradicting our assumption. □

Corollary 12 *For any integrally weighted digraph* (G, w) *on* n *vertices, there exists a maximum integral dicycle packing in* G *of value* $\nu_c(G, w)$ *whose characteristic set contains* $O(n^3)$ *dicycles.*

References

1. N. Alon, R.A. Duke, H. Lefmann, V. Rödl and R. Yuster, *The algorithmic aspects of the Regularity Lemma,* Journal of Algorithms 16 (1994), 80–109.
2. N. Alon and A. Shapira, *Testing subgraphs in directed graphs,* Proc. 35th ACM STOC, ACM Press (2003), 700–709.
3. N. Alon and J. H. Spencer, **The Probabilistic Method,** *Second Edition,* Wiley, New York, 2000.
4. P. Balister, *Packing digraphs with directed closed trails,* Combin. Probab. Comput. 12 (2003), 1–15.
5. B. Bollobás, *Extremal Graph Theory,* Academic Press, 1978.
6. D. Dor and M. Tarsi, *Graph decomposition is NPC - A complete proof of Holyer's conjecture,* Proc. 20th ACM STOC, ACM Press (1992), 252–263.
7. P. Frankl and V. Rödl, *Near perfect coverings in graphs and hypergraphs,* European J. Combinatorics 6 (1985), 317–326.
8. Z. Füredi, *Matchings and covers in hypergraphs,* Graphs and Combinatorics 4 (1988), 115–206.
9. D. Grable, *Nearly-perfect hypergraph packing is in NC,* Information Processing Letters 60 (1996), 295–299.
10. M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization,* Springer-Verlag, Berlin, 1988.
11. P. E. Haxell and V. Rödl, *Integer and fractional packings in dense graphs,* Combinatorica 21 (2001), 13–38.
12. M. Krivelevich, Z. Nutov and R. Yuster, *Approximation algorithms for cycle packing problems,* preprint.
13. Z. Nutov and M. Penn, *On the integral dicycle packings and covers and the linear ordering polytope,* Discrete Applied Math. 60 (1995), 293–309.
14. P.D. Seymour, *Packing directed circuits fractionally,* Combinatorica 15 (1995), 281–288.
15. E. Szemerédi, Regular partitions of graphs, in: *Proc. Colloque Inter. CNRS* 260, CNRS, Paris, 1978, 399–401.
16. É. Tardos, *A strongly polynomial algorithm to solve combinatorial linear programs,* Operations Research 34 (1986), 250–256.
17. R. Yuster, *Integer and fractional packing of families of graphs,* Random Structures and Algorithms, to appear.

The Complexity of Membership Problems for Circuits over Sets of Integers

Stephen D. Travers

Theoretische Informatik
Bayerische Julius-Maximilians-Universität Würzburg
Am Hubland, D-97074 Würzburg, Germany
travers@informatik.uni-wuerzburg.de

Abstract. We investigate the complexity of membership problems for $\{\cup, \cap, {}^-, +, \times\}$-circuits computing sets of integers. These problems are a natural modification of the membership problems for circuits computing sets of natural numbers studied by McKenzie and Wagner (2003). We show that there are several membership problems for which the complexity in the case of integers differs significantly from the case of the natural numbers: Testing membership in the subset of integers produced at the output of a $\{\cup, +, \times\}$-circuit is NEXPTIME-complete, whereas it is PSPACE-complete for the natural numbers. As another result, evaluating $\{{}^-, +\}$-circuits is shown to be P-complete for the integers and PSPACE-complete for the natural numbers. The latter result extends work by McKenzie and Wagner (2003) in nontrivial ways. Furthermore, evaluating $\{\times\}$-circuits is shown to be $\mathrm{NL} \wedge \oplus\mathrm{L}$-complete, and several other cases are resolved.

1 Introduction

In complexity theory, combinational circuits play an important role. There is a variety of different kinds of circuits, and those over the boolean semiring certainly belong to the best investigated. Circuits over more general algebraic structures can be considered as well. A circuit over a universe U with operations $o_1, \ldots, o_r$ defined on the elements of U describes a way to compute an element $a \in U$.

In this paper, we study circuits over the power set of the integers. Besides the arithmetical gates + and ×, which compute set-theoretic addition and multiplication, we also allow gates computing the set operations $\cup, \cap$ and ${}^-$. The main question is the following: Given an integer b and a $\{\cup, \cap, {}^-, +, \times\}$-circuit C with integer inputs, does b belong to the set computed by C?
This is the membership problem $\mathrm{MC}_{\mathbb{Z}}(\cup, \cap, {}^-, +, \times)$ and $\mathrm{MF}_{\mathbb{Z}}(\cup, \cap, {}^-, +, \times)$ is the same problem restricted to formulas. The notion for other restricted versions of these problems, for example $\mathrm{MC}_{\mathbb{Z}}({}^-, +)$, is self-explanatory. The complexity of such membership problems varies considerably depending on the kinds of gates allowed in the circuit.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 322–333, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In 2003, McKenzie and Wagner investigated the complexity of analogously defined membership problems for circuits over sets of natural numbers, and presented their results at STACS 2003 [7]. We denote these problems with $\mathrm{MC}_{\mathbb{N}}(\ldots)$ and $\mathrm{MF}_{\mathbb{N}}(\ldots)$, respectively.

Changing the universe to the power set of the integers instead of the power set of the natural numbers, we investigate the complexity of these modified membership problems. As will be seen, for many of these new problems the complexity coincides with the corresponding problem in the $\mathbb{N}$-case. Interestingly, there also are several membership problems for which the complexity in the two cases differs significantly. We highlight here some of the differences.

Membership problem	$\mathbb{N}$-case ($X = \mathbb{N}$)	$\mathbb{Z}$-case ($X = \mathbb{Z}$)
$\mathrm{MC}_X(\cup, +, \times)$	PSPACE-complete	NEXPTIME-complete
$\mathrm{MC}_X(\times)$	NL-complete	NL$\wedge\oplus$L-complete
$\mathrm{MC}_X(^-, +)$	PSPACE-complete	P-complete
$\mathrm{MF}_X(^-, +)$	PSPACE-complete	L-complete

It turns out that it is not always possible to reduce the $\mathbb{N}$-case to the $\mathbb{Z}$-case, e.g. it holds that $\mathrm{MF}_{\mathbb{N}}(^-, +) \not\leq_m^{\log} \mathrm{MF}_{\mathbb{Z}}(^-, +)$ and $\mathrm{MC}_{\mathbb{N}}(^-, +) \leq_m^{\log} \mathrm{MC}_{\mathbb{Z}}(^-, +)$ if and only if P = PSPACE.

This paper is organized as follows: Section 2 gives the basic definitions and preliminaries. Section 3 is a digression to membership problems for circuits over sets of natural numbers. Since the problems $\mathrm{MF}_{\mathbb{N}}(^-, +)$ and $\mathrm{MC}_{\mathbb{N}}(^-, +)$ were omitted in [7], we provide a proof of their PSPACE-completeness. As an intermediate step, we introduce the problem *Quantified Sum of Subset* and prove that it is PSPACE-complete. In section 4, we analyze differences in complexity between membership problems in the $\mathbb{N}$-case and the $\mathbb{Z}$-case. Section 5 then presents several membership problems with the same complexity in both $\mathbb{N}$- and $\mathbb{Z}$-case. Finally, we conclude with several open problems. A summary of our results can be found in Table 1 at the end of the paper.

2 Preliminaries

We fix the alphabet $\Sigma = \{0, 1\}$. Σ^* is the set of words, and $|w|$ is the length of a word $w \in \Sigma^*$. We denote with L, NL, P, NP, coNP, PSPACE and NEXPTIME the standard complexity classes whose definitions can be found in any textbook on computational complexity (cf. [8], for example).
Furthermore, we need the function class #L and the complexity classes $\oplus$L and $\mathrm{C}_{=}\mathrm{L}$. For a nondeterministic logarithmic space machine M, define $\mathrm{acc}_M(x)$ as the number of accepting paths of M on input x. The class #L consists of precisely these functions. A set A is in $\oplus$L if there exists $f \in \#\mathrm{L}$ such that $x \in A \Leftrightarrow f(x) \equiv 1(2)$ for every $x \in \Sigma^*$. A set A is in $\mathrm{C}_{=}\mathrm{L}$ if there exist

$f, g \in \#\mathrm{L}$ such that $x \in A \Leftrightarrow f(x) = g(x)$ for every $x \in \Sigma^*$. See [1] for a survey on these counting classes. For complexity classes $\mathcal{K}$ and $\mathcal{M}$ we define $\mathcal{K} \wedge \mathcal{M} =_{\text{def}} \{A \cap B : A \in \mathcal{K}, B \in \mathcal{M}\}$. For sets A and B we say that A is many-one logspace reducible to B and write $A \leq_m^{\log} B$ if there exists a logarithmic space computable function f such that $x \in A \Leftrightarrow f(x) \in B$ for every $x \in \Sigma^*$.

$\mathbb{N}$ denotes the set of the natural numbers including 0, $\mathbb{Z}$ denotes the set of the integers. Like natural numbers, integers are represented in binary but have an additional sign; 0 does not have a sign. We denote the absolute value of an integer z with $\mathrm{abs}(z)$. For any integer b, let $\mathrm{sgn}(b) =_{\text{def}} 0$, if $b < 0$ and $\mathrm{sgn}(b) =_{\text{def}} 1$, otherwise.

A *circuit* $C = (G, E, g_C)$ is a finite, directed, acyclic graph *(G, E),* where G is the set of nodes and E is the set of vertices. The graph can contain multi-edges and does not necessarily have to be connected. As we consider circuits, we will call nodes *gates* from now on. C contains a specified gate g_C, the *output gate.* Gates with indegree 0 are called the *input gates.* Let $\mathcal{O} \subseteq \{\cup, \cap, \bar{}, +, \times\}$. An $\mathcal{O}$-circuit $C = (G, E, g_C, \alpha)$ is a circuit (C, E, g_C), whose gates have indegree 0, 1, or 2 and are labelled by the function $\alpha : G \to \mathcal{O} \cup \mathbb{Z}$ in the following way: Every input gate is labelled with an integer, every gate with indegree 1 is labelled with $\bar{}$, and every gate with indegree 2 with $\cup, \cap, +$, or $\times$. For each of its gates g, C computes a set $I(G) \subseteq \mathbb{Z}$, inductively defined as follows:

- If g is an input gate with $\alpha(g) = a$ then $I(g) =_{\text{def}} \{a\}$.
- If g is a $+$-gate with predecessors g_1, g_2 then $I(g) =_{\text{def}} \{a+b : a \in I(g_1) \wedge b \in I(g_2)\}$
- If g is a $\times$-gate with predecessors g_1, g_2 then $I(g) =_{\text{def}} \{a \cdot b : a \in I(g_1) \wedge b \in I(g_2)\}$.
- If g is a $\cup$-gate with predecessors g_1, g_2 then $I(g) =_{\text{def}} I(g_1) \cup I(g_2)$.
- If g is a $\cap$-gate with predecessors g_1, g_2 then $I(g) =_{\text{def}} I(g_1) \cap I(g_2)$.
- If g is a $\bar{}$-gate with predecessor g_1 then $I(g) =_{\text{def}} \mathbb{Z} \setminus I(g_1)$.

The set computed by C is $I(C) =_{\text{def}} I(g_C)$. If a gate $g \in G$ computes a singleton $\{a\}$, we will sometimes write $I(g) = a$ for simplicity. An *$\mathcal{O}$-formula* is an $\mathcal{O}$-circuit with maximal outdegree 1.

For $\mathcal{O} \subseteq \{\cup, \cap, \bar{}, +, \times\}$ we define *membership problems* for $\mathcal{O}$-circuits and $\mathcal{O}$-formulas over sets of integers by

$$\mathrm{MC}_{\mathbb{Z}}(\mathcal{O}) =_{\text{def}} \{(C, b) : C \text{ is an } \mathcal{O}\text{-circuit, } b \in \mathbb{Z} \text{ and } b \in I(C)\}$$

and

$$\mathrm{MF}_{\mathbb{Z}}(\mathcal{O}) =_{\text{def}} \{(C, b) : C \text{ is an } \mathcal{O}\text{-formula, } b \in \mathbb{Z} \text{ and } b \in I(C)\}.$$

For simplicity, we write $\mathrm{MC}_{\mathbb{Z}}(o_1, \ldots, o_r)$ instead of $\mathrm{MC}_{\mathbb{Z}}(\{o_1, \ldots, o_r\})$ and $\mathrm{MF}_{\mathbb{Z}}(o_1, \ldots, o_r)$ instead of $\mathrm{MF}_{\mathbb{Z}}(\{o_1, \ldots, o_r\})$.

We denote membership problems for circuits over sets of natural numbers (see [7] for details) by $\mathrm{MC}_{\mathbb{N}}(\mathcal{O})$ and $\mathrm{MF}_{\mathbb{N}}(\mathcal{O})$, respectively. These problems are defined analogously. The only differences are that we solely allow input gates with non-negative labels and that $\bar{}$-gates compute the complement with respect to $\mathbb{N}$. To

avoid confusion, we denote the set computed by a circuit C over sets of natural numbers by $I_{\mathbb{N}}(C)$.

We assume any appropriate circuit and formula encoding, where gates are sorted topologically and neighborhoods are readily available. All completeness results are in terms of many-one logspace reducibility.

Example 1. Defining the circuits $\mathrm{Z} =_{\mathrm{def}} \overline{\{0\}} \cup \{0\}$ and $\mathrm{ODD} =_{\mathrm{def}} (\{2\} \times \mathrm{Z}) + \{1\}$, we obtain $I(\mathrm{Z}) = \mathbb{Z}$ and $I(\mathrm{ODD}) = \{z \in \mathbb{Z} : z \text{ odd}\}$.

Example 2. Consider the circuit $\mathbb{Z}\text{-POWER2} =_{\mathrm{def}} \overline{(\mathrm{ODD} \cap \overline{-1 \cup 1}) \times \mathrm{Z}}$. It then holds that $I(\mathbb{Z}\text{-POWER2}) = \{2^k : k \geq 0\} \cup \{-2^k : k \geq 0\}$.

Example 3. The circuit $\mathbb{Z}\text{-PRIMES} =_{\mathrm{def}} C \cap \overline{C \times C}$, where C is a subcircuit defined as $\overline{-1 \cup 0 \cup 1}$, computes the set $\{p : p \text{ is prime}\} \cup \{-p : p \text{ is prime}\}$.

3 Digression: $\mathrm{MF}_{\mathbb{N}}(^-, +)$ is PSPACE-Complete

Before we start our analysis of membership problems over sets of integers, we draw our attention to circuits over sets of natural numbers. In this section we extend work by McKenzie and Wagner [7] by proving the problem $\mathrm{MC}_{\mathbb{N}}(^-, +)$ to be PSPACE-complete. As will be seen later, this result is of importance to us since the case $\{^-, +\}$ is one where the complexity of the $\mathbb{N}$- and $\mathbb{Z}$-membership problems diverges the most.

As an auxiliary tool, we define a new PSPACE-complete problem by introducing alternation into the well known NP-complete *Sum of Subset* problem: The problem *Quantified Sum of Subset* is defined as

$$\mathrm{QSOS} =_{\mathrm{def}} \Big\{(a_1, \ldots, a_n, b) : a_1, \ldots, a_n, b \in \mathbb{N}, n \equiv 1(2) \text{ and } \underbrace{\exists_{c_1 \in \{0,1\}} \forall_{c_2 \in \{0,1\}} \cdots \exists_{c_n \in \{0,1\}}}_{\text{strict alternation}} \Big(\sum_{i=1}^{n} c_i a_i = b \Big) \Big\}.$$

It is known that the problem of evaluating quantified boolean formulas (QBF) is PSPACE-complete [11]. QBF is defined as

$$\mathrm{QBF} =_{\mathrm{def}} \Big\{H : H \text{ is a boolean formula in 3-CNF with variables } x_1, \ldots, x_n \text{ such that } \underbrace{\exists x_1 \forall x_2 \cdots \exists x_n}_{\text{strict alternation}} \big(H(x_1, \ldots, x_n) = 1\big)\Big\}.$$

Observe that for both problems, the condition of strict alternation is not a real restriction. In the case of QSOS, any tuple with arbitrarily quantified numbers can be transformed into an equivalent tuple with strictly alternating quantifiers by simply inserting zeroes.

Lemma 1. *It holds that* $\mathrm{QBF} \leq_m^{\log} \mathrm{QSOS}$.

Proof. We define a logspace computable function f such that $H \in \mathrm{QBF} \Leftrightarrow f(H) \in \mathrm{QSOS}$. Let $n \in \mathbb{N}$ be odd and $H = \bigwedge_{i=1}^{m}(z_{i_1} \vee z_{i_2} \vee z_{i_3})$ with $z_{i_j} \in \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}$ be a boolean formula. We then define f by $f(H) =_{\mathrm{def}} (v_1, v_2, \ldots, v_n, 0, v'_1, 0, v'_2, 0, \ldots, v'_n, 0, c_1, 0, c_2, 0, \ldots, c_m, 0, d_1, 0, d_2, 0, \ldots, d_m, b)$, where $v_1, \ldots, v_n, v'_1, \ldots, v'_n$ and $c_1, \ldots c_m, d_1, \ldots, d_m$ and b are natural numbers which have the following decimal representations:

$v_i =_{\mathrm{def}} \underbrace{k_1 k_2 \ldots k_m}_{m} \underbrace{0 \ldots 0 \overset{(i)}{1} 0 \ldots 0}_{n}$, where

k_j is the number of occurrences of literal x_i in the j-th clause of H,

$v'_i =_{\mathrm{def}} \underbrace{k'_1 k'_2 \ldots k'_m}_{m} \underbrace{0 \ldots 0 \overset{(i)}{1} 0 \ldots 0}_{n}$, where

k'_j is the number of occurrences of literal $\neg x_i$ in the j-th clause of H,

$c_i =_{\mathrm{def}} \underbrace{0 \ldots 0 \overset{(i)}{1} 0 \ldots 0}_{m} \underbrace{0 \ldots 0 0 \ldots 0}_{n}$, (balancing vectors)

$d_i =_{\mathrm{def}} \underbrace{0 \ldots 0 \overset{(i)}{2} 0 \ldots 0}_{m} \underbrace{0 \ldots 0 0 \ldots 0}_{n}$, (balancing vectors)

$b =_{\mathrm{def}} \underbrace{4 \ldots 4 4 \ldots 4}_{m} \underbrace{1 \ldots 1 1 \ldots 1}_{n}$ (target vector).

Note that by defining f in this way, we achieve that $v_1, \ldots, v_n$ are quantified strictly alternating while vectors $v'_1, \ldots, v'_n$ and $c_1, \ldots c_m, d_1, \ldots, d_m$ are all quantified existentially.

In the following, let $I_{a_1,\ldots,a_n}$ be the interpretation which, for $1 \leq i \leq n$, assigns truth-value $a_i \in \{0, 1\}$ to variable x_i in H.

The following equivalences now hold:

$$
\begin{aligned}
H \in \mathrm{QBF} &\Leftrightarrow \exists a_1 \forall a_2 \cdots \exists a_n (I_{a_1,\ldots,a_n} \text{ satisfies } H) \\
&\Leftrightarrow \exists_{a_1 \in \{0,1\}} \forall_{a_2 \in \{0,1\}} \cdots \exists_{a_n \in \{0,1\}} (I_{a_1,\ldots,a_n} \text{ satisfies each clause of } H) \\
&\Leftrightarrow \exists_{e_1 \in \{0,1\}} \forall_{e_2 \in \{0,1\}} \cdots \exists_{e_n \in \{0,1\}} \\
&\quad \exists_{k_1,\ldots,k_m \in \{1,2,3\}} \left(\left(\left(\sum_{i=1}^{n} e_i v_i + (1 - e_i) v'_i \right) \right) = k_1 k_2 \ldots k_m \underbrace{1 \ldots 1}_{n} \right) \\
&\Leftrightarrow \exists_{e_1 \in \{0,1\}} \forall_{e_2 \in \{0,1\}} \cdots \exists_{e_n \in \{0,1\}} \exists_{l_1 \in \{0,1\}} \exists_{l_2 \in \{0,1\}} \cdots \exists_{l_n \in \{0,1\}} \\
&\quad \exists_{k_1,\ldots,k_m \in \{1,2,3\}} \left(\left(\left(\sum_{i=1}^{n} e_i v_i \right) + \left(\sum_{i=1}^{n} l_i v'_i \right) \right) = k_1 k_2 \ldots k_m \underbrace{1 \ldots 1}_{n} \right) \\
&\Leftrightarrow \exists_{e_1 \in \{0,1\}} \forall_{e_2 \in \{0,1\}} \cdots \exists_{e_n \in \{0,1\}} \exists_{l_1,\ldots,l_n \in \{0,1\}} \exists_{r_1,\ldots,r_m,r'_1,\ldots,r'_m \in \{0,1\}} \\
&\quad \left(\left(\left(\sum_{i=1}^{n} e_i v_i + l_i v'_i \right) + \left(\sum_{i=1}^{m} r_i c_i + r' d_i \right) \right) = \underbrace{44 \ldots 4}_{m} \underbrace{1 \ldots 1}_{n} \right) \\
&\Leftrightarrow f(H) \in \mathrm{QSOS}.
\end{aligned}
$$

Since f is computable in logarithmic space, our reduction is complete. □

Lemma 2. *It holds that* $\mathrm{QSOS} \leq_m^{\log} \mathrm{MF}_{\mathbb{N}}(^-, +)$.

Proof sketch. Let $(a_1, \ldots, a_n, b) \in \mathbb{N}^{n+1}$ for any odd n. Note that $(a_1, \ldots, a_n, b) \in \mathrm{QSOS} \Leftrightarrow (a_1, \ldots, a_n, 0, (\sum_{i=1}^{n} a_i) + 1, b + (\sum_{i=1}^{n} a_i) + 1) \in \mathrm{QSOS}$. Hence, without loss of generality, we may assume that $b \geq \sum_{i=1}^{n-1} a_i$. For $i = 1, \ldots, n$ we construct formulas A_i:

$$A_i =_{\mathrm{def}} \begin{cases} \overline{1 + \overline{(a_i - 1)}}, & \text{if } a_i \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Note that $I_{\mathbb{N}}(A_i) = \{0, a_i\}$. We now construct a $\{^-, +\}$-formula F as follows:

$$F_n =_{\mathrm{def}} A_n,$$
$$F_{k-1} =_{\mathrm{def}} A_{k-1} + \overline{F_k} \text{ for } 2 \leq k \leq n \text{ and}$$
$$F =_{\mathrm{def}} F_1.$$

An induction shows that

$$b \in I_{\mathbb{N}}(F) \Leftrightarrow \exists_{a_1' \in I_{\mathbb{N}}(A_1)} \forall_{a_2' \in I_{\mathbb{N}}(A_2)} \cdots \exists_{a_n' \in I_{\mathbb{N}}(A_n)} \left(\left(\sum_{i=1}^{n} a_i'\right) = b\right).$$

So we conclude $b \in I_{\mathbb{N}}(F) \Leftrightarrow \exists_{c_1 \in \{0,1\}} \forall_{c_2 \in \{0,1\}} \cdots \exists_{c_n \in \{0,1\}} ((\sum_{i=1}^{n} c_i a_i) = b)$ $\Leftrightarrow (a_1, \ldots, a_n, b) \in \mathrm{QSOS}$. F can be constructed in logarithmic space. Hence, $\mathrm{MF}_{\mathbb{N}}(^-, +)$ is PSPACE-hard. □

Theorem 1. *1.* $\mathrm{MF}_{\mathbb{N}}(^-, +)$ *is* PSPACE*-complete.*
2. $\mathrm{MC}_{\mathbb{N}}(^-, +)$ *is* PSPACE*-complete.*

Proof. This is an immediate consequence of Lemma 2 and $\mathrm{MC}_{\mathbb{N}}(\cup, \cap, ^-, +) \in$ PSPACE [7]. □

4 Differences from the Case of the Natural Numbers

We start with some simple observations.

Lemma 3. *1. Let* $\mathcal{O} \subseteq \{\cup, \cap, +, \times\}$. *Then* $\mathrm{MC}_{\mathbb{N}}(\mathcal{O}) \leq_m^{\log} \mathrm{MC}_{\mathbb{Z}}(\mathcal{O})$ *and* $\mathrm{MF}_{\mathbb{N}}(\mathcal{O}) \leq_m^{\log} \mathrm{MF}_{\mathbb{Z}}(\mathcal{O})$.
2. Let $\mathcal{O} \subseteq \{\cup, \cap, ^-\}$. *Then* $\mathrm{MC}_{\mathbb{N}}(\mathcal{O}) \equiv_m^{\log} \mathrm{MC}_{\mathbb{Z}}(\mathcal{O})$ *and* $\mathrm{MF}_{\mathbb{N}}(\mathcal{O}) \equiv_m^{\log} \mathrm{MF}_{\mathbb{Z}}(\mathcal{O})$.

Hence, we can omit $\mathrm{MC}_{\mathbb{Z}}(\mathcal{O})$ for $\mathcal{O} \subseteq \{\cup, \cap, ^-\}$ from our study, as in these cases the complexity coincides with the complexity of the corresponding $\mathrm{MC}_{\mathbb{N}}(\mathcal{O})$-problem. Furthermore, we can take lower bounds for membership problems for circuits without complementation from [7].

The *Graph Accessibility Problem* (GAP) for directed graphs, which is defined as $\{(G, s, t) : G$ is a directed graph and there is a path from s to t in $G\}$, is NL-complete [10].

Theorem 2. *The problem* $\mathrm{MC}_{\mathbb{Z}}(\times)$ *is* $\mathrm{NL} \wedge \oplus\mathrm{L}$*-complete.*

Proof. Containedness: Observe that $\mathrm{MC}_{\mathbb{Z}}(\times) = A \cap B$, where $A =_{\mathrm{def}} \{(C,b) : C$ is a $\{\times\}$-circuit, $b \in \mathbb{Z}$ and $\mathrm{abs}(b) = \mathrm{abs}(I(C))\}$ and $B =_{\mathrm{def}} \{(C,b) : C$ is a $\{\times\}$-circuit, $b \in \mathbb{Z}$ and $\mathrm{sgn}(b) = \mathrm{sgn}(I(C))\}$. Since A easily reduces to $\mathrm{MC}_{\mathbb{N}}(\times)$ which is in NL [7], A is in NL. It remains to show that B is in $\oplus\mathrm{L}$. We construct a nondeterministic logarithmic space machine M working on input (C,b) as follows:
For every path in C from the output gate to an input gate, M produces a computation path. M accepts precisely on those paths, where the input gate of the corresponding path in C is labelled with a negative integer. If $b \geq 0$, M produces an additional accepting path. It now holds that $\mathrm{acc}_M((C,b)) \equiv 1(2) \Leftrightarrow (C,b) \in B$. Hence, it follows that $B \in \oplus\mathrm{L}$ and we conclude $\mathrm{MC}_{\mathbb{Z}}(\times) \in \mathrm{NL} \wedge \oplus\mathrm{L}$.

Hardness: Let $A \in \mathrm{NL} \wedge \oplus\mathrm{L}$, thus there exist sets $B \in \mathrm{NL}$ and $C \in \oplus\mathrm{L}$ such that $A = B \cap C$. Let $x \in \Sigma^*$. GAP is NL-complete and so is $\overline{\mathrm{GAP}}$ due to [6] and [12]. Consequently, we can reduce in logarithmic space B to $\overline{\mathrm{GAP}}$ via f. Let $(G,s,t) = f(x)$. It is easy to construct in logarithmic space a $\{\times\}$-circuit C_1 such that $(G,s,t) \in \overline{\mathrm{GAP}} \Rightarrow 1 \in I(C_1)$ and $(G,s,t) \notin \overline{\mathrm{GAP}} \Rightarrow 0 \in I(C_1)$ (see [7], Theorem 8.5). We obtain $x \in B \Leftrightarrow (G,s,t) \in \overline{\mathrm{GAP}} \Leftrightarrow (C_1,1) \in \mathrm{MC}_{\mathbb{Z}}(\times)$.
Due to $C \in \oplus\mathrm{L}$, there exists a nondeterministic logspace machine M such that $x \in C \Leftrightarrow \mathrm{acc}_M(x) \equiv 1(2)$. Let C_x be the transition graph of M on input x. We transform C_x into a $\{\times\}$-circuit C_2, where accepting and rejecting nodes in C_x become inputs with labels -1 and 1 respectively, and the node representing the starting configuration of M becomes the output gate of C_2. Observe that C_x and C_2 can be constructed in logarithmic space. We obtain $x \in C \Rightarrow I(C_2) = -1$ and $x \notin C \Rightarrow I(C_2) = 1$.
So we can construct in logarithmic space the circuit $C_3 =_{\mathrm{def}} C_1 \times C_2$. It now holds that $x \in A \Leftrightarrow (x \in B \wedge x \in C) \Leftrightarrow (I(C_1) = 1 \wedge I(C_2) = -1) \Leftrightarrow (C_3,-1) \in \mathrm{MC}_{\mathbb{Z}}(\times)$, which yields the desired reduction. □

Remark 1. Under the assumption $\oplus\mathrm{L} \not\subseteq \mathrm{NL}$, $\mathrm{MC}_{\mathbb{Z}}(\times)$ is harder than $\mathrm{MC}_{\mathbb{N}}(\times)$. That is in a way surprising, as at first glance it does not seem more complex to multiply integers than natural numbers. We can in fact spot the difficulty evaluating $\{\times\}$-circuits over the integers: If we forbid -1 as a label for the input gates of a $\{\times\}$-circuit, the membership problem for these altered circuits is again NL-complete. This is due to the fact that -1 is the only negative number that can be multiplied by itself many times without its absolute value becoming too large.

Similarly, we obtain:

Theorem 3. *1. The problem* $\mathrm{MC}_{\mathbb{Z}}(\cap,\times)$ *is hard for* $\mathrm{C}_{=}\mathrm{L} \wedge \oplus\mathrm{L}$.
2. The problem $\mathrm{MC}_{\mathbb{Z}}(\cap,\times)$ *is in* P.

In contrast to the $\mathbb{N}$-case, we do not have a completeness result for $\mathrm{MC}_{\mathbb{Z}}(+,\times)$. The following lemma states that $\mathrm{MC}_{\mathbb{Z}}(+,\times)$ is more related to $\mathrm{MC}_{\mathbb{N}}(\cap,+,\times)$, for which no completeness result is known either [7].

Lemma 4. *It holds that* $\mathrm{MC}_{\mathbb{N}}(\cap,+,\times) \leq_m^{\log} \mathrm{MC}_{\mathbb{Z}}(+,\times)$.

4.1 NEXPTIME-Hard Membership Problems

It is known that $\mathrm{MC}_{\mathbb{N}}(\cup, +, \times)$ is PSPACE-complete [13], [14]. In this section we will show that the corresponding problem over the integers is complete for NEXPTIME. Intuitively, the difficulty when evaluating $\{\cup, +, \times\}$-circuits with integer labels is, that—unlike in the $\mathbb{N}$-case—we have to deal with very large (up to exponential in length) numbers: Our target number has polynomial length, but adding a very small and a very large integer can result in a number with a small absolute value again. In the $\mathbb{N}$-case, numbers can only become smaller when multiplied by 0, hence it suffices to compute numbers in the length of the target number when evaluating a circuit.

Lemma 5. *The problem* $\mathrm{MC}_{\mathbb{Z}}(\cup, +, \times)$ *is* NEXPTIME*-hard.*

Proof Sketch. The proof is by a generic reduction. We sketch here some of the main ideas. We consider a nondeterministic exponential time Turing machine M deciding a language L. For an input x we show how to construct in logarithmic space a $\{\cup, +, \times\}$-circuit C such that $x \in L \Leftrightarrow (C, 0) \in \mathrm{MC}_{\mathbb{Z}}(\cup, +, \times)$. Similar to Cook's famous proof of the NP-completeness of SAT, there exists a (now exponentially long) boolean formula F_x such that F_x is satisfiable if and only if there exists an accepting computation path of M on input x. Due to its exponential length, we can neither construct F_x in logarithmic space nor test its satisfiability directly.

Nevertheless, the key is to realize that we can assume F_x to be a formula which has a highly regular structure. Hence, one part of our proof is basically a reduction from *Succinct*-SAT to *Succinct*-SOS (see [2] for a survey on succinct problems). In fact, we can even assume that each variable in F_x only occurs in a constant number of clauses only depending on the machine M. Moreover, these occurrences follow a strict pattern. Quite similar to the proof of Lemma 1, we construct for each boolean variable φ_i in F_x vectors v_i, v_i'. These vectors describe occurrences of positive and negative literals of φ_i in clauses of F_x such that there is a subset of vectors whose sum has a certain value b if and only if F_x is satisfiable. Recall that F_x contains an exponential number of variables and clauses, thus we have to create exponentially many different vectors and each vector has exponential length. Due to the high-grade regularity of F_x, the vectors we have to construct have a very regular structure as well. For this reason, we can construct a polynomial size circuit which creates the vectors corresponding to the variables in F_x. Roughly speaking, we then construct a circuit which computes all sums of valid subsets, adds $-b$ to that set and ask whether 0 is an element of the resulting set.

Care is needed to ensure that only correct subsets—those corresponding to a satisfying assignment of truth-values to the variables in F_x—have the target subset sum b. In order to obtain the same subset sum for all satisfying assignments, we need to construct circuits computing appropriate balancing vectors. □

Theorem 4. *The problems* $\mathrm{MC}_{\mathbb{Z}}(\cup, +, \times)$ *and* $\mathrm{MC}_{\mathbb{Z}}(\cup, \cap, +, \times)$ *are complete for* NEXPTIME.

We do not have any decidable upper bound for the unrestricted versions of the membership problems. As the next example shows, there is evidence suggesting that $\mathrm{MF}_{\mathbb{Z}}(\cup, \cap, {}^{-}, +, \times)$—and thus also $\mathrm{MC}_{\mathbb{Z}}(\cup, \cap, {}^{-}, +, \times)$—might be undecidable:

Example 4. A prime p is called a *Fermat-Prime* if there exists $k > 0$ such that $p = 2^{2^k} + 1$. The only known Fermat-Primes are $2^{2^k} + 1$ for $k = 0, 1, 2, 3, 4$. It is currently known that $2^{2^k} + 1$ is composite for $5 \leq k \leq 32$ [3]. Moreover, it is conjectured that the number of Fermat-Primes is finite with $2^{2^4} + 1 = 65537$ being the largest Fermat-Prime (see, for example, [5]). There is a simple $\{\cup, \cap, {}^{-}, +, \times\}$-formula FERMAT having the property that $0 \in I(F)$ iff there is no Fermat-Prime greater than $2^{2^4} + 1$.[1] Hence, a decision procedure for $\mathrm{MF}_{\mathbb{Z}}(\cup, \cap, {}^{-}, +, \times)$ could test whether there exists a sixth Fermat-Prime. This would be surprising.

We construct FERMAT out of several subcircuits. Recall circuits Z, $\mathbb{Z}$-POWER2 and $\mathbb{Z}$-PRIMES from section 2. We now define the circuits $\mathbb{N}\text{-POWER8} =_{\mathrm{def}} \mathbb{Z}\text{-POWER2} \cap (\{\{7\} \times \mathrm{Z}) + \{1\})$ and $\mathbb{N}\text{-POWER2} =_{\mathrm{def}} \{1, 2, 4\} \times \mathbb{N}\text{-POWER8}$. Observe that $I(\mathbb{N}\text{-POWER8}) = \{8^k : k \geq 0\}$ and thus $I(\mathbb{N}\text{-POWER2}) = \{2^k : k \geq 0\}$. The circuit $\mathbb{Z}\text{-PRIMES}' =_{\mathrm{def}} \mathbb{Z}\text{-PRIMES} \cap \overline{3 \cup 5 \cup 17 \cup 257 \cup 65537}$ computes the set of positive and negative primes minus the set of known Fermat-Primes. All circuits constructed so far can easily be unfolded into formulas.

Finally we define FERMAT as $\overline{(\mathbb{Z}\text{-PRIMES}' \cap (\mathbb{N}\text{-POWER2} + \{1\})) \times \{0\}}$ and obtain: There is no Fermat-Prime greater than $2^{2^4}+1 \Leftrightarrow 0 \in I(\mathrm{FERMAT})$. The implication "$\Rightarrow$" follows directly from the construction. To see "$\Leftarrow$", note that the following is well known:

Lemma 6. *Let $l \geq 0$. If $2^l + 1$ is prime, then there exists $k \geq 0$ such that $l = 2^k$.*

4.2 Complementation as the Only Set Operation

In this section, we analyze membership problems for circuits with addition, whose only set operation is complementation. We will see that sets computed by such circuits have a very simple structure.

Lemma 7. *Let $C = (G, E, g_C, \alpha)$ be a $\{{}^{-}, +\}$-circuit. For every $g \in G$, there exists an $a_g \in \mathbb{Z}$ such that $I(g) \in \{\{a_g\}, \mathbb{Z} \setminus \{a_g\}, \mathbb{Z}, \emptyset\}$.*

Proof. Observe that $\{\{a\}, \mathbb{Z} \setminus \{a\}, \mathbb{Z}, \emptyset\}$ is closed under complementation. For $a, b, c, d \in \mathbb{Z}$ we have $\{a\} + \{b\} = \{a + b\}$, $(\mathbb{Z} \setminus \{a\}) + \{b\} = \mathbb{Z} \setminus \{a + b\}$, $(\mathbb{Z} \setminus \{a\}) + (\mathbb{Z} \setminus \{b\}) = \mathbb{Z} + (\mathbb{Z} \setminus \{c\}) = \mathbb{Z} + \{d\} = \mathbb{Z}$. Adding $\emptyset$ always results in $\emptyset$. □

We say that $I(g)$ is of type I, II, III or IV if $I(g) = \{a\}$, $\mathbb{Z} \setminus \{a\}$, $\mathbb{Z}$, $\emptyset$, respectively.

[1] In an earlier version, we used a weak variant of Goldbach's Conjecture as evidence. Holger Petersen (Stuttgart) observed that this can be improved to the statement about Fermat-Primes.

Theorem 5. *1. The problem* $\mathrm{MC}_{\mathbb{Z}}(^{-},+)$ *is* P*-complete.*
2. The problem $\mathrm{MF}_{\mathbb{Z}}(^{-},+)$ *is in* L.

Proof. 1. Containedness: Let $C = (G, E, g_C, \alpha)$ be a $\{^{-},+\}$-circuit and $b \in \mathbb{Z}$. Note that, by using the above calculation rule, we can determine in polynomial time what the type of $I(g_C)$ is. The following polynomial time algorithm works on input (C, b) and decides $\mathrm{MC}_{\mathbb{Z}}(^{-},+)$. If $I(g_C)$ is of type III or IV we accept or reject, respectively. Otherwise, we eliminate all $^{-}$-gates from C and obtain a $\{+\}$-circuit C'. Since $\mathrm{MC}_{\mathbb{Z}}(+) \in C_{=}\mathrm{L} \subseteq \mathrm{P}$ (Theorem 6), we can decide $(C', b) \in \mathrm{MC}_{\mathbb{Z}}(+)$ in polynomial time. If $I(g_C)$ is of type I, we accept iff $(C', b) \in \mathrm{MC}_{\mathbb{Z}}(+)$. If $I(g_C)$ is of type II, we accept iff $(C', b) \notin \mathrm{MC}_{\mathbb{Z}}(+)$.
Hardness: We show that the P-complete *monotone boolean circuit value problem* [4] can be reduced to $\mathrm{MC}_{\mathbb{Z}}(^{-},+)$. To do so, we transform a monotone boolean circuit C into a $\{^{-},+\}$-circuit C' of basically the same structure. Every input gate in C with boolean value 0 is replaced by a sub-circuit computing $\mathbb{Z}$, e.g $\overline{0} + \overline{0}$, every input gate in C with boolean value 1 is replaced by a sub-circuit computing $\emptyset$, e.g $\overline{\overline{0} + \overline{0}}$. Every $\vee$-gate in C is replaced by a $+$-gate and every $\wedge$-gate in C with predecessors g_1, g_2 is replaced by a sub-circuit $\overline{\overline{g_1} + \overline{g_2}}$. For every gate g in C' we now have either $I(g) = \mathbb{Z}$ or $I(g) = \emptyset$. Note that for every $+$-gate g in C' with predecessors g_1, g_2 it now holds that $I(g) = \emptyset \Leftrightarrow \big(I(g_1) = \emptyset \vee I(g_2) = \emptyset\big)$ and $I(g) = \mathbb{Z} \Leftrightarrow \big(I(g_1) = \mathbb{Z} \wedge I(g_2) = \mathbb{Z}\big)$. We obtain ($C$ evaluates to 0) $\Leftrightarrow I(C') = \mathbb{Z} \Leftrightarrow 0 \in I(C') \Leftrightarrow (C', 0) \in \mathrm{MC}_{\mathbb{Z}}(^{-},+)$.
2. For a $\{^{-},+\}$-formula F, determining the type of $I(F)$ can be done in logarithmic space, and $\mathrm{MC}_{\mathbb{Z}}(+)$ is in L, thus $\mathrm{MF}_{\mathbb{Z}}(^{-},+)$ is in L. □

Contrary to these results, we already know that $\{^{-},+\}$-circuits over sets of natural numbers can compute very complex sets (recall section 3).

5 Similarities Between $\mathbb{Z}$-Case and $\mathbb{N}$-Case

In this section, we present several results where the complexity in the two cases coincides.

Theorem 6. *The problem* $\mathrm{MC}_{\mathbb{Z}}(+)$ *is complete for* $\mathrm{C}_{=}\mathrm{L}$.

In contrast to $\{\times\}$-circuits, evaluating $\{+\}$-circuits is not harder in the $\mathbb{Z}$-case than it is in the $\mathbb{N}$-case.

Theorem 7. $\mathrm{MF}_{\mathbb{Z}}(\cup, \cap, ^{-}, +)$ *and* $\mathrm{MC}_{\mathbb{Z}}(\cup, \cap, ^{-}, +)$ *are* PSPACE*-complete.*

Proof Sketch: Containedness can be shown by a simple modification of the proofs for the corresponding problems in the $\mathbb{N}$-case (cf. [7]). For PSPACE-hardness, observe that the PSPACE-complete problem QSOS can be reduced to $\mathrm{MF}_{\mathbb{Z}}(\cup, \cap, ^{-}, +)$ by adapting the proof of Lemma 2: For $i = 1, \ldots, n$, define formulas A_i as $0 \cup a_i$. The rest of the proof is analog. □

Theorem 8. $\mathrm{MF}_{\mathbb{Z}}(\cup, \cap, ^{-}, \times)$ *and* $\mathrm{MC}_{\mathbb{Z}}(\cup, \cap, ^{-}, \times)$ *are* PSPACE*-complete.*

Proof Sketch: With respect to containedness, the same as above holds. We here sketch a reduction of the PSPACE-complete problem $\mathrm{MF}_{\mathbb{N}}(\cup, \cap, \bar{}, \times)$ [7] to $\mathrm{MF}_{\mathbb{Z}}(\cup, \cap, \bar{}, \times)$. Let F be a $\{\cup, \cap, \bar{}, \times\}$-formula over sets of natural numbers. A $\{\cup, \cap, \bar{}, \times\}$-formula F' is obtained by replacing every input gate g of F with a formula $(\alpha(g) \cup (\alpha(g) \times \{-1\}))$. Arguing by induction on the number of gates in F, we obtain $I(F') = I_{\mathbb{N}}(F) \cup (I_{\mathbb{N}}(F) \times \{-1\})$. Therefore, it now holds for all $b \in \mathbb{N}$ that $b \in I_{\mathbb{N}}(F) \Leftrightarrow b \in I(F')$. This yields the desired reduction. □

For the complexity of the remaining membership problems in our analysis, we also have the same upper and lower bounds as McKenzie and Wagner have for the corresponding problems in the $\mathbb{N}$-case. The proofs for the $\mathbb{Z}$-case bounds only require straightforward modifications applied to those in the $\mathbb{N}$-case (see [7]). The results for these membership problems are also included in the table, but lack a reference to a theorem or lemma in this paper.

6 Conclusion and Open Problems

Table 1 summaries our results, and several open problems are apparent from it. If one could prove $\mathrm{MC}_{\mathbb{Z}}(+, \times) \in \mathrm{P}$, then it would follow that the problems $\mathrm{MC}_{\mathbb{Z}}(+, \times)$, $\mathrm{MC}_{\mathbb{Z}}(\cap, +, \times)$, and $\mathrm{MC}_{\mathbb{N}}(\cap, +, \times)$ are all P-complete (cf. [7]).

Table 1. Membership problems for circuits over subsets of $\mathbb{Z}$

$\mathcal{O}$	$\mathrm{MC}_{\mathbb{Z}}(\mathcal{O})$ lower bound	$\mathrm{MC}_{\mathbb{Z}}(\mathcal{O})$ upper bound	Th. L.	$\mathrm{MF}_{\mathbb{Z}}(\mathcal{O})$ lower bound	$\mathrm{MF}_{\mathbb{Z}}(\mathcal{O})$ upper bound	Th. L.
$\cup, \cap, \bar{}, +, \times$	NEXPTIME	?	T4	PSPACE	?	
$\cup, \cap, +, \times$	NEXPTIME	NEXPTIME	T4	NP	NP	
$\cup, +, \times$	NEXPTIME	NEXPTIME	T4	NP	NP	
$\cap, +, \times$	P	coNP		L	DLOGCFL	
$+, \times$	P	coNP	L4	L	DLOGCFL	
$\cup, \cap, \bar{}, +$	PSPACE	PSPACE	T7	PSPACE	PSPACE	
$\cup, \cap, +$	PSPACE	PSPACE		NP	NP	
$\cup, +$	NP	NP		NP	NP	
$\bar{}, +$	P	P	T5	L	L	T5
$\cap, +$	$\mathrm{C}_{=}\mathrm{L}$	$\mathrm{C}_{=}\mathrm{L}$		L	L	
$+$	$\mathrm{C}_{=}\mathrm{L}$	$\mathrm{C}_{=}\mathrm{L}$	T6	L	L	
$\cup, \cap, \bar{}, \times$	PSPACE	PSPACE	T8	PSPACE	PSPACE	
$\cup, \cap, \times$	PSPACE	PSPACE		NP	NP	
$\cup, \times$	NP	NP		NP	NP	
$\cap, \times$	$\mathrm{C}_{=}\mathrm{L} \wedge \oplus\mathrm{L}$	P	T3	L	L	
$\times$	$\mathrm{NL} \wedge \oplus\mathrm{L}$	$\mathrm{NL} \wedge \oplus\mathrm{L}$	T2	L	L	
$\cup, \cap, \bar{}$	P	P	L3	L	L	L3
$\cup, \cap$	P	P	L3	L	L	L3
$\cup$	NL	NL	L3	L	L	L3
$\cap$	NL	NL	L3	L	L	L3

Another open problem is finding a decidable upper bound for the complexity of $\mathrm{MC}_{\mathbb{Z}}(\cup, \cap, {}^{-}, +, \times)$, or proving it undecidable. As the corresponding problem in the $\mathbb{N}$-case shares that fate, we are interested in how the complexities of $\mathrm{MC}_{\mathbb{Z}}(\cup, \cap, {}^{-}, +, \times)$ and $\mathrm{MC}_{\mathbb{N}}(\cup, \cap, {}^{-}, +, \times)$ compare.
Related to this, there is a further open question: Is there a circuit over sets of integers computing precisely the set $\mathbb{N}$?
If such a circuit exists, $\mathrm{MC}_{\mathbb{N}}(\cup, \cap, {}^{-}, +, \times) \leq_m^{\log} \mathrm{MC}_{\mathbb{Z}}(\cup, \cap, {}^{-}, +, \times)$ would immediately follow. Holger Petersen (private communication) was the first to observe that there exists a circuit C having the property that $I(C) \cap \mathbb{N}$ is infinite while $I(C) \cap -\mathbb{N}$ is finite (see section 4.1).

Acknowledgments. The author is grateful to Klaus W. Wagner, Christian Glaßer, Daniel Meister, Bernhard Schwarz (Würzburg) and Holger Petersen (Stuttgart) for very useful discussions and important hints.

References

1. E. Allender, Making computation count: Arithmetic Circuits in the Nineties, in the Complexity Theory Column, *SIGACT NEWS 28 (4),* pp. 2–15, 1997.
2. J. L. Balcázar, A. Lozano, and J. Torán, The complexity of algorithmic problems in succinct instances, *Computer Science,* Plenum, New York, 1992.
3. R. E. Crandall, E. W. Mayer, and J. S. Papadopoulos, The Twenty-Fourth Fermat Number is Composite, *Math. Comput. 72,* 1555-1572, 2003.
4. L. M. Goldschlager, The monotone and planar circuit value problems are logspace complete for P, *SIGACT NEWS 9,* pp. 25–29, 1977.
5. G. H. Hardy and E. M. Wright, An introduction to the theory of numbers, *Oxford University Press,* 1979.
6. N. Immerman, Nondeterministic space is closed under complementation, *SIAM Journal on Computing 17,* pp. 935–938, 1988.
7. P. McKenzie and K. W. Wagner. The Complexity of Membership Problems for Circuits over Sets of Natural Numbers, *Lecture Notes in Computer Science 2607,* pp. 571–582, 2003.
8. C. H. Papadimitriou. Computational Complexity, *Addison-Wesley,* 1994.
9. H. Petersen, Bemerkungen zu ganzzahligen Ausdrücken, *Private Communication,* 2004.
10. W. J. Savitch, Maze recognizing automata and nondeterministic tape complexity, *Journal of Computer and System Sciences 7,* pp. 389–403, 1973.
11. L. J. Stockmeyer, A. R. Meyer, Word Problems Requiring Exponential Time, *Proceedings of the 5th ACM Symposium on the Theory of Computing,* pp. 1–9, 1973.
12. R. Szelepcsényi, The method of forced enumeration for nondeterministic automata, *AI 26,* pp. 279–284, 1984.
13. K. W. Wagner, The complexity of problems concerning graphs with regularities, *Proceedings 11th Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 176,* pp. 544–552, 1984.
14. K. Yang, Integer circuit evaluation is PSPACE-complete, *Proceedings 15th Conference on Computational Complexity,* pp. 204–211, 2000.

Some Meet-in-the-Middle Circuit Lower Bounds

Kristoffer Arnsfelt Hansen and Peter Bro Miltersen

Department of Computer Science, University of Aarhus, Denmark
{arnsfelt,bromille}@daimi.au.dk

Abstract. We observe that a combination of known top-down and bottom-up lower bound techniques of circuit complexity may yield new circuit lower bounds.
An important example is this: Razborov and Wigderson showed that a certain function f in $\mathbf{ACC}^0$ cannot be computed by polynomial size circuits consisting of two layers of MAJORITY gates at the top and a layer of AND gates at the bottom. We observe that a simple combination of their result with the Håstad switching lemma yields the following seemingly much stronger result: The same function f cannot be computed by polynomial size circuits consisting of two layers of MAJORITY gates at the top and an arbitrary $\mathbf{AC}^0$ circuit feeding the MAJORITY gates.

1 Introduction

During the 1980's and 1990's significant progress was made understanding the power of *bounded depth* computation, i.e. computation performed by unbounded fan-in circuits of constant depth and polynomial size containing layers of gates of various kinds. Such research was motivated by its fundamental nature. Indeed, it can be argued that understanding such restricted computation must be a prerequisite to solving problems such as **P** vs. **NP**.

Two methodologies for proving lower bounds on the power of constant depth computation were developed in the 1980's:

1. *top-down* methods, exemplified by lower bounds obtained using two- or multi-party communication complexity [1,2] and
2. *bottom-up* methods, exemplified by lower bounds obtained using the Håstad switching lemma [3,4,5,6,7,8,9].

Roughly speaking, a top-down argument proves that a problem cannot be solved by a certain kind of circuit by assuming to the contrary that it can, looking at the output gate of the circuit and working its way downwards to an input wire where a contradiction is reached. In contrast, a bottom-up argument starts at the input and successively eliminates gates at the bottom of the circuit, until the circuit is gone and a contradiction is reached.

Progress on the understanding of bounded depth computation in the last decade has been less encouraging than the rate of progress in the 1980s. One reason for this is indicated by the Razborov-Rudich *natural proof* concept [10] that shows, under certain cryptographic assumptions, that the kind of arguments

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 334–345, 2004.
© Springer-Verlag Berlin Heidelberg 2004

we are currently using will not be strong enough to, say, separate $\mathbf{TC}^0$ from classes above it.

Still, there seems to be plenty of possible results not ruled out by the natural-proof obstacle. In this paper, we show a surprisingly simple way of *combining* top-down and bottom-up arguments yielding some new circuit lower bounds.

The general principle of the technique we describe is as follows: Suppose we start out with a theorem (typically obtained using top-down methods) that a certain boolean function f cannot be computed by a small circuit of type $\mathcal{C} \circ \mathrm{AND}$, i.e., a layered Boolean circuits of which the bottom layer consists of AND-gates feeding into a circuit of some class $\mathcal{C}$. Furthermore assume that the bottom layer of each circuit in $\mathcal{C}$ consists of gates computing *symmetric* functions (say, MAJORITY). Further, assume that we have a function f' and that hitting f' with random restrictions of certain parameters results in a function of which f is a restriction (for instance, f' and f could both be the parity function). Then, using the Beame/Håstad switching lemma, we may obtain a new result stating that f' cannot be computed by $\mathcal{C} \circ \mathbf{AC}^0$ circuits, i.e., circuits consisting a multi-output $\mathbf{AC}^0$ circuit feeding its results into a $\mathcal{C}$-circuit. Thus, the switching lemma allows us to replace the bottom AND gate in the original lower bound with an entire $\mathbf{AC}^0$ circuit. As the technique combines a top-down argument with a bottom-up argument (i.e., random restrictions and the switching lemma) we call the resulting bounds *meet-in-the-middle lower bounds.* This kind of argument has been used sporadically before: Zhang, Barrington and Tarui [11] characterized the symmetric functions computable by quasi-polynomial size perceptrons. Goldmann [5] proved that computing the MOD_2-function requires $\mathbf{Th} \circ \mathbf{AC}^0$ circuits of size $2^{n^{\Omega(1)}}$. Furthermore, Berg and Ulfberg [4] and also Krause and Pudlák [6] gave $\mathbf{\Sigma}_k$ functions which require $\mathbf{Th} \circ \mathbf{AC}^0_{k-2}$ circuits of size $2^{n^{\Omega(1)}}$. These proofs all used ideas along the lines of what we describe above. However, the approach does not seem to have been investigated systematically before. In this paper we use it to prove some, to the best of our knowledge, previously unstated lower bounds.

We find that the most interesting application of the technique in this paper is the following result: $\mathbf{ACC}^0$ cannot be computed by $\mathbf{MAJ} \circ \mathbf{MAJ} \circ \mathbf{AC}^0$ circuits, i.e., circuits consisting of two layers of majority gates above an $\mathbf{AC}^0$ circuit (for a full explanation of our notation for circuit classes, see Sect. 2.1). The previous best result along these lines and the result we base our proof upon is the result by Razborov and Wigderson [9], showing that $\mathbf{ACC}^0$ cannot be simulated by $\mathbf{MAJ} \circ \mathbf{MAJ} \circ \mathbf{AND}$ circuits, i.e., circuits consisting of two layers of majority gates above a layer of $\mathbf{AND}$ gates. As $\mathbf{MAJ}$ is symmetric, the meet-in-the-middle technique applies. More precisely, we prove the following theorem.

Theorem 1. *Any* $\mathbf{MAJ} \circ \mathbf{SYM} \circ \mathbf{AC}^0$ *circuit computing* $(\mathrm{MOD}_2)_n \circ \mathrm{AND}_{\log n} \circ (\mathrm{MOD}_2)_n$ *has size* $n^{\Omega(\log n)}$.

Tsai [12] proved strong lower bounds on the degree of the MOD_l function viewed as a polynomial over $\mathbf{Z}_m$, with l having a prime factor not dividing m. As a polynomial over $\mathbf{Z}_m$ may be viewed as a $\mathbf{MOD}_m \circ \mathbf{AND}$ circuit, and MOD_m

is a symmetric function, the meet-in-the-middle technique applies. Using this, as well as a well known lower bound for the MAJORITY function, we can prove the following theorems.

Theorem 2. *Let m and l be positive integers. There is a constant c such that, if l has a prime factor not dividing m, then depth $d+1$ $\mathbf{MOD}_m \circ \mathbf{AC}^0_d$ circuits require more than size $2^{\frac{1}{14}(cn)^{\frac{1}{d}}}$ to compute either of MOD_l and $\neg\mathrm{MOD}_l$, for sufficiently large n, and if m is not a prime power, this size is also required to compute $\neg\mathrm{MOD}_m$.*

Theorem 3. *Let m be any integer. Then for any integer d, any $\epsilon > 0$ and $c < \frac{1}{2}$, depth $d+1$ $\mathbf{MOD}_m \circ \mathbf{AC}^0_d$ circuits require more than size $2^{\frac{1-\epsilon}{14}(cn)^{\frac{1}{d}}}$ to compute* MAJORITY *for sufficiently large n.*

Finally, we consider a slightly modified version of the Sipser functions f^m_k used by Håstad [8], computed by $\mathbf{\Pi}_k$ circuits of polynomial size but shown there to require exponential size $\mathbf{AC}^0_{k-1}$ circuits. We define the function f^m_k on $m^{k-1}\sqrt{\frac{k}{2}m\log m}$ variables by a $\mathbf{\Pi}_k$ formula, with all different unnegated variables at the leaves. The fanin on all levels is m except at the bottom level where the fanin is $\sqrt{\frac{k}{2}m\log m}$. We prove the following theorem.

Theorem 4. *For $k > 1$ and sufficiently large m, there are no depth $k+1$ $\mathbf{MOD}_m \circ \mathbf{AC}^0_k$ circuits computing f^m_k with bottom fanin $\frac{1}{12\sqrt{2k}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}$ and size $2^{\frac{1}{12\sqrt{2k}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}}$.*

Using a weaker notion of representation of boolean functions by polynomials and using a similar proof technique as well as extending an idea of Håstad ([8] Theorem 9.1), we can obtain the following lower bounds for $\mathbf{AC}^0$ circuits with a single MOD_m gate (allowed to appear anywhere in the circuit), thus generalizing Theorem 2.

Theorem 5. *Let m and l be positive integers. There is a constant c such that, if l has a prime factor not dividing m, then depth d $\mathbf{AC}^0_d$ circuits containing a single MOD_m gate, require more than size $2^{\frac{1}{28}(cn)^{\frac{1}{d}}}$ to compute either of MOD_l and $\neg\mathrm{MOD}_l$ for sufficiently large n.*

This is somewhat similar to a result by Aspnes et al [13] (later generalized by Beigel [14] and Barrington and Straubing [15]) concerning $\mathbf{AC}^0$ circuits with a single MAJORITY gate.

This paper is organized as follows: In Sect. 2 we make precise the notation we use for circuit classes and state the previous results we use as bottom-up and top-down ingredients in order to derive the theorems above. In Sect. 3, we derive our five main results stated above. Using the approach of Furst, Saxe and Sipser [7] one can use super-quasi-polynomial lower bounds for constant depth circuits to derive oracle separations of corresponding subclasses of **PSPACE.** In Sect. 4, we describe the oracle consequences of the three first exponential lower bounds above.

2 Preliminaries

2.1 Circuit Classes

The circuits we consider will be built from various families of unbounded fanin gates. The inputs are allowed to be both unnegated and negated variables as well as the constants 0 and 1. Let $x_1, \ldots, x_n$ be boolean inputs. For a positive integer m, the MOD_m function outputs 1 if and only if $\sum_{i=1}^{n} x_i \not\equiv 0 \pmod{m}$. The MAJORITY function outputs 1 if and only if $\sum_{i=1}^{n} x_i \geq \frac{n}{2}$. Given integer weights $w = (w_1, \ldots, w_n)$ and an integer threshold t, a threshold gate with weights w and threshold t outputs 1 if and only if $\sum_{i=1}^{n} w_i x_i \geq t$.

Let **AND** and **OR** denote the families of unbounded fanin AND and OR gates. Let $\mathbf{MOD}_m$, **MAJ** and **Th** denote the families of MOD_m, MAJORITY and threshold gates. Finally let **SYM** be the family of all symmetric gates.

If G is a family of boolean gates and $\mathcal{C}$ is a family of circuits we let $G \circ \mathcal{C}$ denote the class of circuits consisting of a G gate taking circuits from $\mathcal{C}$ as inputs. If we need to specify a specific bound on the fanin of some of the gates, this will be specified by a subscript.

$\mathbf{\Pi}_k$ is the class of functions computed by depth k circuits of alternating AND and OR gates with an AND gate as output. $\mathbf{\Sigma}_k$ is the analogous class with an OR gate as output instead. Let $\mathbf{AC}^0_k$ be the union of $\mathbf{\Pi}_k$ and $\mathbf{\Sigma}_k$ and $\mathbf{AC}^0$ be the union of $\mathbf{AC}^0_k$ for all k.

Theorem 5 consider $\mathbf{AC}^0$ circuits containing just a single MOD_m gate. Under this special restriction, our assumption that NOT gates occur only as negations on the input variables, cannot in general be assumed without loss of generality: the standard way of moving the negations to the bottom of the circuits may increase the number of MOD_m gates. The theorem can, however, be proved with NOT gates occurring anywhere in the circuits and we do *not* have to charge the NOT gates when defining the depth of the circuits.

Usually we would by our notation above implicitly consider the circuit classes defined to be of polynomial size. It will be convenient however, to use the same notation, but giving the size explicitly in the statement of our lower bounds.

2.2 Bottom-Up Ingredients

A *restriction* on a set V of boolean variables is a map $\rho : V \to \{0, 1, \star\}$. It acts on a boolean function $f : V \to \{0, 1\}$, creating a new boolean function f_ρ on the set of variables for which $\rho(x) = \star$, obtained by substituting $\rho(x)$ for $x \in V$ whenever $\rho(x) \neq \star$. The variables x for which $\rho(x) = \star$ are called *free;* the other variables *set.*

Let R^l_n denote the set of all restriction ρ leaving l variables free. We will use the following version of Håstads Switching Lemma due to Beame [3].

Lemma 1. *Let f be a DNF formula in n variables with terms of length at most r. Let $l = pn$ and pick ρ uniformly at random from R^l_n. Then*

$$\Pr[f_\rho \textit{ does not have a decision tree of depth } d] < (7pr)^d \ .$$

The main feature of Beame's switching lemma crucial to us is that it gives us a decision tree. A decision tree may be viewed as a DNF *where all terms are mutually contradictory.* This means that instead of viewing it as an OR of AND's, we can also view it as a sum of products, over any ring, representing false as 0 and true as 1. As this property is explicit in Beame's lemma, we shall us it whenever we can.

In order to prove Theorem 4, we will need a special kind of restrictions, however. The corresponding switching lemma is not stated in terms of decision trees, yet we will use it similarly obtaining the crucial property in a different way.

Let q be a real number and $(B_i)_{i=1}^r$ a partition of the variables. Let $R_{q,B}^+$ be the probability distribution on restrictions ρ given as follows. For every B_i, we let $s_i = \star$ independently with probability q and $s_i = 0$ otherwise. Then for every $x_k \in B_i$, we let $\rho(x_k) = s_i$ independently with probability q and $\rho(x_k) = 1$ otherwise.

Additionally we define, for $\rho \in R_{q,B}^+$ the restriction $g(\rho)$ as follows: For every B_i with $s_i = \star$, $g(\rho)$ assigns the value 1 to every variable except the variable with the highest index, which is assigned $\star$.

Similarly we define the distribution $R_{q,B}^-$, and for $\rho \in R_{q,B}^-$ also $g(\rho)$ by interchanging 0 and 1 in the definitions.

Lemma 2 ([8], Lemma 6.3). *Let f be a DNF formula with terms of length at most r. Pick ρ at random from $R_{q,B}^+$. Then*

$$\Pr[f_{\rho g(\rho)} \text{ can not be written as a CNF with clauses of size at most } s] < (6qr)^s \ .$$

The same statement holds for converting a CNF into a DNF, and both of these also holds for $\rho \in R_{q,B}^-$ instead.

Berg and Ulfberg showed, that by closely inspecting the proof of Lemma 2, one actually already obtains a DNF with small mutually contradictory terms, when applying a restriction to a CNF with small clauses [4].

Let in the following $q = \sqrt{\frac{2k \log m}{m}}$ and let B_i be the partition of the variables given by the inputs to the gates at the bottom of f_k^m. We will then have that applying a restriction from $R_{q,B}^+$ does not destroy f_k^m too much.

Lemma 3 ([8], Lemma 6.8). *For odd k and for random $\rho \in R_{q,B}^+$, then the circuit defining $(f_k^m)_{\rho g(\rho)}$ will contain the circuit defining f_{k-1}^m with probability at least $\frac{2}{3}$, when $\frac{m}{\log m} \geq 100k$ for $m > m_1$ where m_1 is some absolute constant.*

The same statement holds for even k, using $\rho \in R_{q,B}^-$ instead.

2.3 Top-Down Ingredients

Håstad and Goldrnann gave exponenential lower bounds for $\mathbf{TC}_3^0$ circuits with small bottom fanin [2].

Theorem 6. *Any depth 3 circuit with a* MAJORITY *gate at the top,* **SYM** *gates below and any kind of gates with fanin at most* $s-1$ *at the bottom which computes* $(\mathrm{MOD}_2)_n \circ \mathrm{AND}_s$ *must be of size* $2^{\Omega(\frac{n}{s4^s})}$.

Corollary 1. *Any depth 3 circuit with a* MAJORITY *gate at the top,* **SYM** *gates below and any kind of gates with fanin at most* $\frac{1}{3}\log n$ *at the bottom which computes* $(\mathrm{MOD}_2)_n \circ \mathrm{AND}_{\log n}$ *must be of size* $2^{n^{\Omega(1)}}$.

The results of Håstad and Goldmann were used by Razborov and Wigderson [9] to show the following lower bound.

Theorem 7. *Any* $\mathbf{MAJ} \circ \mathbf{SYM} \circ \mathbf{AND}$ *circuit computing* $(\mathrm{MOD}_2)_n \circ \mathrm{AND}_{\log n} \circ (\mathrm{MOD}_2)_n$ *must be of size* $n^{\Omega(\log n)}$.

Their proof involved applying a random restriction to the circuit, reducing the fanin of the AND gates to $\frac{1}{3}\log n$, while still having a circuit computing $(\mathrm{MOD}_2)_n \circ \mathrm{AND}_{\log n}$ as a subfunction.

The MOD_m-degree of a boolean function f on n variables is defined as the minimum degree of a polynomial p over $\mathbf{Z}_m$ such that $f(x) = 0 \Leftrightarrow p(x) = 0$ for all $x \in \{0,1\}^n$. Barrington, Beigel and Rudich gave $n^{\Omega(1)}$ lower bounds on the MOD_m-degree of the MOD_l and $\neg\mathrm{MOD}_l$ function when l has a prime factor not dividing m and also on the $\neg\mathrm{MOD}_m$ function when m is not a prime power [16]. These lower bounds was improved to $\Omega(n)$ by Tsai [12].

Green studied a weaker notion of MOD_m-degree [17], and generalized the lower bound of Tsai. The *weak* MOD_m-degree of a boolean function f (which is not the zero function) is the minimum degree of a polynomial p over $\mathbf{Z}_m$, which is not the zero polynomial, such that $p(x) \not\equiv 0 \pmod m \Rightarrow f(x) = 1$ for all $x \in \{0,1\}^n$.

Using this definition the the following lower bounds was obtained.

Theorem 8. *Let* m *and* l *be positive integers. If* l *has a prime factor not dividing* m *then the weak* MOD_m*-degree of the* MOD_l *and* $\neg\mathrm{MOD}_l$ *functions on* n *variables is* $\Omega(n)$. *If* m *is not a prime power the same holds for* $\neg\mathrm{MOD}_m$.

We will also need the following well known (weak) MOD_m-degree lower bounds for the MAJORITY function and the AND function ([8] Lemma 9.8, [18] Proposition 1).

Lemma 4. *Let* m *be any positive integer. The weak* MOD_m*-degree of the* MAJORITY *and* AND *functions on* n *variables are* $\geq \frac{n}{2}$ *and* n, *respectively.*

3 Lower Bound Proofs

In all proofs, we shall without loss of generality assume that the various real quantities appearing are integers.

3.1 Proof of Theorem 1

Let d be any constant and let C be a $\mathbf{MAJ} \circ \mathbf{SYM} \circ \mathbf{\Pi}_d$ circuit computing $f_n = (\mathrm{MOD}_2)_n \circ \mathrm{AND}_{\log n} \circ (\mathrm{MOD}_2)_n$, i.e., the circuit has $m = n^2 \log n$ Boolean inputs. Assume, to the contrary that the size S of C is $n^{o(\log n)}$. Also assume that n is sufficiently large.

We shall hit the circuit C and the function f_n with a random restriction ρ taken from $R_m^{m^{0.99}}$ and argue that

1. With overwhelming probability, $(f_n)_\rho$ contains as a subfunction (i.e., can be further restricted so as to obtain) the function $f_{\sqrt{n}}$.
2. With probability at least $\frac{1}{2}$, the function computed by $(C_n)_\rho$ can also be computed by a $\mathbf{MAJ} \circ \mathbf{SYM} \circ \mathbf{AND}$ circuit of size $n^{o(\log n)}$.

The desired lower bound then follows immediately from Theorem 7.

- Proof of item 1: The variables of f_n are divided into $n \log n$ group of n variables, each group corresponding to a (bottom) MOD_2 operation in the definition of f_n. If the random restriction ρ leaves $\sqrt{n}$ variables in each and all of these groups free, we have the desired subfunction, so we just have to show that this is the case with overwhelming probability. For a fixed set A of $n \geq \frac{\sqrt{m}}{\log m}$ variables, the probability that a random subset B of size $m^{0.99}$ of the $m = n^2 \log n$, variables (B being the free variables of the random restriction), has $|A \cap B| < n^{1/2}$ is negligible. Thus, the probability that ρ leaves less than $n^{\frac{1}{2}}$ variables free in any of the $n \log n$ groups is also negligible, as desired.
- Proof of item 2: We may view ρ as a composition of several random restrictions ρ_i, i=1, ..., d, with ρ_i taken at random from $R_m^{l_i}$ with $l_i = \frac{n}{(28 \log S)^d}$ and one last random restriction making the final number of free variables the correct number.
  Look at the Π_d circuit at the bottom of C. Assume the gates at level $i+1$ from the bottom are OR gates and that after having applied $\rho_1, \ldots, \rho_i$ that gates at level $i+1$ represent decision trees of depth at most $\log S$ and hence DNFs of term length at most $\log S$. By Lemma 1, the probability that the function computed at a gate in level $i+1$ can not be written as a decision tree of depth at most $\log S$ is bounded by $\left(\frac{7}{28}\right)^{-\log S} = 2^{-2 \log S}$. A similar argument holds for the levels of AND gates. (Also, we view the AND or OR gates at the bottom level, as taking OR or AND gates of fanin 1 as input). Thus, the probability that ρ does not turn *all* the gates of the Π_d circuit into decision trees of depth at most $\log S$ is at most $2^{-2 \log S} S \leq \frac{1}{2}$. Thus, with probability at least $\frac{1}{2}$, C_ρ can be computed as the majority of a number of symmetric gates, each taking as input a function computed by a decision tree of depth $\log S$. Decision trees are also DNFs *with the additional property that any input assignment makes at most one term true.* Thus, a symmetric gate G that takes as input a number of such trees $T_1, T_2, \ldots, T_k$ can be replaced with another symmetric gate G' that takes as inputs all terms (i.e.,

conjunctions of variables) of all the DNFs corresponding to all the trees T_j. Thus, with probability at least $\frac{1}{2}$ we have that C_ρ can be computed by a $\mathbf{MAJ} \circ \mathbf{SYM} \circ \mathbf{AND}$ circuit of size polynomial in S and $2^{\log S}$ (the latter quantity being the size of a decision tree of depth $\log S$), i.e., polynomial in s. Since $S = n^{o(\log n)}$, the size of the new circuit is also $n^{o(\log n)}$ and we are done. □

3.2 Proof of Theorem 2

First observe that after applying a restriction to MOD_l or $\neg\mathrm{MOD}_l$, by further setting at most $l-1$ extra variables, it results in the same function on fewer variables. Thus we can, using Theorem 8 choose c such that none of the functions on n variables that can appear after applying a restriction, has MOD_m-degree less than cn.

Now assume that C is a $\mathrm{MOD}_m \circ \mathbf{AC}^0_d$ circuit of depth $d+1$ and size $S = 2^{\frac{1}{14}(cn)^{\frac{1}{d}}}$ that computes a function with that property.

For $0 \leq i \leq d-1$ let $n_{i+1} = \frac{n}{14}(14\log_2 S)^{-i} = \frac{n}{14}\left((cn)^{\frac{1}{d}}\right)^{-i}$. Pick a restriction ρ at random from $R_n^{n_d}$ by picking it as a composition of randomly chosen restrictions $\rho_1, \ldots, \rho_d$, where $\rho_1 \in R_n^{n_1}$ and $\rho_{i+1} \in R_{n_i}^{n_{i+1}}$ for $0 \leq i \leq d-1$.

Viewing a gate at level 1 as a DNF with terms of length 1 or a CNF with clauses of size 1, the probability that the function computed does not have a decision tree of height at most $\log_2 S$ after applying ρ_1, is less than $(7\frac{n_1}{n})^{\log_2 S} = \frac{1}{S}$ by Lemma 1.

Now assuming that after applying the restrictions $\rho_1, \ldots, \rho_i$, every function computed at a gate at level i has a decision tree of height at most $\log_2 S$, consider a gate at level $i+1$. If the gate is an OR gate we rewrite all it's inputs as CNFs with clauses of size at most $\log_2 S$, and merge them with the gate obtaining a CNF with clauses of size at most $\log_2 S$. If the gate is an AND gate we similarly obtain a DNF with terms of size at most $\log_2 S$.

Then using Lemma 1 again, the probability that the function computed at the gate at level $i+1$ does not have a decision of height at most $\log_2 S$ after applying ρ_{i+1}, is less than $(7\frac{n_{i+1}}{n_i}\log_2 S)^{\log_2 S} = \frac{1}{S}$.

If in fact $i = d$, that is the gate at level $i+1$ is the top MOD_m gate, we also rewrite all it's inputs as DNFs with terms of size at most $\log_2 S$. Since these are constructed from decision trees, we can argue as in the proof of Theorem 1 to simply absorb the top OR gates into the MOD_m gate, resulting in a $\mathbf{MOD}_m \circ \mathbf{AND}$ circuit with bottom fanin less than $\log_2 S$.

Since the probability of failure at every gate is less than $\frac{1}{S}$, there exists an appropriate restriction ρ, making the restricted function on $n_d = \frac{(cn)^{\frac{1}{d}}}{14c}$ variables be computed by a $\mathbf{MOD}_m \circ \mathbf{AND}$ circuit with bottom fanin less than $\log_2 S$. This implies that the restricted function has MOD_m-degree at most $\log_2 S = \frac{(cn)^{\frac{1}{d}}}{14}$, thus contradicting the existence of C. □

3.3 Proof of Theorem 3

The proof will closely follow the outline of the proof of Theorem 2. Let S and n_{i+1} be as before, where $c < \frac{1}{2}$, and assume assume that C is a $\mathbf{MOD}_m \circ \mathbf{AC}^0_d$ circuit of depth $d+1$ but of size $S^{1-\epsilon}$ computing MAJORITY.

Now since the size of C is only $S^{1-\epsilon}$ the probability that a random restriction $\rho \in R^{n_d}_n$ does not create a $\mathbf{MOD}_m \circ \mathbf{AND}$ circuit with bottom fanin at most $\log_2 S$ is less than $S^{-\epsilon} = o(n^{-\frac{1}{2}})$.

The probability that the number of variables assigned 0 and 1 by ρ differ by at most 1 is at least $\frac{1}{2^{n-n_d}}\binom{n-n_d}{\frac{n-n_d}{2}}$. Using Stirling's approximations this is $\Omega\left((n-n_d)^{-\frac{1}{2}}\right) = \Omega\left(n^{-\frac{1}{2}}\right)$.

Thus for all large enough n there exists a restriction ρ implying the existence of a $\mathbf{MOD} \circ \mathbf{AND}$ circuit with bottom fanin less than $\log_2 S$ computing MAJORITY on n_d variables, contradicting the existence of C as before, now using the MOD_m-degree lower bound for MAJORITY in Lemma 4. □

3.4 Proof of Theorem 4

The proof consists of several cases. We will only give the proof for $\mathbf{MOD}_m \circ \mathbf{\Sigma}_k$ circuits and for $k = 2$ and for odd $k > 2$.

For $k = 2$, assume that C is a $\mathrm{MOD}_m \circ \mathbf{\Sigma}_2$ circuit of size $S = 2^{\frac{1}{24}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}}$ with bottom fanin $r = \frac{1}{24}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}$ computing f^m_2. Pick at random ρ from $R^-_{q,B}$, and apply the restriction $\rho g(\rho)$. Using Lemma 2 the probability that a single $\mathbf{\Sigma}_2$ subcircuit cannot be represented as a CNF with clauses of size at most $s = \frac{1}{12\sqrt{2}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}$ after applying the restriction is at most $(6qr)^s = \left(\frac{1}{2}\left(\frac{\log m}{m}\right)^{\frac{1}{4}}\right)^s \leq 2^{-s}$. Thus the probability of failure at any $\mathbf{\Sigma}_2$ subcircuit is at most $S2^{-s} = 2^{\frac{1}{12}\left(\frac{1}{2}-\frac{1}{\sqrt{2}}\right)\left(\frac{m}{\log m}\right)^{\frac{1}{4}}} = o(1)$.

Thus for m large enough, we can using using Lemma 3 pick a restriction where simultaneously all $\mathbf{\Sigma}_2$ subcircuits can be converted to CNFs with clauses of size at most s and the circuit still computes f^m_1. Now, after the restriction the function computed at every $\mathbf{\Sigma}_2$ subcircuit has *both* a CNF with clauses of size at most s and a DNF with terms of size at most r (since it already had that before applying the restriction). It then follows that they also have decision trees of depth at most sr. These are then converted into DNFs with mutually contradictory terms of size at most sr and the top OR gates can be absorbed into the MOD_m gate, resulting in a $\mathrm{MOD}_m \circ \mathrm{AND}$ circuit with bottom fanin $sr = \frac{1}{288\sqrt{2}}\sqrt{\frac{m}{\log m}}$ computing f^m_1, which is the AND function on $\sqrt{\frac{1}{2}m\log m}$ variables, which contradicts Lemma 4.

For general $k > 2$ the proof goes by induction. Assume C is a $\mathrm{MOD}_m \circ \mathbf{\Sigma}_k$ circuit of size $S = 2^{\frac{1}{12\sqrt{2k}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}}$ with bottom fanin $r = \frac{1}{12\sqrt{2k}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}$

computing f_k^m. Assuming k is odd means that the gates at the bottom are OR gates.

Pick at random ρ from $R_{q,B}^+$, and apply the restriction $\rho g(\rho)$. Then the $\mathbf{\Pi_2}$ subcircuits of C at the two bottom levels can all be converted to $\mathbf{\Sigma_2}$ circuits with bottom fanin $s = \frac{1}{12\sqrt{2(k-1)}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}$ except with probability at most $S2^{-s} = 2^{\left(\frac{1}{\sqrt{k}}-\frac{1}{\sqrt{k-1}}\right)\frac{1}{12\sqrt{2}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}} = o(1)$ and the top OR gates can then be merged with OR gates just above. Now as before, for m large enough, we can pick a restriction obtaining a $\mathrm{MOD}_m \circ \mathbf{\Sigma}_{k-1}$ circuit of size $S \leq 2^{\frac{1}{12\sqrt{2(k-1)}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}}$ and bottom fanin s computing f_{k-1}^m, thereby obtaining a contradiction. □

By adding gates of fanin 1 at the bottom of depth k $\mathbf{MOD}_m \circ \mathbf{AC}^0$ circuits we obtain the following.

Corollary 2. *For sufficiently large m, $\mathbf{MOD}_m \circ \mathbf{AC}_{k-1}^0$ circuits computing f_k^m require size at least $2^{\frac{1}{12\sqrt{2k}}\left(\frac{m}{\log m}\right)^{\frac{1}{4}}}$.*

One can obtain the stronger lower bound $2^{\frac{1}{12\sqrt{2k}}\sqrt{\frac{m}{\log m}}}$ by using the observation of Berg and Ulfberg mentioned in Sect. 2.2.

3.5 Proof of Theorem 5

Much of the proof is the same as the proof of Theorem 2. In particular choose c in the same way, and assume that C is a $\mathbf{AC}_d^0$ circuit, containing a single MOD_m gate G, of depth d and size $\frac{1}{3}S^{\frac{1}{2}}$, where $S = 2^{\frac{1}{14}(cn)^{\frac{1}{d}}}$, that computes a function f with the stated property. Define n_{i+1} and pick the restriction ρ in the same way.

Let C_G be the subcircuit of C with G as its output. Let C_0 and C_1 be the subcircuits obtained from C by substituting the constants 0 and 1 instead of G. We will apply the restriction to all three circuits simultaneously and obtain decision trees of height at most $\frac{1}{2}\log_2 S$ for every gate except G.

At any time, when encountering a NOT gate (which are not included in the counting of levels), by negating the leaves of the decision tree obtained for its input, we can obtain a similar decision tree for itself, and in this way ignore them in this proof.

By the choice of n_{i+1}, we obtain the desired decision trees at every gate except with probability at most $\frac{1}{S^{\frac{1}{2}}}$. When we in fact we encounter the MOD_m gate G, we can as in the proof of Theorem 2 obtain a $\mathbf{MOD}_m \circ \mathbf{AND}$ circuit with bottom fanin less than $\frac{1}{2}\log_2 S$ for the function computed.

Since the probability of failure at every gate is less than $\frac{1}{S^{\frac{1}{2}}}$, and the total number of gates in the 3 circuits is less than $S^{\frac{1}{2}}$, there exists an appropriate restriction ρ, such that we in this way obtain polynomials p, q_0 and q_1 of degree $\frac{1}{2}\log_2 S = \frac{(cn)^{\frac{1}{d}}}{28}$ over $\mathbf{Z}_m$ on $n_d = \frac{(cn)^{\frac{1}{d}}}{14c}$ variables variables, such that $p(x) \not\equiv 0 \pmod m \Leftrightarrow C_{G,\rho}(x) = 1$, $q_0(x) = C_{0,\rho}(x)$ and $q_1(x) = C_{1,\rho}(x)$ for all x.

From these polynomials we will construct a *weak* representation, in the sense of Lemma 8, of either f_ρ or $\neg f_\rho$. We will consider 3 cases.

1. There exists x such that $C_{G,\rho} = 1$ and $C_{1,\rho}(x) = 1$. In this case $p(x)q_1(x)$ weakly represents f_ρ, because q_1 is an *exact* representation of $C_{1,\rho}$.
2. There exists x such that $C_{G,\rho}(x) = 1$ and for every x it holds that $C_{G,\rho}(x) = 1$ implies $C_{1,\rho}(x) = 0$. In this case $p(x)$ weakly represents $\neg f_\rho$.
3. For every x it holds that $C_{G,\rho}(x) = 0$. In this case $q_0(x)$ weakly represents f_ρ, since f_ρ is not identically zero.

Thus we always obtain a weak representation of degree at most $2\frac{1}{2}\log_2 S = \frac{(cn)^{\frac{1}{d}}}{14}$ of either f_ρ or $\neg f_\rho$ on $n_d = \frac{(cn)^{\frac{1}{d}}}{14c}$ variables contradicting the existence of C. Finally, by picking c sufficiently small we obtain the result as stated.

4 Oracle Consequences

Using the approach of Furst, Saxe and Sipser [7] one can use super-quasi-polynomial lower bounds for constant depth circuits to derive oracle separations of corresponding subclasses of PSPACE. Vollmer showed how to obtain such separations in a uniform way under several conditions, obtaining oracle separations for subclasses of PSPACE defined in terms of generalized operators [19].

More precisely, we will use the following part of a more general theorem.

Theorem 9 ([19], Theorem 4.4). *Let $G_1, \ldots, G_k$ and $G'_1, \ldots, G'_l$ be families of boolean gates such that a given function computed by polynomial size $G_1 \circ \cdots \circ G_k$ circuits require $G'_1 \circ \cdots \circ G'_l \circ \Sigma \circ \mathrm{AND}_{\log^{O(1)} n}$ circuits of super-quasi-polynomial size. Then relative to an oracle $G_1 \cdots G_k \cdot \mathrm{P} \not\subseteq G'_1 \cdots G'_l \cdot \mathrm{P}$.*

Here the Σ denotes gates which takes inputs of which at most one of them is 1 at any time, and outputs 1 if in fact any of its inputs is 1. In all our applications these can be removed. Indeed we always have that G'_l is symmetric, and thus they can absorb the sums.

The super-quasi-polynomial lower bounds of this paper would give oracle separations for the classes $\mathrm{MOD}_m\mathrm{P} \cdot \mathbf{\Pi}^p_k$ and $\mathrm{MOD}_m\mathrm{P} \cdot \mathbf{\Sigma}^p_k$. Using the approach of Torán [20] showing that $\mathrm{PP}^K = C \cdot K$ for any class K in the counting hierarchy, one can similarly show that $\mathrm{MOD}_m\mathrm{P}^{\mathbf{\Sigma}^p_k} = \mathrm{MOD}_m\mathrm{P} \cdot \mathbf{\Pi}^p_k$. Now from Theorem 2 we then directly obtain the following, using the theorem of Vollmer.

Corollary 3. *Let m and l be positive integers. If l has a prime factor not dividing m then relative to some oracle, $\mathrm{MOD}_l\mathrm{P} \not\subseteq \mathrm{MOD}_m\mathrm{P}^{\mathrm{PH}}$ and $\mathrm{coMOD}_l\mathrm{P} \not\subseteq \mathrm{MOD}_m\mathrm{P}^{\mathrm{PH}}$. If m is not a prime power then relative to some oracle, $\mathrm{coMOD}_m\mathrm{P} \not\subseteq \mathrm{MOD}_m\mathrm{P}^{\mathrm{PH}}$.*

From Theorem 3 and finally from Theorem 4 we obtain

Corollary 4. *For any integer m, relative to some oracle, $\mathrm{PP} \not\subseteq \mathrm{MOD}_m\mathrm{P}^{\mathrm{PH}}$.*

Corollary 5. *For any m and every $k > 1$, relative to some oracle, $\mathbf{\Pi}^p_k \not\subseteq \mathrm{MOD}_m\mathrm{P}^{\mathbf{\Sigma}^p_{k-1}}$.*

The case $k = 1$ follows by using Lemma 4 instead.

Acknowledgements. The authors are supported by BRICS, Basic Research in Computer Science, a Centre of the Danish National Research Foundation.

References

1. Karchmer, M., Wigderson, A.: Monotone circuits for connectivity require super-logarithmic depth. SIAM Journal on Discrete Mathematics **3** (1990) 255–265
2. Håstad, J., Goldmann, M.: On the power of small-depth threshold circuits. Computational Complexity **1** (1991) 113–129
3. Beame, P.: A switching lemma primer. Technical Report UW-CSE-95-07-01, Department of Computer Science and Engineering, University of Washington (1994) Availible online at www.cs.washington.edu/homes/beame.
4. Berg, C., Ulfberg, S.: A lower bound for perceptrons and an oracle sepration of the $\mathrm{PP}^{\mathrm{PH}}$ hierarchy. Journal of Computer and System Sciences **56** (1998) 263–271
5. Goldmann, M.: On the power of a threshold gate at the top. Information Processing Letters **63** (1997) 287–293
6. Krause, M., Pudlák, P.: Computing boolean functions by polynomials and threshold circuits. Computational Complexity **7** (1998) 346–370
7. Furst, M., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Mathematical Systems Theory **17** (1984) 13–27
8. Håstad, J.: Computational limitations of small-depth circuits. MIT Press (1987)
9. Razborov, A., Wigderson, A.: $n^{\Omega(\log n)}$ lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom. Information Processing Letters **45** (1993) 303–307
10. Razborov, A.A., Rudich, S.: Natural proofs. Journal of Computer and System Sciences **55** (1997) 24–35
11. Zhang, Z.L., Barrington, D.A.M., Tarui, J.: Computing symmetric functions with AND/OR circuits and a single MAJORITY gate. In: Proceedings of the 10th Annual Symposium on Theoretical Aspects of Computer Science. Volume 665 of Lecture Notes in Computer Science., Springer (1993) 535–544
12. Tsai, S.C.: Lower bounds on representing Boolean functions as polynomials in Z_m. SIAM Journal on Discrete Mathematics **9** (1996) 55–62
13. Aspnes, J., Beigel, R., Furst, M., Rudich, S.: The expressive power of voting polynomials. Combinatorica **14** (1994) 1–14
14. Beigel, R.: When do extra majority gates help? $\mathrm{Polylog}(n)$ majority gates are equivalent to one. Computational Complexity **4** (1994) 314–324
15. Barrington, D.A.M., Straubing, H.: Complex polynomials and circuit lower bounds for modular counting. Computational Complexity **4** (1994) 325–338
16. Barrington, D.A.M., Beigel, R., Rudich, S.: Representing Boolean functions as polynomials modulo composite numbers. Computational Complexity **4** (1994) 367–382
17. Green, F.: A complex-number fourier technique for lower bounds on the mod-m degre. Computational Complexity **9** (2000) 16–38
18. Tarui, J.: Degree complexity of Boolean functions and its applications to relativized separations. In: Proceedings of the Sixth Annual Structure in Complexity Theory Conference, IEEE Computer Society Press, (1991) 382–390
19. Vollmer, H.: Relating polynomial time to constant depth. Theoretical Computer Science **207** (1998) 159–170
20. Torán, J.: Complexity classes defined by counting quantifiers. Journal of the ACM **38** (1991) 753–774

The Enumerability of P Collapses P to NC

Alina Beygelzimer and Mitsunori Ogihara

[1] IBM T.J. Watson Research Center, Hawthorne, NY 10532.
beygel@us.ibm.com
[2] Department of Computer Science, University of Rochester, Rochester, NY 14627.
ogihara@cs.rochester.edu

Abstract. We show that one cannot rule out even a single possibility for the value of an arithmetic circuit on a given input using an NC algorithm, unless P collapses to NC (i.e., unless all problems with polynomial-time sequential solutions can be efficiently parallelized). Thus excluding any possible solution in this case is as hard as finding the solution exactly. The result is robust with respect to NC algorithms that err (i.e., exclude the correct value) with small probability. We also show that P collapses all the way down to NC^1 when the characteristic of the field that the problem is over is sufficiently large (but in this case under a stronger elimination hypothesis that depends on the characteristic).

1 Introduction

Can ruling out any possible solution to a problem be as hard as solving the problem exactly? We consider the task of computing the value of an arithmetic circuit on a given input, both over some finite field $\mathrm{GF}(q)$. Suppose that one could rule out a single possibility for this value, via an NC algorithm. We show that then the Boolean Circuit Value Problem (CVP for short) is in NC. Since CVP is complete for P under NC^1-computable many-one reductions, this would collapse P to NC. Thus one cannot rule out *any* candidate for the value of an arithmetic circuit on a given input without proving that there are no inherently sequential problems in P.

The task of ruling out possible values can be formalized using the notion of *enumerability,* introduced by Cai and Hemachandra [8] (and used to prove a number of results since then, e.g., [9,1,5,3]). Let $k : \mathbb{N} \to \mathbb{N}^+$ be a monotone non-decreasing function. A $k(n)$-*enumerator* for a function f is a procedure that reduces the set of possibilities for the value of f on a given input x of length n by enumerating a list of $k(n)$ values guaranteed to contain $f(x)$ (thereby eliminating the rest of the values). For such a procedure to be interesting, it has to do something different than simply compute $f(x)$ and include it in the output; hence we consider enumerators that are, in some computational sense, weaker than the best known algorithms for the functions they enumerate. An enumerator yields an approximation of f, but instead of restricting the value of $f(x)$ to an interval (as would a classical approximator), it restricts this value to a set of possibilities. The enumerability assumption in this paper is much weaker

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 346–355, 2004.
© Springer-Verlag Berlin Heidelberg 2004

than usual: instead of reducing the set of candidates substantially, we just want to exclude a single possibility.

We initiate the study of probabilistic enumerators that are allowed to err (i.e., exclude the correct value) with small probability. The result is robust with respect to such enumerators. We also prove a stronger collapse of P all the way down to NC^1 for the special case when the characteristic of the ground field is sufficiently large (and under a stronger hypothesis that depends on the characteristic). More precisely, let $\mathbb{F}$ be any fixed finite field with characteristic at least k^2 for some integer $k \geq 2$. We show that $\mathrm{CVP}_{\mathbb{F}}$ is not k-enumerable, unless $\mathrm{CVP}_{\mathbb{F}}$ is solvable in NC^1, modulo the complexity of the enumerator. If $\mathbb{F}$ is not fixed but is of size at most polynomial in the size of the input, then the construction is in TC^1.

The first collapse is proven by adapting the techniques used in Sivakumar's proof that SAT is not $O(\log n)$-membership comparable[1], unless UniqueSAT is in P [24], which builds on the technique of Ar et al. [2] for reconstructing polynomials from noisy data. The second, stronger collapse all the way down to NC^1 is proven using different techniques.

2 Preliminaries

Let k be a positive integer. We denote the set $\{1, \ldots, k\}$ by $[k]$. A *set system* is a subset of $\mathcal{P}(k)$, the power set of $[k]$. We write $\langle x_1, \ldots, x_n \rangle$ to denote the n-tuple of $x_1, \ldots, x_n$ as well as the encoding of this n-tuple as a bit string, using a fixed, well-behaved pairing function extended to tuples (computable and invertible in time and space affordable in the context).

An *arithmetic circuit* over some algebraic structure $\mathbb{F}$ is a circuit with gates computing operations over $\mathbb{F}$ and taking elements and constants of $\mathbb{F}$ as inputs. There is a unique output gate. A circuit with n inputs computes a function from $\mathbb{F}^n$ to $\mathbb{F}$ in the natural way. A Boolean circuit is just an arithmetic circuit over GF(2).

We use standard definitions of circuit complexity classes. Since we want to make circuit classes comparable to uniform classes defined in terms of time and space, we need to place uniformity restrictions on circuit families. For our purposes, it will be sufficient to use logspace uniformity, meaning that there exists a logspace machine that, on input 1^n, generates a standard encoding [21] of the n-th circuit in the family.

We use Cook's generalization [11] of Wilson's model [29] to define circuit-based reductions. (We do not make a distinction between circuits computing functions and circuits deciding languages.) In this model, circuits are allowed to have oracle gates, in addition to the usual gates. An oracle gate takes an ordered list of r inputs, interpreted as a query string. The sequence of s ordered outputs corresponds to the value of the oracle on the query string. For the purpose of

[1] A $g(n)$-*membership comparing function* for a language A takes a sequence of $g(n)$ strings, each of length at most n, and outputs a string in $\{0,1\}^{g(n)}$ that is not their characteristic string in A. (See [19,4].)

defining depth in bounded fan-in circuits, an oracle gate with r inputs and s outputs counts as depth $\lceil \log(r+s) \rceil$.

An instance of the *arithmetic circuit value problem* over $\mathbb{F}$ (denoted $\mathrm{CVP}_{\mathbb{F}}$) is an encoding of an arithmetic circuit C over $\mathbb{F}$, together with an encoding of inputs $x_1, \ldots, x_n$ from $\mathbb{F}$. The circuit C has n inputs, and the problem is to compute the value of C on $x = \langle x_1, \ldots, x_n \rangle$, denoted $C(x)$.

Ladner [15] introduced the (Boolean) circuit value problem, and showed that it is complete for P (under logspace computable many-one reductions). Cook observed that the problem is actually complete under NC^1 reductions [11]. For infinite fields, the problem is not necessarily in FP, since gate values need not be of polynomial length. It is in FP, however, for any finite field, and remains FP-hard in any field. Correspondingly, it is P-hard to decide whether C on x outputs 1, where 1 is the multiplicative identity of the field. (See [13], Appendix A.1.) The fields we consider are finite. The hardness above is with respect to NC^1-computable many-one reductions; hence it suffices to show that CVP (or $\mathrm{CVP}_{\mathbb{F}}$) is in NC^i to collapse P to NC^i for some $i \geq 1$. We will establish such a collapse under the hypothesis that, given an instance $\langle C, x \rangle$ of $\mathrm{CVP}_{\mathbb{F}}$, one can efficiently eliminate at least *one* of $|\mathbb{F}|$ possibilities for the value of C on x.

Let $k : \mathbb{N} \to \mathbb{N}^+$ be a monotone non-decreasing function. Recall that a *$k(n)$-enumerator* for a function f, defined by Cai and Hemachandra [8], is a deterministic procedure E_f that on input x of length n outputs a set of $k(n)$ values, one of which is guaranteed to be $f(x)$. This yields a certain notion of approximation: instead of restricting $f(x)$ to an interval (for example, by outputting a value guaranteed to be within some multiplicative factor from $f(x)$), an enumerator restricts $f(x)$ to a small set, namely $E_f(x)$. The number of elements in this set depends only on n. In contrast, the size of the approximating interval is a function of $f(x)$, and thus if $f(x)$ is exponentially large, then so is the interval. Besides the case when the range is exponentially large (with the permanent, determinant, and #GA being natural examples), enumerative counting may be an approximation of choice in a more general context where there is no natural ordering on the range of f, hence restricting the value to an interval is not particularly meaningful (as, arguably, in the case of computations over finite fields).

3 Finite Fields of Large Characteristic

We first prove a deeper collapse mentioned in the introduction (for fields with large characteristic).

Theorem 1. Let $\mathbb{F}$ be a (fixed) finite field with characteristic at least k^2 for some integer $k \geq 2$, and let $i \geq 1$. If $\mathrm{CVP}_{\mathbb{F}}$ is k-enumerable in NC^i, then $\mathrm{CVP}_{\mathbb{F}}$ is in NC^i, modulo the complexity of the enumerator. If $\mathbb{F}$ is described in the input [2], then the same result holds with TC^i in place of NC^i.

[2] A Note on Representation: To implement the arithmetics in a finite field $\mathbb{F}$ with p^q elements, it suffices to have a monic irreducible polynomial g in $\mathbb{Z}_p[x]$ of degree q,

Proof: Let $\langle C, x\rangle$ be an instance of $\mathrm{CVP}_{\mathbb{F}}$, where C is an encoding of an arithmetic circuit over $\mathbb{F}$ with n inputs, and x refers to the encoding of $\langle x_1, \ldots, x_n\rangle \in \mathbb{F}^n$. To avoid new notation, assume that C has n gates. Notice that, given an assignment to the gates of C, we can check it for validity in NC^1; we just need to locally verify that the value assigned to every gate is in fact the value of the operation it computes on the values assigned to its inputs. Thus all we need is a constructible in NC^1 set of gate assignments containing the correct one for C on input x. Of course, we cannot simply run the enumerator on every gate of C (viewing it as an output gate), as it would yield exponentially many assignments. Instead we will build a new circuit on top of C such that the enumerator will be forced to reduce the set of assignments to polynomially many.

Consider any two gates g and h of C. We can run the enumerator to obtain k candidates for the value computed by each on input x. Let the corresponding lists of candidates be $a = \{a_1, \ldots, a_k\}$ and $b = \{b_1, \ldots, b_k\}$. (We may clearly assume that all values on each list are distinct.) We will view g as a univariate degree $k-1$ polynomial $P_a(u) = \sum_{j=0}^{k-1} \beta_j u^j$ in $\mathbb{F}[u]$ such that $P_a(a_i) = d(i-1)$ for every $i \in [k]$, where d is a generator of an additive group of $\mathbb{F}$ of size at least k^2. Recall that the promise is that $\mathbb{F}$ has characteristic at least k^2. The multiplicative identity 1 of $\mathbb{F}$ is such a generator, so we can take $d = 1$. Similarly define P_b for h. Thus each of the polynomials is given by its value on a set of k distinct points in $\mathbb{F}$ specified by the enumerator. Since both polynomials are of degree $k-1$, the coefficients (we will follow P_a) can be recovered by interpolation as the solution to the $k \times k$ Vandermonde system

$$\begin{pmatrix} a_1^0 & \cdots & a_1^{k-1} \\ a_2^0 & \cdots & a_2^{k-1} \\ \vdots & \ddots & \vdots \\ a_k^0 & \cdots & a_k^{k-1} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{k-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ k-1 \end{pmatrix}.$$

Since $a_1, \ldots, a_k$ are distinct, the system has full rank over $\mathbb{F}$. If k is a constant independent of n, we can find the solution using a circuit of constant size. When k and $|F|$ are polynomial in n (i.e., our instance of the circuit-value problem is over a field that depends on the size of the instance), we can solve such systems in logspace uniform TC^0 [10]. (See also [12,20].) The proofs there are for fields of characteristic 2, but extensions to finite fields of characteristic other than 2 are straightforward.

---

since $\mathbb{F}$ can be viewed as $\mathbb{Z}_p[x]/(g)$, the ring of polynomials over $\mathbb{Z}_p$ modulo g. If the field $\mathbb{F}$ is fixed, then such g can be precomputed in logspace. If $\mathbb{F}$ depends on the instance length n, then such an irreducible polynomial can be specified as a part of the input by its characteristic p, order p^q (in unary, so p^q is polynomial in n), and a monic irreducible polynomial $g \in \mathbb{Z}_p[x]$ of degree q. Note that the characteristic of $\mathbb{F}$ is *small* in the sense that the parallel complexity is measured in terms of $\log p$, rather than $\log \log p$. In what follows, we assume that the representation of $\mathbb{F}$ has been fixed.

Let us return to our gates g and h with associated polynomials $P_a(u)$ and $P_b(v)$ (whose coefficients we now know). We will create a new gate f with inputs g and h. The function computed by f will be $F(u,v) = kP_a(u) + P_b(v)$. Thus $F(a_i, b_j) = k(i-1) + (j-1)$ for every $i, j \in [k]$. If we run the enumerator on $\langle f, x\rangle$, we will get back k candidates for $f(x)$, each corresponding to a *unique* combination of claimed values for $g(x)$ and $h(x)$, provided that $k^2 \leq p$. In particular, if the enumerator claims that $f(x) = c$, then c must be in the form $k(i-1) + (j-1)$ for some $i, j \in [k]$, yielding the claim that $g(x) = a_i$ and $h(x) = b_j$. Otherwise, $f(x) \neq c$.

Now we can group the gates of C in pairs (arbitrarily) constructing a new level of gates as described above. We repeat the same procedure for the new level; continuing this way, eventually we get to a single gate (after having constructed a full binary tree on top of the original gates). We run the enumerator on this gate, obtaining k candidates for its value on input x. Each candidate value corresponds, via a downward chain of decodings, to a unique gate assignment for the original gates of C. As long as k is polynomial in n, we can verify all candidate assignments in parallel, singling out the correct one.

We now describe the structure of the circuit that does this. Our parallel algorithm will first run the enumerator on every gate g_i in C. Each enumerator gate takes a description of C with a marked output gate g_i, together with the original input x, and outputs a list of k numbers, one of which is $g_i(x)$. Let us return to our pair of gates g and h with lists $(a_1, \ldots, a_k)$ and $(b_1, \ldots, b_k)$. All other pairs are treated in the same way in parallel. We feed the lists and the original input x to auxiliary circuitry $\textsc{Aux}(g,h)$ that outputs a description of a new circuit with the output gate f. (The new circuit contains the original C plus the circuitry that has already been built on top of it. The input to the new circuit will be the original input x.) $\textsc{Aux}(g,h)$ reconstructs the polynomials P_a, P_b, and uses them to compute a description of f. It follows from the above discussion that $\textsc{Aux}(g,h)$ can be implemented in NC^0 if k is constant, and in TC^0 if p^q (and thus k) are polynomial in n. (Given P_a and P_b, the description of f can clearly be computed by NC^0 and TC^0 circuits respectively.) Since there are $O(\log n)$ levels, the entire construction is in NC^1 (respectively in TC^1). Note that the output of an NC^0 circuit can depend only on constantly many inputs, thus improving the complexity significantly in the case of non-constant k appears impossible. Once we get to a single gate, we run the enumerator on this gate, obtaining k candidates for its value on x. Each value can be uniquely passed back to the original gates of C. Passing amounts to finding i and j from the value of $k(i-1) + (j-1)$, which can obviously be done in NC^0 or TC^0, depending on whether k is constant. □

4 Main Theorem

We now turn to arbitrary finite fields.

Theorem 2. Let $\mathbb{F}_q$ be a finite field with q elements, and let $i \geq 3$ be an integer. If $\mathrm{CVP}_{\mathbb{F}_q}$ is $(q-1)$-enumerable in NC^i then $\mathrm{CVP} \in \mathrm{NC}^i$.

Remark. To avoid inessential complications with the presentation (in particular, to simplify the Boolean complexity of field operations), we will take $q = 2^r$. The result readily holds for any structure of $\mathbb{F}$. Similarly to Theorem 1, the result also holds for fields that are not fixed but are given as a part of the input.

Proof: Let $\langle C, x\rangle$ be the instance of the (Boolean) circuit-value problem whose solution we wish to find, and let $g_0, \ldots, g_{n-1}$ denote the gate values of C on input x. We view the gate values of C on x as the degree $(n-1)$ univariate polynomial $P_g(u) = \sum_{i=0}^{n-1} g_i u^i$ with coefficients from GF(2). We will try to get some information about the value of this polynomial on all elements an appropriately constructed field $\mathrm{GF}(2^m)$, where $m \sim \log n$ (to be chosen later). [3] Each polynomial in $\mathrm{GF}(2^m)$ can also be represented as an m-bit string, treating bits as the corresponding coefficients.

We will use our $(q-1)$-enumerator to reduce the number of candidates for $P_g(u)$ for each $u \in \mathrm{GF}(2^m)$, and then use the technique of Ar et al. [2] (as in [24]) to reconstruct the coefficients of P_g, which are the desired gate values of C on x.

Fix $u \in \mathrm{GF}(2^m)$. The value of $P_g(u)$ is an element of $\mathrm{GF}(2^m)$, and thus has an m-bit representation. We will use the enumerator to eliminate one possible setting for each subset of r bits of $P_g(u)$ by disguising this question as an instance of $\mathrm{CVP}_{\mathbb{F}_q}$ whose solution (viewed as an r-bit string) gives the desired r bits of $P_g(u)$. We will then use Sauer's lemma (as in [24]) to argue that this rules out a large portion of all possible settings for all the m bits.

To encode r-element subsets as instances of $\mathrm{CVP}_{\mathbb{F}_q}$, we will need a bit version of $\mathrm{CVP}_{\mathbb{F}_q}$, consisting of tuples $\langle D, y, i\rangle$ such that D is an encoding of an arithmetic circuit, y is an input to D, both D and y are over $\mathbb{F}_q$, and the i-th bit of $D(y)$ is 1, when $D(y)$ is viewed as an r-bit string. It is readily seen that the bit version is complete for P. Consider the set

$$A = \{\langle C, x, u, i\rangle \mid \text{the } i\text{-th bit of } P_g(u) \text{ is } 1\},$$

where C, x, u, and $P_g(u)$ are defined as above. Clearly A is in P, since the required $\mathrm{GF}(2^m)$-arithmetic needed to compute any bit of $P_g(u)$ can be easily implemented in NC^1. Thus A many-one reduces (via some NC^1-computable function h) to the bit version of $\mathrm{CVP}_{\mathbb{F}_q}$. We will use h to map each bit of $P_g(u)$ to an instance of the bit version of $\mathrm{CVP}_{\mathbb{F}_q}$. Our parallel algorithm will first run h (in parallel) on all tuples $\langle C, x, u, i\rangle$ for i from 0 to $m-1$. Given $\langle C, x, u, i\rangle$, h produces an encoding of some circuit D_i together with an input y_i, both over $\mathbb{F}_q$, and a number j_i such that the i-th bit of $P_g(u)$ is precisely the j_i-th bit of D_i on y_i.

Let us fix a subset S of r bits of $P_g(u)$ and turn to the task of eliminating a possible setting of these bits. Denote the r-bit string corresponding to $P_g(u)$ restricted to the bits in S by $w = w_0 \ldots w_{r-1}$. Notice that the bits of w are essentially the bits j_i of $D_i(y_i)$, where i ranges over the bits specified by 5. We will view these output bits (denoted $w_0, \ldots, w_{r-1}$ to avoid new notation)

[3] Having the power of NC^3, we can find a generating polynomial exhaustively by cycling through monic polynomials of degree m in $\mathbb{Z}_2[x]$ until an irreducible is found, and construct $\mathrm{GF}(2^m)$ as the ring of polynomials over $\mathbb{Z}_2$ modulo this polynomial.

as specifying the coefficients of the polynomial $P_w(x) = \sum_{i=0}^{r-1} w_i x^i$ over GF(2). Here x is our original input in the Boolean instance of CVP that we are solving. We view $\mathbb{F}_q$ as $\mathbb{Z}_2[z]/(h)$, the ring of polynomials over $\mathbb{Z}_2$ modulo the irreducible polynomial h of degree r in $\mathbb{Z}_2[z]$. (We can find h exhaustively.) Thus P_w is an element of $\mathbb{F}_q$. We will construct an arithmetic circuit D_S with an input *ys* (both over $\mathbb{F}_q$) such that the value of D_S on *ys,* viewed as a polynomial in $\mathbb{Z}_2[z]/(h)$, is equal to P_w. Since the degree of P_w is $r-1$, $D_S(y_S)$ uniquely defines P_w. In fact, we already have such D_S and *ys* at hand. Our *ys* will be the collection of inputs $\{y_i \mid i \in S\}$, and D_S will contain the circuitry $\{D_i \mid i \in S\}$. Each $D_i(y_i)$ (which is an r-bit number) encodes the corresponding coefficient of P_w as its j_i-th bit. The circuit D_S will output these r bits as an r-bit representation of its output (which is an element of $\mathbb{F}_q$). The circuitry needed to do such compilation of outputs is certainly in logspace uniform NC^1. We can finally run the $(q-1)$-enumerator for $\mathrm{CVP}_{\mathbb{F}_q}$ on $\langle D_S, y_S \rangle$ to get a list of $q-1$ values in $\mathbb{F}_q$. We can similarly eliminate one possible setting of any other subset of r bits of $P_g(u)$. Not only can we do it for all sets S in parallel (u has been fixed up until now), but also for all $u \in \mathrm{GF}(2^m)$. Recall that $r = c \log n$, and $m = d \log n$ with $d > c$ to be fixed later. There are 2^m many u's, and $\binom{m}{r} < 2^m = n^{O(1)}$ many subsets for each such u.

How many possible bit settings for each $P_g(u)$ does it rule out? This number follows immediately from (a rather special case of) a lemma proved independently by Sauer [22], Perles and Shelah [23], and Vapnik and Chervonenkis [25]. (For an excellent exposition, including the lemma's proof, see Bollobás [6, p.131].) To state the lemma, we will say that a set $I \subset [m]$ is *traced* by a set system $\mathcal{F} \subset \mathcal{P}(m)$ if the collection of intersections $\{F \cap I \mid F \in \mathcal{F}\}$ contains every element of $\mathcal{P}(I)$. The lemma states that any $\mathcal{F}$ traces at least $|\mathcal{F}|$ sets in $\mathcal{P}(m)$. In particular, this says that if $|\mathcal{F}| > \sum_{i=0}^{r-1} \binom{m}{i}$, then $\mathcal{F}$ must trace some set of size r. For us, $\mathcal{F}$ is the set of remaining possible assignments to the m bits of $P_g(u)$. We know that it does not trace any subset of r bits (since we excluded at least one possible intersection for each such subset), thus the contrapositive of the lemma says that we are left with at most $\sum_{i=0}^{r-1} \binom{m}{i} < r\binom{m}{r}$ possibilities for $P_g(u)$ (out of 2^m such possibilities). Likewise for every other $u \in \mathrm{GF}(2^m)$.

Let $a = d/c$. We use a bound on $\binom{m}{r} = \binom{m}{m/a}$ in the form: $\binom{m}{m/a} < C(a) \cdot (ea)^{n/a}$, where $C(a)$ is some constant that depends on a. A number of different estimates of this form are known; which one is used does not affect the result. We want $k \doteq 2^{m/4} > C(a) \cdot (ea)^{n/a}$, so given c, we take $d = ac > 4$ to be any easily computable constant such that a satisfies the condition above.

It remains to reconstruct the coefficients of P_g using the restricted lists of possibilities for its value on a set of 2^m points. As in [24], we will use the reconstruction technique of Ar et al. [2], which reduces to the factorization of an appropriately constructed bivariate polynomial $Q(u, v)$ which is zero at all points (u, v) such that $u \in \mathbb{F}_q$ and v is in the list of possibilities for $P_g(u)$. The key claim is that $(v - P_g(u))$ must appear as an irreducible factor of Q. The degree of Q will be polynomial in n in both u and v. The factorization thus gives a list of polynomially many candidates for P_g, each corresponding to a gate assignment

for C on x. All of them can be verified in parallel in NC^1, and the correct one can be singled out.

The statements regarding Q need some justification. For completeness (and since we need to bound the parallel complexity rather than sequential time), we sketch the technique of Ar et al. [2].

Reconstruction. Let $\mathbb{F} = \mathrm{GF}(2^m)$ be the extension field. We will construct a bivariate non-zero polynomial $Q(u, v)$ of degree $\deg(u)$ in u and $\deg(v)$ in v (to be fixed momentarily), such that $Q(u, v) = 0$ for all $u \in \mathbb{F}$ and all v in the list of k possibilities for $P_g(u)$, and such that both u and v are present in every monomial of Q. This gives a homogeneous system of $k2^m$ linear equations in $\deg(u) \cdot \deg(v)$ unknowns. If $\deg(u) \cdot \deg(v) \geq k2^m$, we are guaranteed a solution. It remains to show that $v - P_g(u)$ will appear as an irreducible factor. Consider the univariate polynomial $Q_g(u) = Q(u, P_g(u))$, which is of degree at most $\deg(u) + n\deg(v)$ in u. We know that $Q_g(u)$ is 0 on every $u \in \mathrm{GF}(2^m)$. Thus if $\deg(u) + n\deg(v) < 2^m$, $Q_g(u)$ must be identically zero, implying that $(y - P_g(u))$ divides $Q(u, v)$. Let us choose $\deg(v)$ and $\deg(v)$ leading to this desired situation (for sufficiently large n). Recall that $k = m2^{m/4}$. We have to satisfy $\deg(u)\deg(v) \geq m2^{5m/4}$ and $\deg(u) + 2^{\frac{m}{4}}\deg(v) < 2^m$, which can be done by setting, for example, $\deg(u) = \deg(v) = 2^{\frac{2m}{3}}$.

Note crucially that $\mathbb{F}$ has polynomially in n many elements, and thus its characteristic p is small, i.e., we need the algorithms to run in depth polynomial in $\log p$ (rather than $\log \log p$). In what follows, n (the length of the main input) determines the complexity and uniformity of circuits. All circuits are logspace uniform. Univariate polynomials of degree $n^{O(1)}$ over $\mathbb{F}$ (not necessarily square-free) can be factored in NC^3. A deterministic version of the univariate factoring algorithm of von zur Gathen [26], obtained by plugging in the deterministic NC^2 matrix rank algorithm of Mulmuley [18], works in depth $O(\log^2 n \log(np))$ (bounded fan-in, polynomial time). Thus if p is polynomial in n, we are in NC^3 (as noted in [28]). Bivariate polynomial factoring can be reduced to univariate factoring using Hensel's lifting. Let $f(x, y) \in \mathbb{F}[x, y]$ be the polynomial that we wish to factor into irreducible factors. If $\mathbb{F}$ is a unique factorization domain (which it is, since $\mathbb{F}$ is a field), then so are $\mathbb{F}[x]$ and $\mathbb{F}[x, y]$, thus the factorization is unique (up to the order of factors). We will only seek some irreducible factor of f, hence we will then be able to compute the complete factorization in parallel for all factors.

Let the degree of f be at most $d = n^{O(1)}$ in each variable. We seek an irreducible factor $g \in \mathbb{F}[x, y]$ of f. Hensel's lifting extracts information about the factors of $f(x, y)$ by lifting up a univariate factorization of $f_0(x) = f(x, 0)$. Factoring the univariate polynomial $f_0(x) = g(x, 0)h(x, 0)$ is equivalent to factoring $f(x, y) = g(x, y)h(x, y) \bmod y$. Hensel's construction will lift this factorization to the one modulo $y^2, y^{2\cdot 2}$, and so on, until we have a factorization modulo y^{2^t}, where t is sufficiently large to yield the true factorization of $f(x, y)$. Due to space limitations, we will not describe the reduction. (See, for example, a book by von zur Gathen and Gerhard [27].) We only need to show that it is in logspace uniform NC^3. The univariate factorization algorithm runs

in NC^3. The cost of each lifting step is essentially the cost of computing the quotient and remainder of polynomials of degree at most d in $\mathbb{F}[x]$, and is in NC^1 [12], putting $t = O(\log n)$ iterations in NC^2. Once we have a factorization $f(x,y) \equiv g_t(x,y)h_t(x,y) \pmod{y^{2^t}}$, we need to find polynomials $G(x,y)$ and $D(x,y)$ satisfying $G(x,y) \equiv g_t(x,y)D(x,y) \pmod{y^{2^t}}$, as well as certain conditions on the degrees. This step involves solving a system of linear equations with unknowns being the coefficients of $G(x,y)$ and $D(x,y)$, which is in NC^2 [7]. If $\gcd(G,f)$ is non-trivial, G is a factor of f; otherwise f is irreducible. The gcd computation is in NC^2 [7]. The computation of all factors (originating from different ways to factorize $f_0(x)$ as the product of a monic irreducible $g_0(x)$ and a relatively prime $h_0(x)$) can be done in parallel. □

Corollary 1. Let $\mathbb{F}_q$ be the finite field with q elements, and let $i \geq 3$ be an integer. If $\mathrm{CVP}_{\mathbb{F}_q}$ is $(q-1)$-enumerable in NC^i then $\mathrm{P} = \mathrm{NC}^i$. The field $\mathbb{F}_q$ can be given by q (in unary notation) as a part of the input.

Probabilistic Enumerators. We state an analog of Theorem 2 in the probabilistic setting when the enumerator is allowed to err with small probability. The proof can be found in the full version of the paper. For the definition of randomized Boolean circuits and the class RNC, the reader is referred to Cook [11]. It is believed highly unlikely that $\mathrm{P} \subseteq \mathrm{RNC}$. (Note, however, that randomized cicuits inherently have multiple access to randomness, and RNC^1 is not known (or believed) to be contained in randomized logspace.) An enumerator E_f for a function f is said to have error $\epsilon(\cdot)$ if for all inputs x in the domain of f, the set $E_f(x)$ includes $f(x)$ with probability at least $1 - \epsilon(|x|)$, where the probability is taken over the random bits fed to E_f. Here $\epsilon : \mathbb{N} \to (0, \frac{1}{2})$ is some monotone non-increasing function.

Theorem 3. (Randomized analogue of Theorem 2) Let $\mathbb{F}_q$ be as in Theorem 2. There is no $(q-1)$-enumerator for $\mathrm{CVP}_{\mathbb{F}_q}$ with error at most inversely polynomial unless $\mathrm{P} \subseteq \mathrm{RNC}^3$.

Acknowledgements. We would like to thank Joachim von zur Gathen for referring us to the results on the parallel complexity of polynomial factorization. We would also like to thank Eric Allender for pointing out an alternative interpretation of the results in terms of resource-bounded Kolmogorov complexity.

References

1. A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query classes and nonuniform complexity. In the *5th Structure in Complexity Theory Conference,* pages 232–243, 1990.
2. S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing,* 28(2):487–510, 1998.
3. R. Beals, R. Chang, W. Gasarch, and J. Torán. On finding the number of graph automorphisms. *Chicago Journal of Theoretical Computer Science,* 1999(1).

4. R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Information and Computation,* 120(2):304–314, 1995.
5. A. Beygelzimer and L. Ogihara. The (non)enumerability of the determinant and the rank. *Theory of Computing Systems,* 36(4): 359–374.
6. B. Bollobás. *Combinatorics.* Cambridge University Press, 1986.
7. A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control,* 52:241–256, 1982.
8. J. Cai and L. Hemachandra. Enumerative counting is hard. *Information and Computation,* 82(1):34–44, 1989.
9. J. Cai and L. Hemachandra. A note on enumerative counting. *Information Processing Letters,* 38(4):215–219, 1991.
10. J. Cai and D. Sivakumar. Resolution of Hartmanis' conjecture for NL-hard sparse sets. *Theoretical Computer Science,* 240:257–269, 2000.
11. S. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control,* 64:2–22, 1985.
12. W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing,* 18(5):955–976, 1989.
13. R. Greenlaw, H. Hoover, and W. Ruzzo. *Limits to parallel computation.* Oxford University Press, NY, 1995.
14. E. Kaltofen. Polynomial factorization, 1987–1991. In I. Simon, editor, LATIN 92, *Lecture Notes in Computer Science,* 583:294–313, 1992.
15. R. Ladner. The circuit value problem is log space complete for P. *ACM SIGACT News,* 7(1):18–20, January 1975.
16. R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications.* Cambridge University Press, 1994.
17. J. van Lint. *Introduction to coding theory.* Springer-Verlag, 1991.
18. K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica,* 7:101–104, 1987.
19. M. Ogihara. Polynomial-time membership comparable sets. *SIAM Journal on Computing,* 24(5):1068–1081, 1995.
20. F. Preparata. Inverting a Vandermonde matrix in minimum parallel time. *Information Processing Letters,* 38:291–294, 1991.
21. W. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences,* 22(3):365–383, 1981.
22. N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory (A),* 13:145–147, 1972.
23. S. Shelah. A combinatorial problem: Stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics,* 41:247–261, 1972.
24. D. Sivakumar. On membership comparable sets. *Journal of Computer and System Sciences,* 59(2):270–280, 1999.
25. V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Prob. Appl,* 16:264–280, 1971.
26. J. von zur Gathen. Parallel algorithms for algebraic problems. *SIAM Journal on Computing,* 13:802–824, 1984.
27. J. von zur Gathen and J. Gerhard. Modern Computer Algebra. *Cambridge University Press,* Cambridge, 1999.
28. J. von zur Gathen and G. Seroussi. Boolean circuits versus arithmetic circuits. *Information and Computation,* 91:142–154, 1991.
29. C. Wilson. Relatizived circuit complexity. *Journal of Computer and System Sciences,* 31(2):169–181, 1985.

On NC^1 Boolean Circuit Composition of Non-interactive Perfect Zero-Knowledge

Alfredo De Santis[1], Giovanni Di Crescenzo[2], and Giuseppe Persiano[1]

[1] Dipartimento di Informatica ed Applicazioni,
Università di Salerno, Baronissi (SA), Italy.
{ads,giuper}@udsab.dia.unisa.it
[2] Telcordia Technologies, Piscataway, NJ, USA.
giovanni@research.telcordia.com

Abstract. Non-Interactive Perfect Zero-Knowledge (NIPZK) and Perfect Zero-Knowledge (PZK) are the class of languages having a Perfect Zero-Knowledge proof system in the non-interactive and interactive model, respectively.
In this paper we present new techniques for Boolean Circuit Compositions of NIPZK and PZK, and significantly enlarge the class of known languages having such proofs. Our main result is that all NC^1 circuit compositions over a certain class of languages (that includes for example quadratic residuosity languages) have NIPZK proofs. Previous results only applied to single threshold gates and certain CNF formulae.
We also extend the class of known languages in PZK by allowing compositions over random self-reducible languages with respect to polynomial-size monotone circuits with fan-out > 1 and certain additional restrictions on the allowed gates. Previous results only applied to polynomial-size formulae (that is, circuits with fan-out = 1).

1 Introduction

The paradoxical concept of zero-knowledge proofs has been introduced in the seminal paper [15] that gave perfect zero-knowledge proofs for the number-theoretic languages of quadratic residuosity and quadratic non residuosity modulo a composite integer. A zero-knowledge proof is a method allowing a prover to convince a polynomial time bounded verifier that a certain statement $'x \in L'$ is true without revealing any additional information. Since their introduction, zero-knowledge proofs have received a lot of attention for their applications to Cryptography and their relevance in Complexity Theory.

Interactive and Non-Interactive Zero-Knowledge. As the original model for zero-knowledge proofs allowed both private randomness and interaction between prover and verifier, a different model was introduced to simplify the ingredients necessary to achieve zero-knowledge proofs, and therefore improve their efficiency: the non-interactive model [5,4]. In this model prover and verifier share a public random string and the communication consists of a single message sent from prover to verifier.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 356–367, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Computational and Perfect Zero-Knowledge. Zero-knowledge proofs come in two main variants: computational and perfect, according to the level of indistinguishability between the verifier's view during a protocol and the output of an efficient algorithm, called the simulator. In computational zero-knowledge these two distributions are indistinguishable to any polynomial-time algorithm, while in perfect zero-knowledge they are equal. Perfect zero-knowledge implies computational zero-knowledge; on the other hand, the class of languages that are known to have a proof system of the latter type is much wider than proof systems of the former type. Specifically, assuming the existence of non-uniformly secure one-way functions, first any language in NP [13] and then any language having an interactive proof system [16,2] have been shown to have a computational zero-knowledge proof system in the interactive model. Assuming the hardness of a number-theoretic problem, in the non-interactive model, [5,4] proved that any languages in NP (in fact, any language in AM) has a computational zero-knowledge proof system. On the other hand, it is very unlikely that perfect zero-knowledge proofs for all NP exist, as their complexity-theoretic consequences (the collapse of the polynomial hierarchy, see [6] and [12]) are considered to be false. Consequently, perfect or even statistical zero knowledge proofs (in the latter, an exponentially small statistical difference is allowed between the verifier's view and the simulator's output) are known to exist only for some classes of languages. Specifically, languages with some random self-reducibility property or number-theoretic properties have been showed to have such proofs in the interactive model (see, e.g., [15,13,21,10,3,19]), and languages with some random self-reducibility and instance density properties have such proofs (see, e.g., [8],[9],[18],[3],[1]) in the non-interactive model. Perfect zero-knowledge proofs give an insightful context for studying the zero-knowledge notion as techniques for them only rely on intrinsic properties of the languages. (In contrast, computational zero-knowledge proofs always use properties of cryptographic tools such as encryption and commitment.) Another remarkable property of perfect zero-knowledge proofs is that they are based on no unproven assumptions, and are secure both against an infinitely-powerful prover and against an infinitely powerful verifier. Finally, we note that most applications of zero-knowledge proofs involve constructing or adapting some perfect zero-knowledge proof (a notable example is that of [11] that shows applications of the parallelization of a perfect zero-knowledge proof to identification schemes.)

Our results. In this paper we enlarge the class of languages that are known to have perfect zero-knowledge both in the interactive and in the non-interactive model (although we mainly focus on the latter). Specifically, we consider new languages defined using Boolean circuit composition over statements of membership to languages that already have a perfect zero-knowledge proof, extending previous research efforts in, e.g., [10,9,8]. We note that the natural idea of applying composition techniques in the interactive model of [10] does not apply to the non-interactive model, for at least one subtle and technical reason (roughly speaking, it is not clear how to use the common random string to emulate the prover's messages). Our first result is a technique allowing circuit composition

for a class of languages having non-interactive perfect zero-knowledge proofs and for a significantly larger class of circuits: polynomial-size and polynomial-depth boolean circuits whose gates are (promised) exact threshold gates and can have fan-out greater than 1 (but bounded). Ideas from this result can be used to design a perfect zero-knowledge proof system in the interactive model for monotone circuit composition over random self-reducible languages; this extends previous results [10] only applicable to monotone formulae (namely, circuits with fan-out 1) composition. By combining our first result with reductions among classes of boolean circuits, we obtain a non-interactive perfect zero-knowledge proof system for NC^1 circuit (that is, polynomial-size and logarithmic-depth boolean formulae) composition over a class of languages having such a proof system. This result does not require any restriction on the boolean gates in the formula and significantly improves over previous results, as the only classes of circuits for which a composition over some languages having a non-interactive perfect zero-knowledge proof system was possible were single threshold gates [9] and a certain class of CNF formulae [8]. We note that our techniques also apply to the notion of statistical zero-knowledge. Due to lack of space, formal proofs of several results are omitted.

2 Definitions

NP, Interactive and Non-Interactive Proof Systems. The class NP is often defined in terms of a proof system: on input an instance x to a language L, a prover computes a short witness w certifying that $x \in L$ and sends w to a verifier who can use w to verify in polynomial time that $x \in L$.

Interactive proof systems were originally introduced in the seminal paper [15] and can be seen as an extension of the above proof system for NP, obtained by allowing prover and verifier to be interactive and probabilistic algorithms.

Non-interactive proof systems were originally introduced and studied by [5,4], in the so-called public random string model. In this model all parties have access to a short public random string, and the proof consists of a single message sent by the prover to the verifier. We study and now formally define non-interactive perfect zero-knowledge proof systems; that is, non-interactive proof systems that allow the prover to reveal no information at all to the verifier, other than the fact that the theorem being proved is true.

Definition 1. Let L be a language, let P be a probabilistic Turing machine and let V be a deterministic Turing machine that runs in time polynomial in the length of its second input. We say that (P,V) is a *non-interactive perfect zero-knowledge* (NIPZK) *proof system* for L if there exists a polynomial r such that the following properties hold:

1. *Completeness.* $\forall x \in L$, for all constants $c > 0$, it holds that
$$\mathrm{Prob}\left[\sigma \leftarrow \{0,1\}^{r(|x|)}; Proof \leftarrow P(\sigma, x) : V(\sigma, x, Proof) = 1\right] \geq 1 - |x|^{-c}.$$
2. *Soundness.* $\forall x \notin L$, for all Turing machines P', it holds that
$$\mathrm{Prob}\left[\sigma \leftarrow \{0,1\}^{r(|x|)}; Proof \leftarrow P'(\sigma, x) : V(\sigma, x, Proof) = 1\right] \leq 2^{-|x|}.$$

3. *Perfect Zero-Knowledge.* There exists an efficient algorithm S, called the *Simulator,* such that $\forall x \in L$, the probability distributions $S(x)$ and $View(x)$ are equal, where
$View(x) = \{\sigma \leftarrow \{0,1\}^{r(|x|)}; Proof \leftarrow P(\sigma, x) : (\sigma, Proof)\}$.

If pair (P,V) in the above definition satisfies the completeness and soundness requirements, we also say that it is a non-interactive proof system for L. We also call the random string σ, input to both P and V, the *reference string.*

Circuits Composition and PET circuits. We consider boolean circuits with a single output but with internal gates having fan-out possibly larger than 1. We define the *size* and the *depth* of a boolean circuit ϕ in the standard way (e.g., size is the length of its standard encoding; and depth is the length of the longest path from an input to the output). We make the following wlog assumptions on the structure of a circuit: the circuit can be divided into levels, such that the inputs to the circuits are at level 0, the root is at level $d-1$, if d is the depth of the circuit, and all internal gates at level i receive inputs from outputs of gates at level $i-1$ and return outputs that become inputs of gates at level $i+1$. (To see that the assumption is wlog, note that if the output of a gate jumps 2 levels higher, we can always insert, say, an OR gate, in the intermediate level and let this output to divide into two inputs of the OR gate, and use the output of the OR gate for the gates 2 level higher.)

We fix the standard complete base $B_0 = \{$ AND, OR, NOT $\}$ for boolean circuits. An m-input *boolean circuit ϕ with fan-out k* is a boolean circuit that takes inputs $v_1, \cdots, v_m$ and returns one output $v = \phi(v_1, \ldots, v_m)$, where each gate in the circuit is a gate in B_0 with at most k outputs and 2 inputs (in the case of AND and OR gates) or 1 input (in the case of NOT gates). We will assume that all internal gates are AND or OR gates and that the NOT gates only appear at the input level.

We now define boolean circuit compositions over a language L.

Given a language L, let χ_L denote the indicator function for the language L (i.e., $\chi_L(x) = 1$ iff $x \in L$). Let C be a class of boolean circuits and define $C_{k,m}$ to be the subset of circuits of C with m inputs and fan-out k. Moreover, let $\boldsymbol{x} = (x_1, \ldots, x_m)$ be an m-tuple of binary strings. We define the language

$$Comp(C; L) = \{(\phi; \boldsymbol{x}) \mid \exists k > 0 : \phi \in C_{k,m} \wedge \phi(\chi_L(x_1), \ldots, \chi_L(x_m)) = 1\}.$$

We now define the classes of boolean circuits of interest. Consider the set of gates $M_0 = \{$ AND, OR $\}$. An m-input *monotone circuit ϕ with fan-out k* is a boolean circuit where each gate in the circuit is a gate in M_0 with at most k outputs and 2 inputs. We now consider a special set of gates PET_0 (for Promised Exact Threshold). We call a gate a *PET gate with parameters* $(q; f; l; u_0; u_1, \ldots, u_l)$ if it has q inputs and f outputs, and the following holds. On input q values $v_1, \ldots, v_q$, the PET gate returns on each of its f outputs:

- 0 if exactly u_0 values of $v_1, \ldots, v_q$ are true;
- 1 if u_1 or u_2 or $\cdots$ u_l values are true, where $u_l > u_{l-1} > \cdots u_1 > u_0$ and $l \leq q - u_0$.

We also call the difference $u_1 - u_0$ the *threshold gap* of the PET gate.

A *boolean PET circuit* ϕ *with fan-out* k is a boolean circuit with fan-out k where each gate is either a PET gate with at most k outputs or a NOT gate; moreover, NOT gates only appear at the inputs. A *monotone PET circuit* ϕ is a boolean PET circuit which does not contain NOT gates.

We say that a PET circuit ϕ is *gap-bounded* if for each gate g of ϕ, the fan-out of g is not greater than the threshold gap $u_1 - u_0$ of g.

In the next section we will present perfect zero-knowledge proof systems for the language *Comp*(*C; L*), where *C* is a subclass of boolean PET circuits and *L* is a DOL language (see Definition 2 below).

3 PET Circuits Composition

In Section 3.1 we formally define a class of languages in NIPZK from which we draw our atomic languages. In Section 3.2 we present our non-interactive perfect zero-knowledge proof system for PET circuit composition over languages in this class. Finally, in Section 3.3 we discuss an application of these techniques to the interactive model and present perfect zero-knowledge proof systems for PET circuit composition over random self-reducible languages.

3.1 Double Opening Languages

Informally, class DOL is defined as the class of languages that have a method for a prover to reveal to the verifier a bit b from the input x and a portion of the reference string so that the following holds. If x is in the language, then the prover can open both $b = 0$ and $b = 1$ to the verifier, and both openings reveal no additional knowledge to the verifier. If x is not in the language, then the prover can only open one bit (either $b = 0$ or $b = 1$) to the verifier (apart from exponentially small probability) even if he deviates from its program; furthermore, either opening reveals no additional knowledge to the verifier. We now present a formal definition of DOL languages.

Definition 2. We say that DOL is the class of languages L for which there exists a triple of probabilistic polynomial-time algorithms (OP,OV,OS) and a polynomial r such that

1. *Opening-Completeness.* $\forall x \in L$, for all $b = 0, 1$, it holds that
$$\text{Prob}\left[\rho \leftarrow \{0,1\}^{r(|x|)}; ops \leftarrow OP(\rho, x, b) : OV(\rho, x, ops) = b\right] = 1.$$
2. *Opening-Soundness.* $\forall x \notin L$, it holds that
$$\text{Prob}\left[\rho \leftarrow \{0,1\}^{r(|x|)}; \exists\, (ops_0, ops_1) : OV(\rho, x, ops_b) = b, \text{ for } b = 0, 1\right] \leq 2^{-|x|}.$$
3. *Opening-Zero-Knowledge.* For all $x \in L$, and all $b \in \{0,1\}$, the probability distributions $OS(x, b)$ and $opview(x, b)$ are equal, where $opview(x, b) = \{\rho \leftarrow \{0,1\}^{r(|x|)}; ops \leftarrow OP(\rho, x, b) : (\rho, ops)\}$.

   For all $x \notin L$, there exists $b \in \{0,1\}$ such that

Prob $[\rho \leftarrow \{0,1\}^{r(|x|)}; ops \leftarrow OP(\rho, x, b) : OV(\rho, x, ops) = b] = 1$, bit b is uniformly distributed (over choice of ρ), and the probability distributions $OS(x, b)$ and $opview(x, b)$ are equal, where
$opview(x, b) = \{\rho \leftarrow \{0,1\}^{r(|x|)}; ops \leftarrow OP(\rho, x, b) : (\rho, ops)\}$.

The intuition behind the definition of DOL languages is that of collapsing the properties of non-interactive perfect zero-knowledge proof systems to that of the communication from prover to verifier of a single bit.

We will call (OP,OV) the *opening proof system* for L and the algorithms OP, OV, and OS, the *opening-prover, opening-verifier* and *opening-simulator,* respectively. We will also call string ρ the *reference string for the opening proof system* (OP,OV).

In the sequel we will use opening proof systems for *strings.* This is defined in the standard way; that is, as the repeated independent application of an opening proof system for each bit in the given string. Formally, if w is an m-bit string, OP, on input ρ, x, w, returns *ops,* and we call *ops* the *x-opening of ρ as w* and w the *x-opened value from ρ* (we will omit reference string ρ or input x when they are clearly determined from the context). On input $x, \rho, ops,$ OV returns w (if *ops* is a correct x-opening of ρ as w) or a special failure symbol $\perp$ otherwise.

We can show some facts about DOL languages.

Theorem 1. Every language in DOL is in NIPZK.

Theorem 2. Let QR (resp., QNR) be the languages of quadratic residues (resp. non-residues) modulo Blum integers. Then QR and QNR belong to class DOL.

3.2 A NIPZK Proof System for PET Circuit Compositions

We present our non-interactive perfect zero-knowledge proof systems for compositions of languages in DOL through circuits with promised exact threshold gates. Specifically, we consider gap-bounded fan-out PET circuits (that is, for each gate, its fan-out is not larger than its threshold gap). Our techniques provide non-interactive perfect zero-knowledge proof systems for any polynomial-size and polynomial-depth circuits in this class. Our first result is the following

Theorem 3. Let L be a language in DOL, and let MPET be the class of polynomial-size and polynomial-depth PET monotone circuits with gap-bounded fan-out. Then there exists a non-interactive perfect zero-knowledge proof system for the language *Comp*(MPET, L).

The proof of Theorem 3 can be extended to the case of multiple languages in DOL. Furthermore, it is worth mentioning the following result.

Theorem 4. Let L be a language such that L and $\overline{L}$ are in DOL, and let BPET be the class of polynomial-size and polynomial-depth PET boolean circuits with gap-bounded fan-out. Then there exists a non-interactive perfect zero-knowledge proof system for the language *Comp*(BPET, L).

We now proceed with the proof of Theorem 3.

We start with an informal description of the proof system. Let L be a language in DOL, let $x_1, \ldots, x_m$ be n-bit input strings, and let C be a polynomial-size and polynomial-depth PET monotone circuit with gap-bounded fan-out.

We would like to extend the properties of DOL languages to each subcircuit of C. In other words, for each subcircuit ψ of C whose inputs are statements of membership to L, we would like to achieve the following. Fix an input vector $\boldsymbol{x}$. If ψ is true under $\boldsymbol{x}$ we would like the prover to be able to $(\psi, \boldsymbol{x})$-open a portion ρ of the reference string both as a 0 and as a 1, and both these openings should be efficiently simulatable. Instead, if ψ is false under $\boldsymbol{x}$ then for any string ρ there exists only one bit $b = b(\sigma, \psi)$ such that ρ can be $(\psi, \boldsymbol{x})$-opened as b. Moreover, we require this opening should be efficiently simulatable. Once these properties are achieved, then a high-level description of the proof system is simply given, as follows: the prover picks a sufficiently long string from the reference string and opens the entire circuit as this string to the verifier, who accepts if the opening is correct. The hard part however is to enforce the mentioned properties for each subcircuit. We note that these properties are clearly true for an atomic statement $x \in L$ as they follow from the definition of DOL languages. In order to achieve them for larger subcircuits, we extend the notion of opening to an entire subcircuit. This is done by proceeding recursively over the circuit, and applying a 'combinatorial gadget' at each gate, based on a careful application of polynomial interpolation over finite fields. Specifically, the recursion proceeds as follows.

Consider an m-input f-output circuit ψ and let $\boldsymbol{x} = (x_1, \cdots, x_m)$ be an input sequence. To (ψ, x)-open a string ρ as strings $w_1, \ldots, w_f$ (each of which can be associated with one of the f outputs of ψ), the prover first of all writes ψ as $\psi = g(\psi_1, \ldots, \psi_q)$, where g is a PET gate with parameters $(q; f; l; u_0; u_1, \ldots, u_l)$ and the ψ_i's are subcircuits. We note that some of the ψ_i's can be equal, which happens when a ψ_i has fan-out > 1. Then the prover divides the q subcircuits into two sets: the set of subcircuits which are true under $\boldsymbol{x}$, say, $\{\psi_1, \ldots, \psi_{q'}\}$, and the set of false subcircuits, say, $\{\psi_{q'+1}, \ldots, \psi_q\}$. At this point, by inductive hypothesis, the circuits in the latter set uniquely determine values $z_{q'+1}, \ldots, z_q$, respectively, and the prover computes these values. (Here, we note that if two or more values are associated with different outputs of the same subcircuit, the prover computes these values by a single recursive call on the only subcircuit that returns these outputs.) After computing values $z_{q'+1}, \ldots, z_q$ as described, the prover computes values $z_1, \ldots, z_{q'}$ such that the combinatorial gadget associated with gate g allows to associate $w_1, \ldots, w_f$ with the output of g and values $z_1, \ldots, z_q$, and therefore to open ψ as $w_1, \ldots, w_f$. Then the prover continues recursively opening each ψ_i as z_i, for $i = 1, \ldots, q$. (Here, similarly as before, we note that if two or more values are associated with different outputs of the same subcircuit, the prover makes a single recursive call on this subcircuit.)

It only remains to explain how the 'combinatorial gadget' at each gate is constructed. Given a PET gate with parameters $(q; f; l; u_0; u_1, \ldots, u_l)$, where the number of outputs f is not larger than the threshold gap $tg = u_1 - u_0$,

we consider a finite field $(\mathcal{E}, +, \cdot)$ with more than $|C|$ elements and define a polynomial $p(x) = a_0 + a_1 x + \cdots + a_{k-1}^{k-1}$, for $k = q - u_0$, where all operations are over $\mathcal{E}$. Each input value z_i to gate g is associated with $p(i)$, for $i = 1, \ldots, q$, and each output value w_j is equal to a_j, for $j = 0, \ldots, f - 1$.

We note that from well known facts about interpolation of polynomials, the described gadget has two useful properties.

1. If $k = q - u_0$ inputs to g are false, then the input values z_i's associated with these inputs uniquely determine a_j, for $j = 0, \ldots, f - 1$.
2. If at least u_1 inputs to g are true, then at most $q - u_1 = k - (u_1 - u_0) = k - tg \leq k - f$ inputs are false, and for any values z_i's associated with these inputs, at least f values $a_0, \ldots, a_{f-1}$ can be arbitrarily chosen so that $p(i) = z_i$, for $i = 1, \ldots, q$.

These two properties guarantee that a false subcircuit can be opened as a tuple of values determined from the random string, while a true subcircuit can be opened as any desired tuple of values.

A formal description. By (OP,OV) we denote the opening proof system for L.

We will consider polynomial-size and polynomial-depth PET monotone circuits that have gap-bounded fan-out and have gates with parameters $(q; 2; 1; u_0; u_1)$. Here we set, for simplicity of description, $f = 2$ and $l = 1$, but we note that the extension to larger f, l presents no difficulty. We note that $u_1 - u_0 \geq 2$ since $f = 2$ and the circuits have gap-bounded fan-out.

Finally, by $\mathcal{E}$ we denote the field with at least $d = 2^{\lceil \log(|C|+1) \rceil}$ elements.

We divide the description into two procedures: GET-OPENING and OPEN-AS and then the two algorithms P and V.

Procedure GET-OPENING is run by the prover on input a portion of the reference string and a (true or false) subcircuit ψ with q subcircuits and f outputs, and returns an opening of ψ as values $w_1, \ldots, w_f$. The intuition is that ψ is either false or a subcircuit of a false circuit and therefore the prover uses this procedure to obtain the only values $w_1, \ldots, w_f$ in $\mathcal{E}$, fixed by the reference string, such that he can obtain an opening *ops* for ψ as $w_1, \ldots, w_f$.

To simplify notation and description, in our procedures we omit simple steps as dividing the reference string into different portions, each given as input to the recursive calls of the procedures, with the understanding that each call to a procedure uses an independent and sufficiently long portion of the reference string.

Input to Procedure GET-OPENING**:** (ϕ, ρ), where ϕ is a PET circuit with $f = 2$ outputs and q subcircuits and ρ is a reference string.

Instructions for Procedure GET-OPENING**:**

1. If $\phi = x$ for some $x \in \{0,1\}^n$, then for $c = 0, 1$,
   use OP to compute the unique value w_c and an opening ops_c of x as w_c (if more than one value w_c can be found then w_c is randomly chosen among them);
   return: (w_0, ops_0, w_1, ops_1).
2. Let $\phi = g(\phi_1, \ldots, \phi_q)$ for some PET gate g with parameters $(q, 2, 1, u_0, u_1)$; rewrite subcircuits $\phi_1, \ldots, \phi_q$ as the $q/2$ distinct subcircuits $\psi_1, \ldots, \psi_{q/2}$;

compute which ψ_i's are true and which ψ_i's are false (let, say, $\psi_1, \ldots, \psi_{u_0/2}$ be true and the remaining ones be either true or false)
for $i = u_0/2+1, \ldots, q/2$, run procedure GET-OPENING on subcircuit ψ_i and obtain values $fw_{2i-1}, fw_{2i} \in \mathcal{E}$ and related openings $fops_{2i-1}, fops_{2i}$
let $\boldsymbol{fw} = (fw_{u_0+1}, \ldots, fw_q)$ and $\boldsymbol{fops} = (fops_{u_0+1}, \ldots, fw_q)$;
compute the unique polynomial $p(x) = a_0 + a_1 x + \cdots + a_{e-1}x^{e-1}$ such that $e = q - u_0$, and $p(i) = fw_i$, for $i = u_0 + 1, \ldots, q$;
for each $i = 1, \ldots, u_0/2$, run procedure OPEN-AS (described below) on the true subcircuit ψ_i to compute an opening as $w_{2i-1} = p(2i-1)$ and $w_{2i} = p(2i)$ (as each of them has 2 outputs), and let ***tops*** be the sequence of openings obtained;
return values a_0, a_1 and openings (***tops***, ***fops***).

Procedure OPEN-AS is run by the prover on input a portion of the reference string, desired values $w_0, \ldots, w_{f-1}$, and a true subcircuit ψ with q subcircuits and f outputs, and returns an opening of ψ as values $w_1, \ldots, w_f$. The intuition is that ψ is true and therefore the prover uses this procedure to obtain an opening of ψ as desired values $w_0, \ldots, w_{f-1}$ in E. As before, we will omit simple steps as dividing the reference string into different portion, and we assume $f = 2$.

Input to Procedure OPEN-AS**:** (ϕ, ρ, w_0, w_1), where ϕ is a PET circuit with $f = 2$ outputs and q subcircuits, ρ is a reference string and $w_0, w_1 \in \mathcal{E}$.

Instructions for Procedure OPEN-AS**:**

1. If $\phi = x$ for some $x \in \{0,1\}^n$, then for $c = 0, 1$,
   use OP to compute an opening ops_c of x as value w_c;
   return: (w_0, ops_0, w_1, ops_1).
2. Let $\phi = g(\phi_1, \ldots, \phi_q)$ for some PET gate g with parameters $(q, 2, 1, u_0, u_1)$;
   rewrite subcircuits $\phi_1, \ldots, \phi_q$ as the $q/2$ distinct subcircuits $\psi_1, \ldots, \psi_{q/2}$;
   compute which ψ_i's are true and which ψ_i's are false, and let, say, $\psi_1, \ldots, \psi_{u_0/2+f/2}$ be true (and the remaining ones be either true or false)
   for $i = u_0/2 + f/2 + 1, \ldots, q/2$, run procedure GET-OPENING on subcircuit ψ_i and obtain values $fw_{2i-1}, fw_{2i} \in \mathcal{E}$ and related openings $fops_{2i-1}, fops_{2i}$
   let $\boldsymbol{fw} = (fw_{u_1+1}, \ldots, fw_q)$ and $\boldsymbol{fops} = (fops_{u_1+1}, \ldots, fw_q)$;
   compute the unique polynomial $p(x) = a_0 + a_1 x + \cdots + a_{e-1}x^{e-1}$ such that $e = q - u_0$; $a_j = w_j$, for $j = 0, 1$ and $p(i) = fw_i$, for $i = u_0 + f + 1, \ldots, q$;
   for each $i = 1, \ldots, u_0/2 + f/2$, run procedure OPEN-AS on the true subcircuit ψ_i to open it as $p(2i-1)$ and $p(2i)$ (as each of them has 2 outputs), and let ***tops*** be the sequence of openings thus obtained;
   return opening (***tops, fops***).

The prover's algorithm P is run on input the reference string, a circuit C with q subcircuits, two outputs, and a tuple of parameters for each of its gates, and n-bit input strings $x_1, \ldots, x_m$. Given procedures OPEN-AS and GET-OPENING, the description of P's algorithm is extremely simplified as it is enough to run procedure OPEN-AS on input C and random values $w_0, w_1 \in \mathcal{E}$ from the reference string.

Input to prover P: $(\sigma, C, \boldsymbol{x})$, where C is a polynomial-size and polynomial-depth PET monotone circuit that has gap-bounded fan-out and has gates with

parameters $(q; 2; 1; u_0; u_1)$; σ is a reference string, and $\boldsymbol{x} = (x_1, \ldots, x_m)$ is a sequence of the n-bit strings x_i inputs to C.

Instructions for prover P:

1. Write σ as $w_0 \circ w_1 \circ \rho$, where $w_0, w_1 \in \mathcal{E}$.
2. Run OPEN-AS on input C, ρ, w_0, w_1 to compute a C-opening *ops* of C as values w_0, w_1
3. Send *ops* to V.

The verifier's algorithm V is run on input the reference string, a circuit C with q subcircuits, two outputs, and a tuple of parameters for each of its gates; n-bit strings $x_1, \ldots, x_m$, and a message *ops* sent by P. The description of V's algorithm is not hard as it verifies that the opening *ops* sent by P is valid and that the values opened by *ops* define valid openings of all subcircuits. Specifically, V performs computation on the circuit at each level starting from the bottom level (that is, the inputs to the circuit) to the top level (that is, the outputs of the circuit). At each level of C, and for each gate, given opened values for its inputs, V uses polynomial interpolation to compute opened values for the gate's outputs. At the end V can check if the two opened values for the output of C are the values w_0, w_1 in the reference string.

Input to verifier V: $(\sigma, C, \boldsymbol{x})$, and a message *ops* from P.

Instructions for verifier V:

1. Write σ as $w_0 \circ w_1 \circ \rho$, where $w_0, w_1 \in \mathcal{E}$.
2. Write *ops* as $ops_1, \ldots, ops_m$, where ops_i is an x_i-opening as some value v_i.
3. For each level $i = leaf, .., root$ of C,
   for each gate g at level i of C,
   let g have parameters $(q, 2, 1, u_0, u_1)$
   let $v_1, \ldots, v_q$ be the opened values associated with the inputs to g
   verify that there exists a polynomial $p(x) = a_0 + a_1 x + \cdots + a_{e-1} x^{e-1}$
   such that $e = q - u_0$ and $p(i) = v_i$, for $i = 1, \ldots, q$
   associate a_0, a_1 with the two outputs of root g
4. Verify that the output values at the root gate a_0, a_1 are equal to w_0, w_1.
5. If all verifications are satisfied then accept, otherwise reject.

To prove the completeness property of (P,V), we show that, on input circuit ϕ, the following holds: if ϕ is true then procedure OPEN-AS, on input ϕ and values $a_0, a_1 \in \mathcal{E}$, returns an opening *ops* for ϕ as a_0, a_1; furthermore, for any ϕ, procedure GET-OPENING always returns values $a_0, a_1 \in \mathcal{E}$ and opening *ops* for ϕ as a_0, a_1

To prove the soundness property of (P,V), we show that, on input a false circuit ϕ with m inputs $x_1, \ldots, x_m$ such that $|x_i| = n$ for $i = 1, \ldots, m$, the probability that for more than one value $w \in \mathcal{E}$ there exists a $(\phi, \boldsymbol{x})$-opening as w is at most $(m \cdot |\mathcal{E}|)/2^{-n}$.

To prove the perfect zero-knowledge property of (P,V), we recursively use polynomial interpolation to show properties about the distributions of the opening returned by procedures OPEN-AS and GET-OPENING.

3.3 A PZK Proof System for PET Circuits Compositions

The techniques presented in Section 3.2 for circuit composition over languages in class DOL in the non-interactive model can be used for circuit composition over random self-reducible (RSR) languages (in fact, a larger class) in the interactive model. Specifically, we obtain the following

Theorem 5. Let L be a language in RSR, and let MPET be the class of polynomial-size and polynomial-depth PET monotone circuits with gap-bounded fan-out. Then there exists a perfect zero-knowledge proof system for the languages *Comp*(MPET, L) and *Comp*(MPET, $\overline{L}$).

We note that a previous result in [10] showed the analogue result for polynomial-size and polynomial-depth monotone circuits with fan-out 1. Theorem 5 enlarges the class of circuits for which composition of PZK is possible; namely, PET circuits with fan-out > 1.

4 NC1 Boolean Circuits Composition

We show two reductions between boolean circuits that allow to obtain interesting consequences on NIPZK given the results in Section 3. Our reductions convert any NC^1 boolean circuit into a polynomial-size PET circuit with fan-out 1 or 2, respectively (the second resulting in a PET circuit with smaller size). We say that two circuits C_1, C_2 are *equivalent* if they have the same number of inputs and $C_1(x) = C_2(x)$ for any x. Formally, we obtain the following

Theorem 6. For any NC^1 circuit C having depth at most d,

1. there exists an equivalent PET boolean circuit C' with fan-out 1, having size $\Theta(3^d)$
2. there exists an equivalent PET boolean circuit C'' with gap-bounded fan-out 2, having size $\Theta(2^d)$

We note that previous results in [20] might be used to derive a reduction from any NC^1 boolean circuit to a polynomial-size PET circuit where the resulting circuit has size $\Theta(8^d)$. By combining Theorem 4 and Theorem 6, we obtain the following

Theorem 7. If language L and $\overline{L}$ are in DOL and C is an NC^1 circuit, then there exists a non-interactive perfect zero-knowledge proof system for Comp(C, L).

By inspection of the proof of Theorem 4, we also obtain the following

Theorem 8. If language L is in DOL and C is a monotone NC^1 circuit, then language Comp(C, L) is in DOL.

Theorem 9. If language L and $\overline{L}$ are in DOL and C is an NC^1 circuit, then language Comp(C, L) is in DOL.

References

1. M. Bellare, and M. Yung, *Certifying Cryptographic Tools: the case of Trapdoor Permutations,* in Journal of Cryptology.
2. M. Ben-Or, O. Goldreich, S. Goldwasser, J. Hastad, S. Micali, and P. Rogaway, *Everything Provable is Provable in Zero Knowledge,* in Proc. of CRYPTO 88.
3. J. Boyar, K. Friedl, and C. Lund, *Practical Zero-Knowledge Proofs: Giving Hints and Using Deficiencies,* Journal of Cryptology, n. 4, pp. 185–206, 1991.
4. M. Blum, A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge,* SIAM Journal of Computing, vol. 20, no. 6, Dec 1991, pp. 1084–1118.
5. M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and Applications,* Proc. of STOC 88.
6. R. Boppana, J. Hastad, and S. Zachos, *Does co-NP has Short Interactive Proofs ?,* Inf. Proc. Lett., vol. 25, May 1987, pp. 127–132.
7. J. Boyar and R. Peralta, *Short Discreet Proofs,* Proc. of EUROCRYPT 96.
8. A. De Santis, G. Di Crescenzo, and G. Persiano, *Secret Sharing and Perfect Zero-Knowledge,* in Proc. of CRYPTO 93.
9. A. De Santis, G. Di Crescenzo, and G. Persiano, *The Knowledge Complexity of Quadratic Residuosity Languages,* in Theoretical Computer Science, Vol. 132, pp. 291–317 (1994).
10. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung, *On Monotone Formula Closure of SZK,* Proc. of FOCS 94.
11. U. Feige, A. Fiat and A. Shamir, *Zero-Knowledge Proofs of Identity,* in Journal of Cryptology.
12. L. Fortnow, *The Complexity of Perfect Zero-Knowledge,* in Proc. of STOC 87, 1987.
13. O. Goldreich, S. Micali, and A. Wigderson, *Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems,* Journal of the ACM, vol. 38, n. 1, 1991, pp. 691–729.
14. O. Goldreich and Y. Oren, *Definitions and Properties of Zero-Knowledge Proof Systems,* Journal of Cryptology, vol. 7, 1994, pp. 1–32.
15. S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems,* SIAM J. on Computing, vol. 18, n. 1, 1989.
16. R. Impagliazzo and M. Yung, *Direct Minimum Knowledge Computations* in Proc. of CRYPTO 87.
17. T. Itoh, Y. Ohta, and H. Shizuya, *A Language-Dependent Cryptographic Primitive,* in Journal of Cryptology, 10(1): 37-50 (1997)
18. R. Gennaro, D. Micciancio, and T. Rabin, *An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products,* Proc. of 5th ACM Conference on Computer and Communication Security - CCS 1998.
19. D. Micciancio and S. Vadhan, *Statistical zero-knowledge proofs with efficient provers: lattice problems and more,* in Proc. of CRYPTO 2003.
20. T. Sander, A. Young and M. Yung, *Non-Interactive Cryptocomputing for NC1,* in Proc. of FOCS 99.
21. M. Tompa and H. Woll, *Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information,* in Proc. of FOCS 87.

All Superlinear Inverse Schemes Are coNP-Hard*

Edith Hemaspaandra[1], Lane A. Hemaspaandra[2], and Harald Hempel[3]**

[1] Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623-5608, USA.
eh@cs.rit.edu

[2] Department of Computer Science, University of Rochester, Rochester, NY 14627-0226, USA.
lane@cs.rochester.edu

[3] Institut für Informatik, Friedrich-Schiller-Universität Jena, D-07743 Jena, Germany.
hempel@informatik.uni-jena.de

Abstract. How hard is it to invert NP-problems? We show that all superlinearly certified inverses of NP problems are coNP-hard. As part of our work we develop a novel proof technique that builds diagonalizations against certificates directly into a circuit.

1 Introduction

In this paper we show that all superlinear inverse schemes of NP problems are coNP-hard. We develop a novel proof technique that allows us to diagonalize against all possible certificate sets. We feel that this "in-circuit diagonalization" proof technique is of interest in its own right.

The class NP can be viewed as the set of all languages L such that there exist a polynomial-time computable verifier V and a polynomial q such that, for all $x \in \Sigma^*$, $x \in L \iff (\exists y \in \Sigma^*)[|y| = q(|x|) \wedge V(x,y)$ accepts]. A string y such that $V(x,y)$ accepts is called a certificate or proof for x. Verifiers can formally be defined as follows (see Definition 3):

A pair (V,q) is called a standard verifier if and only if
1. $V : \Sigma^* \times \Sigma^* \to \{0,1\}$ is polynomial-time computable mapping, i.e., $V \in \mathrm{FP}$ and
2. q is a strictly monotonic, integer-coefficient polynomial such that

$$(\forall x, y \in \Sigma^*)[V(x,y) = 1 \implies |y| = q(|x|)].$$

Inverting standard verification schemes can now informally be described as follows: Let (V,q) be a standard verifier. Given a set of certificates C, does there

* Supported in part by grants NSF-CCR-9322513, NSF-INT-9815095/DAAD-315-PPP-gü-ab, and NSF-CCR-0311021.

** Work done while visiting the University of Rochester.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 368–379, 2004.
© Springer-Verlag Berlin Heidelberg 2004

exist a string x such that C is exactly the set of certificates for x (relative to (V, q))? It is quite natural to choose a succinct representation of certificates in form of a circuit. This leads to the following definition (see Definition 4) of the inverse problem, which basically asks if a set of strings specified by a circuit is such that some string has precisely those strings as its certificate set.

Let (V, q) be a standard verifier.
$\mathbf{Invs}_{V,q} = \{c \mid c$ encodes a circuit c' having $q(m)$ inputs for some $m \in \mathbb{N}$ such that $(\exists x \in \Sigma^m)[\{w \in \Sigma^{q(m)} \mid V(x, w) = 1\} = \{y \in \Sigma^{q(m)} \mid c'(y) = 1\}]\}$.

We show that inversion for all superlinear standard verification schemes is coNP-hard. In fact we show even more, namely, that inverting any standard verification scheme (V, q) where q grows faster than all outright linear functions $n+k$, $k \in \mathbb{N}$, is coNP-hard (see Theorem 2). So coNP-hardness in fact holds for all $\mathbf{Invs}_{V,q}$ where (V, q) is a standard verification scheme and q is a polynomial of degree either greater than one or of degree one with a degree-one-coefficient $a_1 > 1$.

The proof of our main result is based on a proof technique that can informally be described as an "in-circuit diagonalization" against possible certificate sets. In particular, our in-circuit diagonalization technique uses a circuit *to diagonalize against certificate sets that are potentially accepted by the very same circuit.* The need to diagonalize in such an unusual way arises from the fact that when reducing $\overline{\mathrm{SAT}}$ to $\mathbf{Invs}_{V,q}$ (as we will do in the proof of Theorem 2) one has to map boolean formulas to circuits such that the following holds: If the formula is satisfiable then, for all x, the set of strings accepted by the circuit is not equal to the set of certificates for x (relative to (V, q)). If the formula is not satisfiable then there exists a string x such that the the set of strings accepted by the circuit is exactly the set of certificates for x (relative to (V, q)).

Relatedly, Σ_2^p is clearly an upper bound for the complexity of inverting standard verification schemes and we prove that this upper bound is optimal by constructing a standard verifier such that its inversion problem is Σ_2^p-complete (see Theorem 6). We also show that for every NP set A there is a standard verifier (V, q) such that $\mathbf{Invs}_{V,q}$ is at least as hard as A, $A \leq_m^p \mathbf{Invs}_{V,q}$. As an interesting corollary we obtain that there exists a P set A and a standard verifier (V, q) for A such that $\mathbf{Invs}_{V,q}$ is NP-hard.

Our results can be extended to also hold for the one-sided variant of inversion of verification schemes, $\text{1-}\mathbf{Invs}_{V,q}$. The difference in the definitions of $\mathbf{Invs}_{V,q}$ and $\text{1-}\mathbf{Invs}_{V,q}$ (see Definition 4) is that instead of requiring "$\exists x \in \Sigma^*$ such that the set of strings accepted by the circuit equals the set of certificates of x" as in definition of $\mathbf{Invs}_{V,q}$, we instead require "$\exists x \in L(V, q)$ such that the set of strings accepted by the circuit equals the set of certificates of x" in the definition of $\text{1-}\mathbf{Invs}_{V,q}$.

In a fascinating paper by Chen [3], a type of inversion of NP problems is studied that is somewhat related to the above-described one-sided-inversion problem, $\text{1-}\mathbf{Invs}_{V,q}$, and Σ_2^p results are obtained. However, the models are different; for example, in contrast to our definition, where certificates are given in a very succinct form, i.e., implicitly in form of a circuit, Chen studied one-sided inversions of

NP problems where the certificates are explicitly given, i.e., in form of a set or a list, and also Chen's focus is on the one-sided inversion problem.

Our paper is organized as follows. After formally defining the basic concepts (Section 2), in Section 3 we state and prove our main result—that all superlinearly certified inverses are coNP-hard. In Section 3 we also prove a number of related theorems, in particular the optimality of the Σ_2^p upper complexity bound for $\mathrm{Invs}_{V,q}$. In Section 4, we turn briefly to the complexity of recognizing whether machines compute (general) verifiers and prove a Σ_2^0-completeness result.

2 Preliminaries

We assume the reader to be familiar with the basic definitions and concepts of complexity theory (see [8,4]). Let $\Sigma = \{0,1\}$ be our alphabet. We say a polynomial p is strictly monotonic (by which we always mean strictly monotonically increasing) if, for all $n \in \mathbb{N}$, $p(n+1) > p(n)$. For any set A and any $m \in \mathbb{N}$, $A^{=m}$ denotes $\{z \mid z \in A \wedge |z| = m\}$.

Without defining it formally we will make use of a nice (i.e., polynomial-time computable and polynomial-time invertible) encoding of any boolean circuit (consisting of AND, OR and NOT gates) as a word over the alphabet Σ. As is standard, we denote the outcome (0 or 1, representing reject/false and accept/true) of a circuit c on input x by $c(x)$.

Let FP denote the set of all polynomial-time computable functions, where these functions without loss of generality can be of any fixed arity. We will use the following standard complexity classes.

Definition 1. *1.* P *is the set of all languages that can be accepted in deterministic polynomial time.*

2. NP *is the set of all languages that can be accepted in nondeterministic polynomial time.* coNP *is defined to be the set of all languages* $\overline{A}$ *such that* $A \in$ NP.
3. *[9]* DP *is the set of all languages* L *such that there exist* NP *sets* A *and* B *satisfying* $L = A - B$.
4. *[7,11]* Σ_2^p *is the set of all languages that can be accepted by nondeterministic polynomial-time Turing machines with the help of an* NP *oracle;* $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$. *Let* PH *denote the polynomial hierarchy,* $\mathrm{PH} = \mathrm{P} \cup \mathrm{NP} \cup \mathrm{NP}^{\mathrm{NP}} \cup \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}} \cup \cdots$.

We mention in passing that P, NP, and DP are the low levels of the boolean hierarchy [1,2] and that P, NP, and Σ_2^p are the low levels of the polynomial hierarchy [7,11].

Let REC denote the set of all recursive languages. The second level of the arithmetic hierarchy Σ_2^0 is defined as follows.

Definition 2. *(see [10]) A language* L *is in* Σ_2^0 *if and only if there exists a language* $B \in$ REC *such that for all* $x \in \Sigma^*$,

$$x \in L \iff (\exists y \in \Sigma^*)(\forall z \in \Sigma^*)[\langle x, y, z\rangle \in B],$$

where $\langle \cdot, \cdot, \cdot \rangle$ *here is a standard, nice 3-ary pairing function.*

As is standard we will use $\leq_m$ (respectively, $\leq_m^p$) to denote recursive many-one reductions (respectively, polynomial-time many-one reductions) between languages.

In the following we will define the basic concepts that allow us to study inverse NP problems.

Definition 3. *1. A pair* (R, p) *is called a* general verifier *if and only if*

a) $R : \Sigma^* \times \Sigma^* \to \{0,1\}$ *is polynomial-time computable mapping, i.e.,* $R \in \mathrm{FP}$ *and*

b) $p : \mathbb{N} \to \mathbb{N}$ *is a strictly monotonic polynomial such that*

$$(\forall x, y \in \Sigma^*)[R(x,y) = 1 \implies p(|y|) \geq |x|].$$

2. A pair (V, q) *is called a* standard verifier *if and only if*

a) $V : \Sigma^* \times \Sigma^* \to \{0,1\}$ *is polynomial-time computable mapping, i.e.,* $V \in \mathrm{FP}$ *and*

b) $q : \mathbb{N} \to \mathbb{N}$ *is a strictly monotonic, integer-coefficient polynomial such that*

$$(\forall x, y \in \Sigma^*)[V(x,y) = 1 \implies |y| = q(|x|)].$$

3. We say that (V, q) *is a standard verifier for a language* L *if and only if* (V, q) *is a standard verifier and* $L = L(V, q)$, *where* $L(V, q) = \{x \in \Sigma^* \mid (\exists y \in \Sigma^*)[|y| = q(|x|) \wedge V(x,y) = 1]\}$.

4. We say a 2-ary Turing machine M computes a general (standard) verifier *if there is a polynomial* p *such that* $(\chi_{L(M)}, p)$ *is a general (standard) verifier. (Note that, regarding types,* $L(M) \subseteq \Sigma^* \times \Sigma^*$, *and* $\chi_{L(M)}$*—the characteristic function—maps from* $\Sigma^* \times \Sigma^*$ *to* $\{0, 1\}$.*)*

The following two facts are immediate and standard.

Fact 1.

1. *For every set* $A \in \mathrm{NP}$ *there exists a standard verifier* (V, q) *such that* (V, q) *is a standard verifier for* A.
2. *If* (V, q) *is a standard verifier for a language* L *then* $L \in \mathrm{NP}$.

We now define the inverse problem for NP languages.

Definition 4. *Let* $A \in \mathrm{NP}$ *and let* (V, q) *be a standard verifier for* A.

1. $\mathrm{Invs}_{V,q} = \{c \mid c$ *encodes a circuit* c' *having* $q(m)$ *inputs for some* $m \in \mathbb{N}$ *such that* $(\exists x \in \Sigma^m)[\{w \in \Sigma^{q(m)} \mid V(x,w) = 1\} = \{y \in \Sigma^{q(m)} \mid c'(y) = 1\}]\}$.
2. *We define* $1\text{-}\mathrm{Invs}_{V,q} = \{c \mid c$ *encodes a circuit* c' *having* $q(m)$ *inputs for some* $m \in \mathbb{N}$ *such that* $(\exists x \in A^{=m})[\{w \in \Sigma^{q(m)} \mid V(x,w) = 1\} = \{y \in \Sigma^{q(m)} \mid c'(y) = 1\}]\}$.

It is not hard to see that for standard verifiers (V, q), $\mathrm{Invs}_{V,q}$ and $1\text{-}\mathrm{Invs}_{V,q}$ are both always in Σ_2^p. However $\mathrm{Invs}_{V,q}$ and $1\text{-}\mathrm{Invs}_{V,q}$ seem to differ with respect to their complexity lower bounds.

Proposition 1. *There is a set* $A \in \mathrm{NP}$ *such that for all standard verifiers* (V, q) *for A,* $1\text{-Invs}_{V,q} \in \mathrm{P}$.

One proof is by simply choosing A to be $\emptyset$ or any other finite set. In contrast, for every standard verifier (V, q) for $\emptyset$ we have that $\mathrm{Invs}_{V,q}$ is $\leq_m^p$-complete for coNP.

Proposition 2. *Let* (V, q) *be a standard verifier for* $\emptyset$. *Then* $\mathrm{Invs}_{V,q}$ *is* $\leq_m^p$-*complete for* coNP.

The claim follows from the fact that from (V, q) being a standard verifier for $\emptyset$ the set $\mathrm{Invs}_{V,q}$ is essentially the set of all appropriate-number-of-inputs circuits that for no input evaluate to 1, and is easily seen to be in coNP. Also, it is straightforward to reduce the coNP-complete language $\overline{\mathrm{SAT}}$ to $\mathrm{Invs}_{V,q}$.

3 Inverting NP Problems Is coNP-Complete

Before stating our main theorem we need a technical definition.

Definition 5. *A polynomial* q *is called* miserly *if and only if for all* $\epsilon > 0$ *there exist infinitely many* $n \in \mathbb{N}$ *such that* $q(n) \leq (1 + \epsilon)n$.

Note that for strictly monotonic polynomials p, $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$, with $a_k > 0$, we have that p is nonmiserly if and only if either a) $k \geq 2$ or b) $k = 1$ and $a_1 > 1$.

Theorem 2. *Let* $A \in \mathrm{NP}$ *and* (V, q) *be a standard verifier for A such that* q *is a nonmiserly polynomial. Then* $\mathrm{Invs}_{V,q}$ *is* $\leq_m^p$-*hard for* coNP.

This immediately yields the following, where by "nonmiserly standard verifier" we mean a standard verifier whose second component is a nonmiserly polynomial.

Corollary 1. *No nonmiserly standard verifier for an* NP *set has an inverse problem belonging to* NP, *unless* NP = coNP.

Proof of Theorem 2: Let $A \in \mathrm{NP}$ and let (V, q) be a standard verifier for A. Suppose that q is nonmiserly. We will show that $\overline{\mathrm{SAT}} \leq_m^p \mathrm{Invs}_{V,q}$.

Let F be a formula and suppose that F has n variables. Our reduction g will map F to the encoding $c = g(F)$ of a circuit c'. The circuit c' will have $q(n')$ inputs where n' is the smallest natural number such that $q(n') > n + n'$. Note that since q is nonmiserly n' is linearly related to n and can be found in polynomial time. On input $z \in \{0,1\}^{q(n')}$, let x, α, and r be the unique strings such that $z = x\alpha r$, $x \in \{0,1\}^{n'}$, $\alpha \in \{0,1\}^{n}$, and $r \in \{0,1\}^{q(n')-n'-n}$. The circuit c' consists of three subcircuits that work as follows:

Subcircuit 1: Subcircuit 1 simulates the work of $V(x, z)$. Let $a = V(x, z)$ be the output of subcircuit 1.

Subcircuit 2: Subcircuit 2 is a polynomial-size-bounded circuit for F with α as its input. Let $b = F(\alpha)$ be the output of subcircuit 2.

Subcircuit 3: Subcircuit 3 simulates the work of $V(0^{n'}, z)$. Let $d = V(0^{n'}, z)$ be the output of subcircuit 3.

Output of c': c' outputs 0 if $b = d = 0$ or $a = b = 1$. c' outputs 1 otherwise, that is if either a) $b = 0$ and $d = 1$ or b) $b = 1$ and $a = 0$.

It is obvious that c' and thus also c can be constructed in time polynomial in $|F|$.

It remains to show that for all formulas F, $F \in \overline{\text{SAT}} \iff g(F) \in \text{Invs}_{v,q}$. Suppose that $F \in \overline{\text{SAT}}$. So we have for all inputs z to the circuit c', $b = 0$. Thus, for all inputs z, $c'(z) = 1$ if and only if $d = 1$. By construction $d = 1$ if and only if $V(0^{n'}, z) = 1$. It follows that $\{z \in \Sigma^{q(n')} \mid c'(z) = 1\} = \{y \in \Sigma^{q(n')} \mid V(0^{n'}, y) = 1\}$ and so (via the certificates of $0^{n'}$) $c = g(F) \in \text{Invs}_{V,q}$. For the other direction of the equivalence to be shown assume $F \notin \overline{\text{SAT}}$. So there exists an n-bit assignment $\widehat{\alpha}$ for F such that $F(\widehat{\alpha}) = 1$ and consequently for all inputs z to the circuit c' such that $z = x\widehat{\alpha}r$ with $x \in \{0,1\}^{n'}$ and $r \in \{0,1\}^{q(n')-n'-n}$, we have $b = 1$. It follows that for all $x \in \{0,1\}^{n'}$ there exists some input z' to the circuit, namely $z' = x\widehat{\alpha}0^{q(n')-n'-n}$, such that (a) $c'(z') = 0$ if $a = V(x, z') = 1$ and (b) $c'(z') = 1$ if $a = V(x, z') = 0$. It follows that for all $x \in \{0,1\}^{n'}$, $\{z \in \Sigma^{q(n')} \mid c'(z) = 1\} \neq \{y \in \Sigma^{q(n')} \mid V(x, y) = 1\}$, and so $g(F) \notin \text{Invs}_{V,q}$. □

Since by our remark preceding Theorem 2 any superlinear polynomial is nonmiserly, we have the following corollary.

Corollary 2. *Let $A \in$ NP and let (V, q) be a standard verifier for A such that q is a superlinear polynomial. Then* $\text{Invs}_{V,q}$ *is $\leq_m^p$-hard for* coNP.

Before we can state a similar result for $1\text{-Invs}_{V,q}$ we need a technical concept that (though as far as we know new) we feel is also a very natural concept.

Definition 6. *We say a set A is* P-producible *if and only if there exists a function $h \in$ FP such that for all $x \in \Sigma^*$, $|h(x)| \geq |x|$ and $h(x) \in A$.*

Our definition of P-producibility should be contrasted (especially as to what the polynomial time is in relation to—the input or the output) with the notion of tangibility introduced by Hemachandra and Rudich [5].[1]

Theorem 3. *Let A be any* NP *set that is* P*-producible. Let (V, q) be a standard verifier for A such that q is a nonmiserly polynomial. Then* $1\text{-Invs}_{V,q}$ *is $\leq_m^p$-hard for* coNP.

Proof. Let A be an NP set that is P-producible via a function $h \in$ FP. Let (V, q) be a standard verifier for A such that q is a nonmiserly polynomial.

[1] A set A is called tangible if and only if there exists a total function f that can be computed in time polynomial in the size of its output such that for all $x \in \Sigma^*$, $f(x) \in A$ and $f(x) \geq_{\text{lexicographical}} x$.

The proof proceeds quite similarly to the proof of Theorem 2. Let F be a formula with n variables. Let n' be the smallest natural number such that $q(n') > n+n'$. The difference from the proof of Theorem 2 is that the constructed circuit c' has to be modified as follows: Let $w = h(0^{n'+1})$. c' will have $q(|w|)$ inputs. On input $z \in \{0,1\}^{q(|w|)}$, let $z = x\alpha r$ where $x \in \{0,1\}^{|w|}$, $\alpha \in \{0,1\}^n$, and $r \in \{0,1\}^{q(|w|)-|w|-n}$, the circuit works as follows (note the natural adjustment in Subcircuit 3).

Subcircuit 1: Subcircuit 1 simulates the work of $V(x,z)$. Let $a = V(x,z)$ be the output of subcircuit 1.

Subcircuit 2: Subcircuit 2 is a polynomial-size-bounded circuit for F and uses α as its input. Let $b = F(\alpha)$ be the output of subcircuit 2.

Subcircuit 3: Subcircuit 3 simulates the work of $V(w,z)$. Let $d = V(w,z)$ be the output of subcircuit 3.

Output of c': c' outputs 0 if $b = d = 0$ or $a = b = 1$. c' outputs 1 otherwise, that is if either a) $b = 0$ and $d = 1$ or b) $b = 1$ and $a = 0$.

The correctness of the reduction can be shown as in the proof of Theorem 2, where w now plays the role that $0^{n'}$ played in the proof of Theorem 2. □

In the reminder of this section we would like to derive some NP-hardness and Σ_2^p-completeness results. First we show that there exist standard verifiers (V,q) such that $\text{Invs}_{V,q}$ is NP-hard.

Theorem 4. *For every set $A \in \text{NP}$ there exists a standard verifier (V,q) such that $A \leq_m^p \text{Invs}_{V,q}$.*

Proof. Let $A \in \text{NP}$ and let (R,p) be a standard verifier for A. Recall that p thus must be strictly monotonic, and can be taken to be polynomial-time computable. Let $q(n) = n$ for all n and define a standard verifier (V,q) as follows:

V accepts on input (a,b) if and only if
1. $|a| = |b|$ and
2. $(\exists x \in \Sigma^*)(\exists y \in \Sigma^* : |y| = p(|x|))[a = xy \wedge b = 0^{p(|x|)}x \wedge R(x,y) = 1]$.

It is not hard to see that V is polynomial-time computable and that (V,q) is a standard verifier. It remains to show that $A \leq_m^p \text{Invs}_{V,q}$. The reduction will map any string $x \in \Sigma^*$ to the encoding c_x of a circuit with $|x| + p(|x|)$ inputs that accepts just one string, namely, the string $0^{p(|x|)}x$. It is obvious that this is a polynomial-time reduction. It remains to show that for all $x \in \Sigma^*$, $x \in A \iff c_x \in \text{Invs}_{V,q}$.

Suppose that $x \in A$. Since (R,p) is a standard verifier for A there exists a string $y \in \Sigma^*$, $|y| = p(|x|)$, such that $R(x,y) = 1$. Hence there exists a string y, $|y| = p(|x|)$ such that $V(xy, 0^{p(|x|)}x) = 1$. Since $0^{p(|x|)}x$ is the only certificate for xy relative to (V,q) it follows that $c_x \in \text{Invs}_{V,q}$. For the other implication of the equivalence to be shown assume $x \notin A$. It follows that there exists no string $y \in \Sigma^*$, $|y| = p(|x|)$, such that $R(x,y) = 1$ and hence there exists no string y, $|y| = p(|x|)$, such that $V(xy, 0^{p(|x|)}x) = 1$. It follows that there exists no string w,

$|w| = |x| + p(|x|)$, such that $V(w, 0^{p(|x|)}x) = 1$, and thus $c_x \notin \text{Invs}_{V,q}$ (note that since p is strictly monotonic the only decomposition of w that has any chance of working is into an $|x|$-bit part, and note also that the definition of V lets $0^{p(|x|)}x$ force our focus onto x itself). □

In fact, it is not hard to see that the just-given proof also establishes the following one-sided theorem (note in particular how when $x \in A$ our c_x does have at least one certificate—in fact, exactly one, but that is not critical).

Corollary (to the proof) 5. *For every set $A \in \text{NP}$ there exists a standard verifier (V, q) such that $A \leq_m^p$ 1-$\text{Invs}_{V,q}$.*

It is interesting to note that the verifier (V, q) constructed in the proof of Theorem 4 is a verifier for a language from P. In particular, (V, q) is a verifier for the P language $\{w \in \Sigma^* \mid (\exists x \in \Sigma^*)(\exists y \in \Sigma^* : |y| = p(|x|))[w = xy \wedge R(x, y) = 1]\}$. However, when choosing (R, p) to be a standard verifier for an NP-complete language A, for instance $A = \text{SAT}$ (which clearly does have standard verifiers, of course), we obtain the following corollary (we do not claim that $B = A$; rather, B can be the P language just discussed).

Corollary 3. *1. There exists a language $B \in \text{P}$ and a standard verifier (V, q) for B such that $\text{Invs}_{V,q}$ is NP-hard.*

2. There exists a language $B \in \text{P}$ and a standard verifier (V, q) for B such that 1-$\text{Invs}_{V,q}$ is NP-hard.

As already mentioned in Section 2, we have seen that $\text{Invs}_{V,q} \in \Sigma_2^p$ for all standard verifiers (V, q). We will now show that this upper complexity bound is optimal.

Theorem 6. *There exists a standard verifier (V, q) such that $\text{Invs}_{V,q}$ is Σ_2^p-complete.*

Proof. Since $\text{Invs}_{V,q} \in \Sigma_2^p$ for all standard verifiers (V, q), it suffices to show that there exists a standard verifier (V, q) such that $\text{Invs}_{V,q}$ is Σ_2^p-hard.

Consider the language $\exists\forall 3\text{SAT}$,

$$\begin{aligned}\exists\forall 3\text{SAT} = \{F \mid\ & F \text{ is a boolean formula in 3-DNF having } 2n \text{ variables} \\ & x_1, x_2, \ldots, x_n \text{ and } y_1, y_2, \ldots, y_n \text{ for some } n \in \mathbb{N} \text{ and} \\ & (\exists \alpha \in \{0,1\}^n)(\forall \beta \in \{0,1\}^n)[F(\alpha, \beta) = 1]\},\end{aligned}$$

where $F(\alpha, \beta)$ denotes the truth value of F when using α and β as assignments for the variables $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$, respectively.

$\exists\forall 3\text{SAT}$ is known to be Σ_2^p-complete [12]. Let *encode* be a polynomial-time computable and polynomial-time invertible encoding function for boolean formulas in 3-DNF. Let *double* be a mapping from {0,1}* to {0,1}* such that for all $k \in \mathbb{N}$ and all $a_1, a_2, \ldots, a_k \in \{0, 1\}$, $double(a_1 a_2 \ldots a_k) = a_1 a_1 a_2 a_2 \ldots a_k a_k$.

Let $q(n) = n$ for all $n \in \mathbb{N}$. We define the following verifier (V, q):

> V accepts on input (u, v) if and only if there exist a natural number n, a boolean formula F in 3-DNF with $2n$ variables $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$, and strings $\alpha, \beta \in \{0,1\}^n$ such that $u = encode(F)01double(\alpha)$ and $v = encode(F)01double(\beta)$ and $F(\alpha, \beta) = 1$.

It is not hard to see that (V, q) is a standard verifier.

To show that $\exists\forall 3\text{SAT} \leq_m^p \text{Invs}_{V,q}$ we will map formulas F having the required syntactic properties (3-DNF, even number of variables) to the encoding c_F of acircuit—having $|encode(F)| + 2n + 2$ inputs—that accepts all strings of the form $encode(F)01double(\beta)$ for any $\beta \in \{0,1\}^n$ and rejects all other strings. All other formulas, i.e., those formulas not in 3-DNF or having an odd number of variables, are mapped to the encoding c of a circuit that accepts exactly one string, namely 0 (this ensures that if F does not have the the required syntactic properties and thus $F \notin \exists\forall 3\text{SAT}$, then $c \notin \text{Invs}_{V,q}$). The described reduction is clearly polynomial-time computable.

It remains to show that for all formulas F having the above-mentioned syntactic properties (3-DNF, even number of variables) it holds that $F \in \exists\forall 3\text{SAT} \iff c_F \in \text{Invs}_{V,q}$. Let $F \in \exists\forall 3\text{SAT}$. It follows that there exists a partial assignment $\alpha \in \{0,1\}^n$ such that for all partial assignments $\beta \in \{0,1\}^n$, $F(\alpha, \beta) = 1$. Hence there exists $u = encode(F)01double(\alpha)$ such that for all $v = encode(F)01double(\beta)$, $V(u, v) = 1$. By construction of c_F we thus have $c_F \in \text{Invs}_{V,q}$. For the other implication assume $F \notin \exists\forall 3\text{SAT}$. Hence for all $\alpha \in \{0,1\}^n$ there exists $\beta \in \{0,1\}^n$ such that $F(\alpha, \beta) = 0$. It follows from the definition of V that for all $u = encode(F)01double(\alpha)$ there exists $v = encode(F)01double(\beta)$ such that $V(u, v) = 0$. By construction of c_F we thus have $c_F \notin \text{Invs}_{V,q}$.

This completes the proof. □

Note that the verifier V defined in the proof of Theorem 6 is a verifier for the language L of all strings w such that there exist a natural number n, a boolean formula F in 3-DNF with $2n$ variables $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$, and a string $\alpha \in \{0,1\}^n$ such that

$$w = encode(F)01double(\alpha) \wedge (\exists\beta \in \{0,1\}^n)[F(\alpha, \beta) = 1].$$

It is not hard to see that $L \in \text{P}$ since satisfiability for 3-DNF formulas can be checked in polynomial time.

Corollary 4. *There exists a language $L \in \text{P}$ and a standard verifier (V, q) for L such that $\text{Invs}_{V,q}$ is Σ_2^p-complete.*

In fact, looking carefully at the preceding construction, we see that the just-given proof establishes also the following one-sided result (see also the related comments preceding Corollary 5).

Corollary (to the proof) 7. *There exists a language $L \in \text{P}$ and a standard verifier (V, q) for L such that 1-$\text{Invs}_{V,q}$ is Σ_2^p-complete.*

So even simple sets can have very hard inverse problems (Corollaries 4 and 7). Nonetheless, all (NP) sets have at least one standard verifier whose one-sided inverse problem is not too hard, namely, it at least belongs to DP (note: if $\mathrm{DP} = \Sigma_2^p$ then PH collapses to DP).

Theorem 8. *Every set* $A \in \mathrm{NP}$ *has a standard verifier* (V, q) *such that* $1\text{-Invs}_{V,q} \in \mathrm{DP}$.

Proof. Let $A \in \mathrm{NP}$ and let (R, p) be a standard verifier for A. Let $q(n) = n + p(n)$ and define a verifier V as follows:

> V accepts on input (a, b) if and only if there exists a string b' such that $b = ab'$ and $R(a, b') = 1$.

It is not hard to see that (V, q) is a standard verifier for A.

By definition we have

$$1\text{-Invs}_{V,q} = \{c \mid c \text{ encodes a circuit } c' \text{ having } q(m) \text{ inputs for some } m \in \mathbb{N} \text{ such that } (\exists x \in A^{=m}) [\{w \in \Sigma^{q(m)} \mid V(x, w) = 1\} = \{y \in \Sigma^{q(m)} \mid c'(y) = 1\}]\}.$$

This can be rewritten, keeping in mind the particular V we have defined, as follows.

$$1\text{-Invs}_{V,q} = \{c \mid c \text{ encodes a circuit } c' \text{ having } q(m) \text{ inputs for some } m \in \mathbb{N} \text{ such that:}$$
$$(\forall u, v \in \Sigma^{q(m)})[\text{if } c'(u) = c'(v) = 1 \text{ then the first } m \text{ bits of } u \text{ and } v \text{ are identical}] \text{ and}$$
$$(\forall u, v \in \Sigma^{q(m)})(\forall x \in \Sigma^m)[\text{if } c'(u) = 1 \text{ and } c'(v) = 0 \text{ and } u = xu' \text{ and } v = xv' \text{ then } R(x, u') = 1 \text{ and } R(x, v') = 0] \text{ and}$$
$$(\forall u \in \Sigma^{q(m)})(\forall x \in \Sigma^m)[\text{if } c'(u) = 1 \text{ and } u = xu' \text{ then } R(x, u') = 1] \text{ and}$$
$$(\exists v \in \Sigma^{q(m)})[c'(v) = 1]\}.$$

This rewritten version (keeping in mind that the quantification over m is not a "real" quantifier) makes it clear that $1\text{-Invs}_{V,q} \in \mathrm{DP}$, as it is of the form $A \cap B \cap C \cap D$, with $A, B, C \in \mathrm{coNP}$ and $D \in \mathrm{NP}$, and so is of the form of the difference of two NP sets, namely, $D - (\overline{A \cap B \cap C})$. □

4 The Complexity of Recognizing Verifiers

In this section we show that deciding if a given machine computes a (general) verifier is complete for the second level of the arithmetic hierarchy, Σ_2^0.

Let $M_1, M_2, M_3, \ldots$ be a standard enumeration of deterministic 2-ary Turing machines.

Theorem 9. *The index set* $I_{ver,gen} = \{i \in \mathbb{N} \mid M_i \text{ computes a general verifier}\}$ *is* $\leq_m$*-complete for* Σ_2^0.

Proof. It is not hard to see that $I_{ver,gen} \in \Sigma_2^0$ since $I_{ver,gen}$ can be described as follows: $i \in I_{ver,gen} \iff (\exists k \in \mathbb{N})(\forall x, y \in \Sigma^*)[M_i(x,y)$ halts within at most $(|x| + |y|)^k + k$ steps and if $M_i(x,y)$ accepts within at most $(|x| + |y|)^k + k$ steps then $|y|^k + k \geq |x|]$. Note that the right hand side of the above "$\iff$" shows membership in Σ_2^0.

It remains to show that $I_{ver,gen}$ is $\leq_m$-hard for Σ_2^0. Since $I_{finite} = \{i \mid L(N_i)$ is finite$\}$ (where $N_1, N_2, N_3, \ldots$ is a fixed standard enumeration of Turing machines, e.g., that of Hopcroft–Ullman [6]) is $\leq_m$-hard (even $\leq_m$-complete) for Σ_2^0 it suffices to show that $I_{finite} \leq_m I_{ver,gen}$. Given (to our reduction) any $i \in \mathbb{N}$, by the nice properties of the standard enumeration, we can effectively construct from i a machine E that is an enumerator for $L(N_i)$. We now describe a Turing machine $\widehat{M}$. $\widehat{M}$ is a 2-ary Turing machine that on input $(x,y) \in \Sigma^* \times \Sigma^*$ does the following steps:

1. Simulate $|x| + |y|$ steps of the work of E and let A be the set of all strings that are enumerated by E within those $|x| + |y|$ steps.
2. Simulate $2(|x| + |y|)$ steps of the work of E and let B be the set of all strings that are enumerated by E within those $2(|x| + |y|)$ steps.
3. Accept (i.e., output true) if $B - A \neq \emptyset$; otherwise reject (i.e., output false).

Clearly, $\widehat{M}$ is a 2-ary Turing machine. Let j be an index such that $M_j = \widehat{M}$ (we assume our standard enumeration is expansive enough to include all the obviously 2-ary, deterministic machines created by this construction—this is a legal assumption). Since j clearly depends only on i we have implicitly described a mapping $f : \mathbb{N} \to \mathbb{N}$, $f(i) = j$. Note that f is computable.

It suffices to show that for all $i \in \mathbb{N}$, $i \in I_{finite} \iff f(i) = j \in I_{ver,gen}$. Let $i \in \mathbb{N}$ and suppose $i \in I_{finite}$. So $L(M_i)$ is finite and the number of strings enumerated by E is finite as well. Note that since M_j by definition runs in polynomial time and since E enumerates only a finite number of strings it follows from the construction of M_j that M_j accepts only a finite number of inputs and thus it holds that there exists a strictly monotonic polynomial p such that for all $x, y \in \Sigma^*$, if $M_j(x,y)$ outputs true then $p(|y|) \geq |x|$. So (remembering also the polynomial-time claim made above) M_j computes a general verifier and thus $j \in I_{ver,gen}$.

If $i \notin I_{finite}$ then E enumerates an infinite number of strings and thus for all $x \in \Sigma^*$, $M_j(x,y)$ outputs true for infinitely many $y \in \Sigma^*$. So there does not exist a (strictly monotonic) polynomial p such that, for all $x, y \in \Sigma^*$, if $M_j(x,y)$ outputs true then $p(|y|) \geq |x|$. Thus, M_j does not compute a general verifier and so $j \notin I_{ver,gen}$. □

We in this section focused on general rather than standard verifiers since when asking about decidability, the "hit the length on the head"-ness of standard verifiers becomes something of an unnatural technical obstacle. Nonetheless, we mention in passing that the Σ_2^0 upper bound holds cleanly for that case, and that though the particular Σ_2^0-hardness proof given above does not directly apply for

the standard verifier case, we have a more involved construction approach for that case, which we plan to investigate and discuss in the full version of this paper.

5 Conclusions

We have shown that all superlinear inversion schemes are coNP-hard. We have also shown that some inversion schemes are NP-hard and others are Σ_2^p-complete. Note that for finite sets A and any of their standard verifiers (V, q) we have that $\mathbf{Invs}_{V,q}$ is coNP-complete. It is not clear whether the complexity of inverting standard verifiers for infinite NP sets is also independent of the verifier. In particular, does every infinite NP set have a standard verifier (V, q) such that $\mathbf{Invs}_{V,q}$ is Σ_2^p-complete?

References

1. J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing,* 17(6):1232–1252, 1988.
2. J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy II: Applications. *SIAM Journal on Computing,* 18(1):95–111, 1989.
3. H. Chen. Inverse NP problems. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science,* pages 338–347. Springer-Verlag *Lecture Notes in Computer Science #2747,* August 2003.
4. L. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion.* Springer-Verlag, 2002.
5. L. Hemachandra and S. Rudich. On the complexity of ranking. *Journal of Computer and System Sciences,* 41(2):251–271, 1990.
6. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
7. A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory,* pages 125–129, October 1972.
8. C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
9. C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences,* 28(2):244–259, 1984.
10. H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.
11. L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science,* 3(1):1–22, 1976.
12. C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science,* 3(1):23–33, 1976.

The Complexity of Equivalence and Isomorphism of Systems of Equations over Finite Groups

Gustav Nordh*

Department of Computer and Information Science
Linköpings Universitet
S-581 83 Linköping, Sweden
Fax: +46 13 28 44 99
gusno@ida.liu.se

Abstract. We study the computational complexity of the isomorphism and equivalence problems on systems of equations over a fixed finite group. We show that the equivalence problem is in P if the group is Abelian, and coNP-complete if the group is non-Abelian. We prove that if the group is non-Abelian, then the problem of deciding whether two systems of equations over the group are isomorphic is coNP-hard. If the group is Abelian, then the isomorphism problem is graph isomorphism hard. Moreover, if we impose the restriction that all equations are of bounded length, then we prove that the isomorphism problem for systems of equations over finite Abelian groups is graph isomorphism complete. Finally we prove that the problem of counting the number of isomorphisms of systems of equations is no harder than deciding whether there exist any isomorphisms at all.

1 Introduction

The computational complexity of deciding whether systems of equations over a fixed finite group are solvable has been studied in the past. In [5], it is proved that the problem is in P if the group is Abelian and NP-complete otherwise. This line of research continued in [8,10], where the corresponding problem for finite monoids was given a complete solution. Some partial results in the general case of finite semigroups have been proved in [10]. Note that even the restricted problem of determining the computational complexity of solving systems of equations over a fixed regular semigroup is still open. The problem of deciding whether systems of equations over a fixed finite group $(G, \cdot)$ are solvable is denoted $\textsc{Eqn}^*_G$ in the literature.

The computational complexity of counting solutions to systems of equations over a fixed finite semigroup has been studied in [9], where it is proved that if the semigroup is an Abelian group, then the problem is in FP, and if the semigroup is a non-Abelian group, then the problem is #P-complete. This problem is denoted $\#\textsc{Eqn}^*_G$.

* Supported by the *National Graduate School in Computer Science* (CUGS), Sweden.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 380–391, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In this paper we study the computational complexity of deciding whether systems of equations over a fixed finite group are equivalent/isomorphic. We also study the problem of counting the number of isomorphisms. These fundamental problems have as far as we know eluded previous investigations from a computational perspective, except for some results on the Boolean constraint equivalence and isomorphism problems [2,3] that are also relevant in our setting. More specifically the equivalence problem for systems of equations of bounded length over the two element group $\mathbb{Z}_2$ is in P, and the corresponding isomorphism problem is GRAPH ISOMORPHISM-complete. The computational complexity of several other isomorphism and equivalence problems have been intensively studied in the past, most notably the graph isomorphism problem [6] and the formula isomorphism problem [1].

A system of equations over a fixed finite group $(G, \cdot)$ is a collection of equations of the form $x_1 \cdot x_2 \cdot \ldots \cdot x_k = x_{k+1} \cdot \ldots \cdot x_n$, where each x_i is either a variable or a constant in G.

Definition 1. *Let S be a system of equations on variables X and let π be a permutation of X. By $\pi(S)$ we denote the system of equations that results when we replace all variables x in S by $\pi(x)$.*

- EQUIV-EQN$^*_G$ *is the problem of deciding whether two systems of equations S_1 and S_2 on variables X over $(G, \cdot)$ are equivalent, i.e., whether for every assignment of the variables in X to values in G, S_1 is satisfied if and only if S_2 is satisfied. Note that when we say that S_1 and S_2 are systems of equations on variables X we only mean that X is the union of the variables in S_1 and S_2, hence all variables in X need not occur both in S_1 and S_2. If S_1 is equivalent to S_2 we denote this by $S_1 \equiv S_2$.*
- ISO-EQN$^*_G$ *is the problem of deciding whether two systems of equations S_1 and S_2 on variables X over $(G, \cdot)$ are isomorphic, i.e., whether there exist a permutation π of the variables in X such that $\pi(S_1) \equiv S_2$. If S_1 is isomorphic to S_2 we denote this by $S_1 \cong S_2$.*
- #ISO-EQN$^*_G$ *is the counting version of* ISO-EQN$^*_G$*, i.e., the problem of counting the number of permutations π of the variables in X such that $\pi(S_1) \equiv S_2$.*

The complexity of EQUIV-EQN$^*_G$ (ISO-EQN$^*_G$, #ISO-EQN$^*_G$) is measured in the size of the systems of equations (the size of $(G, \cdot)$ is fixed and does not matter).

We prove that if the group is non-Abelian the equivalence problem is coNP-complete and if the group is Abelian the equivalence problem is in P. When it comes to the isomorphism problem we prove that if the group is non-Abelian, the isomorphism problem is coNP-hard. If the group is Abelian, then the isomorphism problem is GRAPH ISOMORPHISM-hard. If we impose the restriction that all equations are of bounded length, then the isomorphism problem for systems of equations over finite Abelian groups becomes GRAPH ISOMORPHISM-complete. For the problem of counting the number of isomorphisms, we give an algorithm that shows that it is no harder to count the number of isomorphisms than to decide whether any isomorphisms exists at all. As a corollary to this

algorithm we obtain the result that ISO-EQN$^*_G$ for Abelian groups is powerless as an oracle to PP, and ISO-EQN$^*_G$ for non-Abelian groups is no more powerful than an NP-oracle for PP. These results indicates that ISO-EQN$^*_G$ for Abelian groups is not NP-complete, and that ISO-EQN$^*_G$ for non-Abelian groups is not Σ_2^P-complete.

The paper is organized as follows. In Sect. 2 we prove our results concerning the equivalence problem, Sect. 3 deals with the isomorphism problem, Sect. 4 treats the corresponding counting problem, and finally in Sect. 5 we present our conclusions and some ideas for future research.

2 Equivalence

In this section we investigate the computational complexity of EQUIV-EQN$^*_G$, that is, the problem of deciding whether two systems of equations over a fixed finite group are equivalent. We prove that EQUIV-EQN$^*_G$ is coNP-complete if $(G, \cdot)$ is non-Abelian, and that EQUIV-EQN$^*_G$ is in P if $(G, \cdot)$ is an Abelian group.

First note that it is easy to see that the equivalence problem is in coNP.

Theorem 1. EQUIV-EQN$^*_G$ *is in coNP.*

Proof. Nondeterministically choose an assignment of the variables in S_1 and S_2 to values in G. It can be checked in polynomial time whether this assignment satisfies exactly one of S_1 and S_2. Hence the problem of deciding whether two systems of equations S_1 and S_2 over $(G, \cdot)$ are inequivalent, is in NP. Thus it follows that the problem of deciding whether two systems of equations over a fixed finite group are equivalent is in coNP. □

The following theorem states that if it is hard to decide whether systems of equations over a group $(G, \cdot)$ are solvable, then it is also hard to decide whether systems of equations over the same group are equivalent.

Theorem 2. *If* EQN$^*_G$ *is NP-complete then* EQUIV-EQN$^*_G$ *is coNP-complete.*

Proof. If EQN$^*_G$ is NP-complete it follows that it is coNP-complete to decide whether a system of equations over $(G, \cdot)$ is insoluble. Since a system of equations is insoluble if and only if it is equivalent to an insoluble system of equations (for example a system of equations containing the equation $0 = 1$), it follows that EQUIV-EQN$^*_G$ is coNP-complete. □

The previous theorem and the fact that EQN$^*_G$ is NP-complete when $(G, \cdot)$ is a non-Abelian group [5] immediately implies the following corollary.

Corollary 1. EQUIV-EQN$^*_G$ *is coNP-complete when* $(G, \cdot)$ *is a non-Abelian group.*

Next we prove that if it is easy to count the number of solutions to systems of equations over a group $(G, \cdot)$, then it is also easy to decide whether systems of equations over the same group are equivalent.

Theorem 3. EQUIV-EQN$^*_G$ *is in P if* #EQN$^*_G$ *is in FP.*

Proof. Let S_1 and S_2 be two systems of equations. Count the number of solutions to S_1 and S_2, if they have different number of solutions they are not equivalent. Thus we can assume that S_1 and S_2 have the same number of solutions. Count the number of solutions to the system of equations consisting of the union of S_1 and S_2. If the number of solutions to this system of equations equals the number of solutions to S_1, we know that S_1 and S_2 have the same set of solutions and hence S_1 is equivalent to S_2, otherwise S_1 and S_2 are inequivalent. □

The following corollary follows directly from Theorem 3 and the fact that #EQN$^*_G$ is in FP when $(G, \cdot)$ is an Abelian group [9].

Corollary 2. EQUIV-EQN$^*_G$ *is in P when* $(G, \cdot)$ *is an Abelian group.*

It should be clear that Theorem 2 and Theorem 3 also holds when generalized to systems of equations over a fixed finite semigroup $(S, \cdot)$. Hence interesting results on the complexity of EQUIV-EQN$^*_S$, where $(S, \cdot)$ is finite semigroup, can be deduced from the results for #EQN$^*_S$ and EQN$^*_S$ proved in [8,9,10]. We collect these results in the following corollary.

Corollary 3. EQUIV-EQN$^*_S$ *is coNP-complete when*

- $(S, \cdot)$ *is a monoid but is not in the join of Abelian groups and idempotent Abelian monoids [8],*
- $(S, \cdot)$ *is a band but is not a normal band [10],*
- $(S, \cdot)$ *is a regular semigroup but is not a strong normal band of Abelian groups [10].*

EQUIV-EQN$^*_S$ *is in P when*

- $(S, \cdot)$ *is a direct product of an Abelian group and a rectangular band [9],*
- $(S, \cdot)$ *is a semigroup with zero, such that for all elements* x, y, $x \cdot y = 0$ *[9].*

3 Isomorphism

In this section we investigate the computational complexity of ISO-EQN$^*_G$, that is, the problem of deciding whether two systems of equations over a fixed finite group are isomorphic. We prove that ISO-EQN$^*_G$ is coNP-hard if $(G, \cdot)$ is non-Abelian, and that ISO-EQN$^*_G$ is GRAPH ISOMORPHISM-hard if $(G, \cdot)$ is an Abelian group. If we restrict the problem and demand that all equations are of bounded length, then ISO-EQN$^*_G$ in the Abelian case becomes GRAPH ISOMORPHISM-complete, and ISO-EQN$^*_G$ in the non-Abelian case is in $\mathrm{P}_{||}^{NP}$, i.e., the class of problems solvable in polynomial time with parallel access to an NP-oracle.

We begin by giving upper bounds for the complexity of the isomorphism problem.

Theorem 4. ISO-EQN$^*_G$ *is in NP when* $(G, \cdot)$ *is an Abelian group and* ISO-EQN$^*_G$ *is in* Σ_2^{P} *when* $(G, \cdot)$ *is a non-Abelian group.*

Proof. We begin by proving the Σ_2^P upper bound. Nondeterministically choose a permutation π of the variables in X and use an NP-oracle to check whether $\pi(S_1) \equiv S_2$. Hence ISO-EQN^*_G is in $\text{NP}^{\text{NP}} = \Sigma_2^P$.

The NP upper bound for ISO-EQN^*_G when $(G, \cdot)$ is an Abelian group follows from the results in the previous section on the equivalence problem. We know from Corollary 2 that $\pi(S_1) \equiv S_2$ can be decided in polynomial time when $(G, \cdot)$ is an Abelian group, hence ISO-EQN^*_G is in NP when $(G, \cdot)$ is an Abelian group. □

The following theorem states that if it is hard to decide whether systems of equations over a group $(G, \cdot)$ are solvable, then it is also hard to decide whether systems of equations over the same group are isomorphic.

Theorem 5. *If* EQN^*_G *is NP-complete then* ISO-EQN^*_G *is coNP-hard.*

Proof. If EQN^*_G is NP-complete, then it follows that it is coNP-hard to decide whether a system of equations over $(G, \cdot)$ is insoluble. Since a system of equations is insoluble if and only if it is isomorphic to an insoluble system of equations, e.g., a system of equations containing the equation $0 = 1$, it follows that ISO-EQN^*_G is coNP-hard. □

The previous theorem and the fact that EQN^*_G is NP-complete when $(G, \cdot)$ is a non-Abelian group [5] immediately implies the following corollary.

Corollary 4. ISO-EQN^*_G *is coNP-hard when* $(G, \cdot)$ *is a non-Abelian group.*

The following theorem indicates that ISO-EQN^*_G for Abelian groups perhaps is not in P, or at least that it is hard to prove that ISO-EQN^*_G is in P for Abelian groups.

Theorem 6. *Let* $(G, \cdot)$ *be a finite Abelian group, then* GRAPH ISOMORPHISM *is polynomial-time many-one reducible to* ISO-EQN^*_G.

Proof. It is known that GRAPH ISOMORPHISM is polynomial-time equivalent to the graph isomorphism problem restricted to bipartite graphs [4]. To simplify the proof we assume from now on that all graphs are bipartite and contains no isolated vertices. Let T be the following transformation from a graph $H = (V, E)$, where $V = \{1, 2, \ldots, n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$, to a system of equations S over $(G,+)$:

$$T(H) = \{x_i + x_j + y_k = 1 \mid e_k = \{i, j\}\} \cup \{x_i + z_i + z_i' = 1 \mid i \in V\}.$$

Note that since the group is Abelian we adopt the common notation and denote the group operation by +. Furthermore, as we will explain later, it is sufficient to consider the case where $(G, +)$ is a cyclic group (that is groups of the form $\mathbb{Z}_{p^r}$ for some prime p and positive integer r). We let 1 denote one of the group generators, and 0 the identity element.

The proof of the theorem relies on the following lemma, which shows that $T(H)$ is a maximum set of equations of type, $a + b + c = 1$. This property is

important since, checking whether two maximum sets of equations are equivalent basically amounts to checking whether the sets are equal. This is the same approach as previously used in the treatment of the isomorphism problem for Boolean constraints in [2]. Note that $T(H)$ would not necessary be a maximum set of equations of type $a+b+c=1$ for non-bipartite graphs H.

Let $T(H) \to a+b+c=1$ (where a,b,c are variables occurring in $T(H)$) denote that the equation $a+b+c=1$ can be inferred from the system of equations, $T(H)$, i.e., there exist no assignment of the variables in $T(H)$ such that $T(H)$ is satisfied and $a+b+c=1$ is not satisfied. Note that since $(G,+)$ is Abelian, we (for example) consider $a+b+c=1$ as being identical to $b+a+c=1$.

Lemma 1. *The following holds for every triple a,b,c of distinct variables in $T(H)$: If $T(H) \to a+b+c=1$, then $a+b+c=1 \in T(H)$.*

The proof of the lemma is a straightforward but tedious case analysis that we have to omit due to space limitations. The importance of the fact that $T(H)$ is a maximum set of equations of the form $a+b+c=1$ lies in the observation that $T(H_1) \equiv T(H_2)$ if and only if $T(H_1) = T(H_2)$. Note again that since $(G,+)$ is Abelian we consider $a+b+c=1$ as being identical to $b+c+a=1$, and so on. Hence $T(H_1)$ is isomorphic to $T(H_2)$ if and only if there exists a permutation ρ of the variables in $T(H_1) \cup T(H_2)$ such that $\rho(T(H_1)) = T(H_2)$.

Let $H_1 = (V,E)$ and $H_2 = (V,E')$ be two bipartite graphs, with an equal number of vertices and edges, where $V = \{1,2,\ldots,n\}$, $E = \{e_1,e_2,\ldots,e_m\}$, and $E' = \{e'_1,e'_2,\ldots,e'_m\}$. If $\pi : V \to V$ is an isomorphism from H_1 to H_2 then ρ is an isomorphism from $T(H_1)$ to $T(H_2)$ defined as follows:

- $\rho(x_i) = x_{\pi(i)}$, $\rho(z_i) = z_{\pi(i)}$, $\rho(z'_i) = z'_{\pi(i)}$, for $i \in V$;
- For $e_k = \{i,j\}$, $\rho(y_k) = y_l$ where $e'_l = \{\pi(i),\pi(j)\}$.

For the converse, suppose that ρ is an isomorphism from $T(H_1)$ to $T(H_2)$. As we have already observed $\rho(T(H_1)) = T(H_2)$. Now consider the properties of the different classes of variables.

- Variables from X are exactly those variables that occur at least twice and that also occur in an element of $T(H)$ together with two variables that occur exactly once. So, ρ will map X onto X.
- Elements of Z are those variables that occur exactly once and that occur together with an element from X and another element that occurs exactly once. So, ρ will map Z to Z.
- Everything else is an element of Y. So, ρ will map Y to Y.

For $i \in V$, define $\pi(i) = j$ if and only if $\rho(x_i) = x_j$. π is one-to-one and onto by the observation above. What remains to be proved is that $\{i,j\} \in E$ if and only if $\{\pi(i),\pi(j)\} \in E'$. Let $e_k = \{i,j\}$. Then $x_i + x_j + e_k = 1 \in T(H_1)$. Hence, $\rho(x_i) + \rho(x_j) + \rho(y_k) = 1 \in T(H_2)$,i.e., $x_{\pi(i)} + x_{\pi(j)} + \rho(y_k) = 1 \in T(H_2)$. This implies that $\rho(y_k) = y_l$ where $e'_l = \{\pi(i),\pi(j)\}$, hence $\{\pi(i),\pi(j)\} \in E'$. For the converse, suppose that $\{\pi(i),\pi(j)\} \in E'$. Then $x_{\pi(i)} + x_{\pi(j)} + y_l = 1 \in T(H_2)$ for

$e_l = \{\pi(i), \pi(j)\}$, and thus $x_i + x_j + \rho^{-1}(y_l) \in T(H_1)$. By the structure of $T(H_1)$, it follows that $\{i, j\} \in E$. Hence we have proved that GRAPH ISOMORPHISM is polynomial-time many-one reducible to ISO-EQN$^*_G$ where $(G, +)$ is a finite cyclic group.

The fundamental theorem of finitely generated Abelian groups makes it possible to extended this result to all finite Abelian groups. Note that all finite Abelian groups are of course finitely generated.

Lemma 2. *Every finite Abelian group $(G, +)$ is isomorphic to a direct product of cyclic groups in the form $\mathbb{Z}_{p_1^{r_1}} \times \mathbb{Z}_{p_2^{r_2}} \times \ldots \times \mathbb{Z}_{p_n^{r_n}}$, where the p_i's are (not necessarily distinct) primes, and $|G| = p_1^{r_1} \cdot \ldots \cdot p_n^{r_n}$.*

Thus given a system of equations S_1 over a finite Abelian group $(G, +)$, we can use the fundamental theorem of finitely generated Abelian groups to view it as n independent systems of equations, each over a cyclic group $\mathbb{Z}_{p_i^{r_i}}$. It should be clear that two systems of equations S_1 and S_2 over $(G, +)$ are isomorphic if and only if S_1 is isomorphic to S_2 when viewed as systems of equations over $\mathbb{Z}_{p_i^{r_i}}$, for all $i \in \{1, \ldots, n\}$. We have already proved that H_1 is isomorphic to H_2 if and only if $T(H_1)$ is isomorphic to $T(H_2)$, when $T(H_1)$ and $T(H_2)$ are systems of equations over an arbitrary cyclic group. Hence given a finite Abelian group $(G, +)$ and two graphs H_1 and H_2, then H_1 is isomorphic to H_2 if and only if the two systems of equations $T(H_1)$ and $T(H_2)$ over $(G, +)$ are isomorphic. □

Despite intensive research GRAPH ISOMORPHISM is not known to be in coNP. Hence in light of the previous theorem it seems hard to prove that ISO-EQN$^*_G$ is in coNP when $(G, \cdot)$ is an Abelian group.

Corollary 5. ISO-EQN$^*_G$ *for Abelian groups is not in coNP unless* GRAPH ISOMORPHISM *is in coNP.*

Let ISO-B-EQN$^*_G$ denote the problem of deciding whether two systems of equations over $(G, \cdot)$, where each equation have bounded length, i.e., every equation are of length at most k for some constant k, are isomorphic.

Theorem 7. ISO-B-EQN$^*_G$ is GRAPH ISOMORPHISM-complete *when $(G, \cdot)$ is an Abelian group.*

Proof. By Theorem 6 we know that GRAPH ISOMORPHISM is polynomial-time reducible to ISO-B-EQN$^*_G$. Hence what remains is to give a polynomial-time reduction from ISO-B-EQN$^*_G$ to GRAPH ISOMORPHISM. We will actually give a reduction from ISO-B-EQN$^*_G$ to the GRAPH ISOMORPHISM-complete problem, vertex colored graph isomorphism (VCGI) [6].

Definition 2. *VCGI is the problem of, given two vertex-colored graphs $H_1 = (V_1, E_1, \chi_1)$, $H_2 = (V_2, E_2, \chi_2)$, $\chi_1, \chi_2 : V \to N$, to determine whether there exist an isomorphism from H_1 to H_2 that preserves colors, i.e., whether there exists a bijection $\pi : V_1 \to V_2$ such that $\{v, v'\} \in E_1$ if and only if $\{\pi(v), \pi(v')\} \in E_2$, and $\chi_1(v) = \chi_2(\pi(v))$ for all $v \in V_1$.*

Let S_1 and S_2 be two systems of equations of bounded length over an Abelian group $(G, \cdot)$ on variables X. We will first bring S_1 and S_2 into normal form. We choose an approach similar to that in Theorem 6 and reduce S_1 and S_2 to the maximum set of equations of length at most k that can be inferred from S_1 and S_2 respectively.

Let $\langle S_1 \rangle$ denote the set of all equations E of length less than k such that all of E's variables occur in X and $S_1 \rightarrow E$. It should be clear that S_1 is equivalent to $\langle S_1 \rangle$, since $S_1 \subseteq \langle S_1 \rangle$ and $S_1 \rightarrow \langle S_1 \rangle$. $\langle S_2 \rangle$ is defined analogously. Note that $\langle S_1 \rangle$ and $\langle S_2 \rangle$ can be computed in polynomial time (in $|S_1| + |S_2|$), since there exist at most $O(|X|^k)$ equations E of length less than k such that all of E's variables occur in X, and since $(G, \cdot)$ is Abelian we can use the polynomial time algorithm for EQN^*_G to decide whether $S \rightarrow E$. Note that $S \rightarrow E$ if and only if for every assignment to the variables in E such that E is not satisfied, this assignment applied to S makes S insoluble. Since there is at most $O(|G|^k)$ possible assignments of the variables in E, and testing whether such an assignment makes S insoluble can be done in polynomial time (by the result in [5]), it follows that $S \rightarrow E$ can be decided in polynomial time. It should be clear that if $\langle S_1 \rangle$ is equivalent to $\langle S_2 \rangle$ then $\langle S_1 \rangle = \langle S_2 \rangle$, hence if π is a permutation of the variables in X such that $\pi(\langle S_1 \rangle) \equiv \langle S_2 \rangle$, then $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$.

Now we proceed and encode $\langle S_1 \rangle$ and $\langle S_2 \rangle$ as vertex colored graphs H_1 and H_2 such that there exist a permutation π of X such that $\pi(\langle S_1 \rangle) \equiv \langle S_2 \rangle$ if and only if $(H_1, H_2) \in VCGI$.

Let $\langle S_1 \rangle$ consist of the equations $A_1, \ldots, A_m$, that is, $\langle S_1 \rangle =$

$$\begin{aligned}
x_{11} + x_{12} + \ldots + x_{1j_1} &= x_{1j_1+1} + \ldots + x_{1k_1} \quad (A_1) \\
x_{21} + x_{22} + \ldots + x_{2j_2} &= x_{2j_2+1} + \ldots + x_{2k_2} \quad (A_2) \\
&\vdots \\
x_{m1} + x_{m2} + \ldots + x_{mj_m} &= x_{mj_m+1} + \ldots + x_{mk_m} \quad (A_m)
\end{aligned}$$

Let $T(\langle S_1 \rangle) = H_1 = (V, E, \chi)$ be the following vertex colored graph:

- $V = \{0, 1, \ldots, |G| - 1\} \cup \{x \mid x \in X\} \cup \{a_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq k_i\} \cup \{A_i \mid 1 \leq i \leq m\}$. Hence, the set of vertices corresponds to the elements in $(G, \cdot)$, the variables in X, the elements in the equations in $\langle S_1 \rangle$, and the equations in $\langle S_1 \rangle$.
- $E = \{\{x, a_{ij}\} \mid x = a_{ij}\} \cup \{\{a_{ij}, A_i\} \mid 1 \leq i \leq m, 1 \leq j \leq k_i\}$.
- The vertex coloring χ is used to distinguish the different categories of vertices. Of course, we must allow any permutation of the variables, so all vertices corresponding to variables in X are assigned the same color. We also need to allow the permutation of vertices corresponding to equations, and because the group is Abelian we allow the permutation of vertices corresponding to elements within the same side of each equation.
  - $\chi(i) = i$,
  - $\chi(x) = |G|$ for all $x \in X$,
  - $\chi(A_r) = |G| + 1$,
  - $\chi(a_{ij}) = |G| + 2$ if $j \leq j_i$,

- $\chi(a_{ij}) = |G| + 3$ if $j > j_i$

Define $T(\langle S_2 \rangle) = H_2$ in the analogous way.

If there is a permutation π of X such that $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$, then it is easy to see that $(T(\langle S_1 \rangle), T(\langle S_2 \rangle)) \in VCGI$. On the other hand, if $(T(\langle S_1 \rangle), T(\langle S_2 \rangle)) \in VCGI$ by a permutation π on $T(\langle S_1 \rangle)$ it should be clear that $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$. Just note that any permutation of vertices corresponding to equations forces a permutation of the vertices corresponding to elements in the equations. Thus if $\pi(A_i) = A_j$ then necessarily $\pi(a_{it}) = a_{js}$, for all $1 \leq t \leq k_i$ and some $1 \leq s \leq k_j$. The only remaining part of the permutation in $T(\langle S_1 \rangle)$ is the permutation of vertices corresponding to variables in X, hence $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$. □

Using the same idea as in the proof of the preceding theorem we can prove a tighter upper bound for Iso-B-Eqn$^*_G$, compared to the trivial Σ_2^P upper bound from Theorem 4. More specifically we prove that Iso-B-Eqn$^*_G$ is in $P_{||}^{NP}$, the class of problems solvable in polynomial time with parallel (i.e., truth-table) access to an NP-oracle.

Theorem 8. Iso-B-Eqn$^*_G$ *is in* $P_{||}^{NP}$.

Proof. Let S_1 and S_2 be two systems of equations on variables X over a non-Abelian group $(G, \cdot)$. We use the same normal form as in the proof of the preceding theorem, i.e., we compute $\langle S_1 \rangle$ and $\langle S_2 \rangle$. Following the proof of the preceding theorem it is easy to verify that $\langle S_1 \rangle$ and $\langle S_2 \rangle$ can be computed in polynomial time with parallel access to an NP-oracle. Now the existence of a permutation π of the variables in X such that $\pi(\langle S_1 \rangle) = \langle S_2 \rangle$ can be determined with one query to an NP-oracle. Hence, it can be decided in polynomial time with two rounds of parallel queries to NP whether S_1 and S_2 are isomorphic, and thus Iso-B-Eqn$^*_G$ is in $P_{||}^{NP}$. □

4 Counting Isomorphisms

Mathon showed that the counting version of Graph Isomorphism is polynomial time Turing reducible to the decision version [7]. Thus Graph Isomorphism behaves differently than the known NP-complete problems. This was historically the first hint that Graph Isomorphism might not be NP-complete. We prove analogous results for #Iso-Eqn$^*_G$.

Given a system of equations over a finite group $(G, \cdot)$ we are interested in the set of permutations π of the variables in X such that $\pi(S) \equiv S$. It is easy to see that this set of permutations forms a group, denoted *aut*(S), the automorphism group of S. Aut-Eqn$^*_G$ is the problem of computing the automorphism group of a system of equations S over $(G, \cdot)$. #Aut-Eqn$^*_G$ is the corresponding counting problem.

The following theorem states that it is no harder to count the number of isomorphisms between two systems of equations over a fixed finite group than to decide whether an isomorphism exists at all. This indicates that Iso-Eqn$^*_G$ is not NP-complete for Abelian groups, and that Iso-Eqn$^*_G$ is not Σ_2^P-complete for non-Abelian groups.

Theorem 9. $\#\text{ISO-EQN}^*_G$ *is Turing equivalent to* ISO-EQN^*_G.

Proof. The proof is based on the same principle as the analogous proof for GRAPH ISOMORPHISM due to Mathon [7]. First note that ISO-EQN^*_G is trivially polynomial time reducible to $\#\text{ISO-EQN}^*_G$. Let N_1 be the number of permutations π of X such that $\pi(S_1) \equiv S_2$. If $S_1 \not\equiv S_2$, i.e., there exist no permutation π of X such that $\pi(S_1) \equiv S_2$, then $N_1 = 0$. Otherwise we claim that $N_1 = |aut(S_1)| = |aut(S_2)|$. This follows easily from the fact that if π is a permutation such that $\pi(S_1) \equiv S_2$, and $\rho \in aut(S_1)$ then $\rho \circ \pi$ is also a permutation such that $\rho \circ \pi(S_1) \equiv S_2$ (note that $\rho \circ \pi$ denotes the composition where ρ is applied prior to π, hence $\rho \circ \pi(S_1) = \pi(\rho(S_1))$). Moreover any permutation π' such that $\pi'(S_1) \equiv S_2$ can be uniquely expressed in the form $\pi' = \rho' \circ \pi$, where $\rho' \in aut(S_1)$. Thus the set of isomorphisms from S_1 to S_2 is the right coset, $aut(S_1) \circ \pi$, where π is an isomorphism from S_1 to S_2. Hence $\#\text{ISO-EQN}^*_G$ is Turing reducible to ISO-EQN^*_G if $\#\text{AUT-EQN}^*_G$ is Turing reducible to ISO-EQN^*_G, which we prove next. □

Lemma 3. $\#\text{AUT-EQN}^*_G$ *is Turing reducible to* ISO-EQN^*_G.

Proof. First note that we can determine whether a system of equations S is solvable or not by making a single query to ISO-EQN^*_G. If S is insoluble then of course $|aut(S)| = |X|!$.

Given a solvable system of equations S we need a construction that essentially forces a variable to be mapped to itself under any permutation $\rho \in aut(S)$. We call variables x_i and x_j equivalent with respect to S if, for any assignment α that satisfies S, we have $\alpha(x_i) = \alpha(x_j)$. By $E_S(x_i)$ we denote the set of variables of S that are equivalent to x_i. Consider any automorphism $\rho \in aut(S)$, if ρ maps x_i to x_k, then clearly $\rho(E_S(x_i)) = E_S(x_k)$.

This property can be used to force a variable to be mapped to itself (or an equivalent variable), since ρ always maps x_i to a variable x_k such that $|E_S(x_i)| = |E_S(x_k)|$. The trick is to make $|E_S(x_i)|$ unique by introducing new equations of the form $x_i = z_j$, where z_j is a new variable called a labelling variable. Let $L(x_i, t)$ denote $\bigcup_{j=1}^{t} x_i = z_j$, and $S_{[I]}$ the system of equations where all (inequivalent) variables in I have been given a unique label.

We construct the system of equations $S_{[I]}$ by successively labelling the variables in I. Let $S_0 = S$. Suppose that we have already labelled the first k variables in I and obtained the system of equations S_k for some $k \geq 0$. Let x_i be the next variable in I to be labelled. That is, x_i is not equivalent to any of the variables already labelled. Let t be the least integer such that $|E_{S_k}(x_i)| + t \neq |E_{S_k}(x_j)|$ for any $x_j \notin E_{S_k}(x_i)$, then let $S_{k+1} = S_k \cup L(x_i, t)$. Note that when S is a system of equations over an Abelian group, then $|E_S(x_i)|$ can be computed in polynomial time, just check whether $S \equiv S \cup x_i = x_j$ for all x_j in X (remember that by Corollary 2 we know that equivalence of systems of equations over Abelian groups is in P). In the case where S is a system of equations over a non-Abelian group we can use an NP-oracle to check whether $S \equiv S \cup x_i = x_j$ for all x_j in

X. Since ISO-EQN$^*_G$ for non-Abelian groups is coNP-hard, we can in particular use this method when we have ISO-EQN$^*_G$ as an oracle.

The procedure above is repeated at most $|I| \leq n$ times and the number t is bounded by n in each iteration, where $n = |X|$. Hence $S_{[I]}$ can be constructed in polynomial time in the Abelian case, and in polynomial time with ISO-EQN$^*_G$ as an oracle in the non-Abelian case.

Now with the technicalities of the labelling out of the way we can proceed to compute $|aut(S)|$. Let $S_{[x_1,\ldots,x_n]}$ be a system of equations where all variables have been labelled in the way described above. First note that we are only interested in automorphisms that acts as the identity mapping on the variables introduced in the labelling and the variables still labelled, this subgroup of the full automorphism group is denoted $aut'(S)$. Hence $|aut'(S_{[x_1,\ldots,x_n]})| = 1$. The crucial observation is that $|aut'(S_{[x_1,\ldots,x_{i-1}]})| = d_i|aut'(S_{[x_1,\ldots,x_{i-1},x_i]})|$, where d_i is the size of the orbit of x_i under $aut'(S_{[x_1,\ldots,x_{i-1}]})$. It should be clear that $|aut'(S_0)| = |aut(S)| = d_1d_2\ldots d_n$. The fact that the order of a permutation group can be computed from the size of the orbits in the way described above is a standard result from group theory, consult [6] for the details.

The orbit of x_i under $aut'(S_{[x_1,\ldots,x_{i-1}]})$ can be found by making $n-i$ queries to ISO-EQN$^*_G$. Ask the query $S_{[x_1,\ldots,x_{i-1}]}[x_i] \cong S_{[x_1,\ldots,x_{i-1}]}[x_j]$ for each variable x_j, $j \geq i$, where $S_{[x_1,\ldots,x_{i-1}]}[x_i]$ and $S_{[x_1,\ldots,x_{i-1}]}[x_j]$ denotes systems of equations where x_i and x_j have been assigned the same (unique) label (using the method described above). If the answer is yes, we know that x_j is in the orbit of x_i under $aut'(S_{[x_1,\ldots,x_{i-1}]})$. Hence $|aut(S)| = d_1d_2\ldots d_n$ can be computed by making $O(n^2)$ queries to ISO-EQN$^*_G$ and thus #AUT-EQN$^*_G$ is Turing reducible to ISO-EQN$^*_G$. □

Mathon's algorithm for computing the number of isomorphisms of two graphs, have the interesting property that at each intermediate stage the number of isomorphisms of the labelled graphs are known. This property has been exploited to prove that GRAPH ISOMORPHISM (GI) is low for PP, i.e., GRAPH ISOMORPHISM is powerless as an oracle to PP ($PP^{GI} = PP$) [6]. This is generally interpreted as further evidence for the hypothesis that GRAPH ISOMORPHISM is not NP-complete. For the details and significance of lowness results, again consult [6]. The properties of our algorithm (that is based on Mathon's algorithm) for computing #ISO-EQN$^*_G$ together with the same line of reason as that preceding Theorem 5.3 in [1] implies the following lowness results.

Corollary 6. *$PP^{IAb} = PP$, and $PP^{IG} = PP^{NP}$, where IAb denotes* ISO-EQN$^*_G$ *for Abelian groups and IG denotes* ISO-EQN$^*_G$ *for non-Abelian groups.*

Hence ISO-EQN$^*_G$ for Abelian groups is powerless as an oracle to PP, and ISO-EQN$^*_G$ for non-Abelian groups is no more powerful than an NP-oracle for PP.

5 Conclusions

We give a dichotomy for the complexity of EQUIV-EQN$^*_G$ for all finite groups $(G, \cdot)$. A natural direction for future research would be to prove a similar dichotomy for EQUIV-EQN$^*_S$ for all finite semigroups $(S, \cdot)$.

When it comes to the complexity of ISO-EQN^*_G, the situation is not as clear. We prove the upper bound of Σ^P_2 and the lower bound of coNP-hard in the non-Abelian case, and the upper bound of NP and the lower bound of GRAPH ISOMORPHISM-hard in the Abelian case. But the results in Theorem 9 and Corollary 6 give strong indications that these upper bounds (Σ^P_2 and NP respectively) are not tight.

Moreover, it has been proved, by the use of interactive proofs, that GRAPH ISOMORPHISM and formula isomorphism is not NP-complete and Σ^P_2-complete respectively, unless the polynomial hierarchy collapses [1,6]. It would be interesting to investigate whether similar techniques can be used to prove analogous results for ISO-EQN^*_G. Another question left open by the present paper is the existence of a polynomial time reduction from ISO-EQN^*_G to ISO-B-EQN^*_G.

Finally, we note that the strong relation between the complexity of GRAPH ISOMORPHISM and ISO-EQN^*_G for Abelian groups that we give, presents a new approach for settling one of the longest standing open problems in complexity theory.

References

1. M. Agrawal and T. Thierauf. The formula isomorphism problem. *SIAM Journal on Computing,* 30(3):990–1009, 2000.
2. E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for boolean constraint satisfaction. In *Conference for Computer Science Logic,* pages 412–426, 2002.
3. E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. The complexity of boolean constraint isomorphism. In *Proceedings of the 21st Symposium on Theoretical Aspects of Computer Science,* pages 164–175, 2004.
4. K.S. Booth and C.J. Colbourn. Problems polynomially equivalent to graph isomorphism. Technical report, CS-77-04, Computer Science Dept., University of Waterloo, 1979.
5. M. Goldmann and A. Russel. The complexity of solving equations over finite groups. *Information and Computation,* 178(1):253–262, 2002.
6. H. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity.* Birkhäuser Boston, 1993.
7. R. Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters,* 8(3):131–132, 1979.
8. C. Moore, P. Tesson, and D. Thérien. Satisfiability of systems of equations over finite monoids. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science,* pages 537–547, 2001.
9. G. Nordh and P. Jonsson. The complexity of counting solutions to systems of equations over finite semigroups. In *Proceedings of the 10th International Computing and Combinatorics Conference,* 2004.
10. P. Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups.* PhD thesis, School of Computer Science, McGill University, Montreal, 2003.

Generation Problems

Elmar Böhler, Christian Glaßer, Bernhard Schwarz, and Klaus Wagner

Theoretische Informatik
Julius–Maximilians–Universität Würzburg
97074 Würzburg, Germany
{boehler,glasser,schwarzb,wagner}@informatik.uni-wuerzburg.de

Abstract. Given a fixed computable binary operation f, we study the complexity of the following generation problem: The input consists of strings $a_1, \ldots, a_n, b$. The question is whether b is in the closure of $\{a_1, \ldots, a_n\}$ under operation f.

For several subclasses of operations we prove tight upper and lower bounds for the generation problems. For example, we prove exponential-time upper and lower bounds for generation problems of length-monotonic polynomial-time computable operations. Other bounds involve classes like NP and PSPACE.

Here the class of bivariate polynomials with positive coefficients turns out to be the most interesting class of operations. We show that many of the corresponding generation problems belong to NP. However, we do not know this for all of them, e.g., for $x^2 + 2y$ this is an open question. We prove NP-completeness for polynomials $x^a y^b c$ where $a, b, c \geq 1$. Also, we show NP-hardness for polynomials like $x^2 + 2y$. As a by-product we obtain NP-completeness of the extended sum-of-subset problem $\mathrm{SOS}_c = \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\}(\sum_{i \in I} w_i^c = z)\}$ where $c \geq 1$.

1 Introduction

No, this paper is not about problems between generations.[1] However, genealogy presents an example that explains the matter we are interested in. There is hardly any other prehistoric question where scientists grope in the dark as in the following: Are Neandertals completely extinct or are there traces left in some of us? To examine whether a person, e.g., one of the authors, is not a descendant of a Neandertal, one would usually build the whole family tree of the author and check whether every leaf of the tree is labeled with a homo sapiens. This becomes a generation problem in the following way. We go back to the time where Neandertals and homo sapiens still lived segregated from each other. It is well-known that it is the operation of marriage (in a very natural sense) that produces children. We start with this first generation of homo sapiens and apply this operation to obtain their children. Then we apply the marriage operation again and again, until we reach today's people. Now we see whether our author has been generated.

[1] Regardless of the different ages of the authors.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 392–403, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Similar generation problems are for example:

- Does b belong to the closure of $\{a_1, \ldots, a_n\}$ under pairwise addition? This is equivalent to a modification of the sum-of-subset problem where factors other than 0 and 1 are allowed. It can be seen that this is NP-complete.
- Does the empty clause belong to the closure of the clauses $\{\phi_1, \ldots, \phi_n\}$ under the rule of the resolution proof system. This problem is coNP-complete.
- Does a given element of a monoid belong to the submonoid that is generated by a given set?

The complexity of generation problems has been investigated earlier, especially for groups. Generation problems for matrix groups [1,3], for finite groups, where the group operation is given by a multiplication table [4], and for permutation groups [2,5,7] have been examined.

In this paper we investigate sets that are generated by arbitrary computable binary operations. For a fixed such operation we study the complexity of the question:

Does a given string b belong to the set that is generated from strings $\{a_1, \ldots, a_n\}$?

To make this precise, let $\Sigma = \{0, 1\}$ be the alphabet and let f be a computable binary operation on Σ^*, i.e., $f : \Sigma^* \times \Sigma^* \to \Sigma^*$. For $B \subseteq \Sigma^*$, let $[B]_f$ be the f-closure of B, i.e., the smallest set that contains B and that is closed under f. For fixed f we define the generation problem.

Generation problem GEN(f)
INPUT: $a_1, \ldots, a_n, b \in \Sigma^*$
QUESTION: Is b in $[\{a_1, \ldots, a_n\}]_f$?

Equivalently we can use this definition in the context of natural numbers, since these can be identified in the standard way with the words from Σ^*. For convenience we write operations like addition in infix form. The process of generating elements by an iterated application of a binary operation can be visualized by a *generation tree.* A finite tree is called *binary tree,* if every node is either a leaf or has exactly two successors. Let L(T) be the set of leaves, root(T) be the root and Node(T) be the set of nodes of a tree T. Let $B \subseteq \Sigma^*$ be the base set. If f is a binary operation, then a binary tree is called *f-generation tree from B for z* if

- every leaf has a value from B,
- every node that has successors with values x and y has value $f(x, y)$,
- the root of the tree has value z.

Note that $z \in [B]_f$, if and only if there exists an f-generation tree from B for z.

In Section 2, we observe that generation problems for computable operations are recursively enumerable, and there exist associative, commutative, polynomial-time computable operations whose generation problems are many-one complete for recursively enumerable sets. There remain undecidable problems even if we further restrict the operation's resources like time and space. However, we achieve decidability when we demand the operation to be length-monotonic which means that in the generation tree of some x, the lengths of

all intermediate results are bounded by the length of x. If the operations are length-monotonic and polynomial-time computable, all generation problems are solvable in exponential time and there are also such operations for which the generation problem is hard for EXPTIME. We study the complexity of various restrictions of these operations. If additionally the operation is associative, then the corresponding generation problem belongs to PSPACE, and is even PSPACE-complete for suitable operations. If we further restrict the operations to be commutative, then we obtain generation problems that belong to NP, and some of them are even NP-complete (e.g., the usual integer addition).

The most interesting operations we consider in this paper are bivariate polynomials with positive coefficients which are studied in Section 3. Such polynomials are length-monotonic and hence, the corresponding generation problems are decidable. But in general, these polynomials are neither associative nor commutative, hence the generation problem for such polynomials turns out to be non-trivial and exciting. For example, does $\mathrm{GEN}(x^2+2y)$ or $\mathrm{GEN}(x^2y^3)$ belong to NP? If so, are they NP-complete?

There are two main results in the first subsection. For one, we show that if p is not of the form $q(x)+ky$ where q is non-linear and $k \geq 2$, then the generation problem belongs to NP. Besides that, we present a proof of NP-completeness for polynomials of the form $x^a y^b c$ where $a, b, c \geq 1$. Proving hardness is difficult already for such simple polynomials, since we have to cope with the various different trees that generate one number. As a tool to control the shape of generation trees we introduce (a,b)-*weighted trees* which are special trees with additional information. In the proof we force the generation trees into the shape of so-called complete (a,b)-weighted trees.

We do not know whether the generation problem belongs to NP, if the generating polynomial is of the form $q(x)+ky$ where q is non-linear and $k \geq 2$. In this regard, as an upper bound we can easily show that all bivariate polynomials with positive coefficients have generation problems in $\mathrm{NTIME\text{-}SPACE}(2^{\log^2 n}, n \log n)$. Our discussion in the second subsection suggests that this class appears to be a class not far from NP. As a special case of these polynomials, we consider $p(x,y)=x^c+ky$ where $c, k \geq 1$. The main result of this subsection shows that $\mathrm{GEN}(p)$ is NP-hard. Here the operation x^c brings the main difficulty for the proof. We have to find a way to encode information to numbers such that this information is not destroyed by taking the numbers to a high power. This is not easy to solve, since already squaring a number heavily changes its (binary) representation. To overcome this, for $c \geq 1$ we introduce generalized sum-of-subset problems

$$\mathrm{SOS}_c \stackrel{\mathrm{df}}{=} \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\} (\textstyle\sum_{i \in I} w_i^c = z)\}.$$

We show that for all $c \geq 1$, SOS_c is NP-complete and then reduce these problems to $\mathrm{GEN}(p)$. Although all SOS_c are just auxiliary problems in our proof, we feel that this new NP-completeness result is interesting in its own right.

Finally, in Section 4 we summarize the results and give a table that shows a convenient overview of the upper and lower bounds of generation problems.

2 Generation Problems for General Operations

Since we are mostly interested in complexity issues, we restrict ourselves to computable operations. All of the corresponding generation problems are recursively enumerable and we show that there are polynomial-time computable operations whose generation problems are undecidable. There remain undecidable problems even if we further restrict the operation's resources like time and space. The reason is that even with restricted resources it is possible to let a generation problem simulate grammatical derivation trees of arbitrary formal languages. We achieve decidability when we demand the operation to be length-monotonic. Hence we study the complexity of various restrictions of length-monotonic operations.

Theorem 1. $\mathrm{GEN}(\circ)$ *is recursively enumerable for every computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

We observe that polynomial-time computable operations are still too difficult for a complexity-oriented examination of generation problems. For example, with such an operation we can simulate single steps of arbitrary Turing machines.

Theorem 2. *There is an associative, commutative, polynomial-time computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that* $\mathrm{GEN}(\circ)$ *is m-complete for recursively enumerable sets.*

2.1 Length-Monotonic Polynomial-Time Operations

We have seen that in order to get decidable generation problems we have to restrict the class of operations. Therefore, we demand that in the generation tree of some x, the lengths of all intermediate results are bounded by $|x|$, the *length* of x. This is equivalent to say that we restrict to operations $\circ$ that satisfy $|x \circ y| \geq \max(|x|, |y|)$. Call such operations *length-monotonic.* If $|x \circ y| = \max(|x|, |y|)$, then the operation is called *minimal length-monotonic.* Generation trees of such operations can be exhaustively searched by an alternating polynomial-space machine.

Theorem 3. $\mathrm{GEN}(\circ) \in \mathrm{EXPTIME}$ *for every length-monotonic, polynomial-space computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

This exponential-time upper bound for length-monotonic, polynomial-space computable operations is tight, even for polynomial-time computable operations.

Theorem 4. *There is a commutative, minimal length-monotonic, polynomial-time computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that* $\mathrm{GEN}(\circ)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* EXPTIME.

2.2 Length-Monotonic Associative Polynomial-Time Operations

We have seen that in general, commutativity does not lower the complexity of the generation problem for length-monotonic, polynomial-time computable operations. In this subsection we show that associativity does. Here we exploit that for associative operations f we do not need to know the exact structure of an f-generation tree for z: Associativity makes all generation trees with the same sequence of leaves equivalent with respect to the generated element. We show that PSPACE is upper bound for all generation problems with associative, polynomial-space computable operations and that it is lower bound even for associative, polynomial-time computable operations.

Theorem 5. $\mathrm{GEN}(\circ) \in \mathrm{PSPACE}$ *for every length-monotonic and associative polynomial-space computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

The polynomial-space bound is tight even for polynomial-time operations $\circ$.

Theorem 6. *There is a minimal length-monotonic and associative polynomial-time computable operation* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$ *such that* $\mathrm{GEN}(\circ)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* PSPACE.

Now let us additionally assume $\circ$ to be commutative. Again, if we want to know whether or not $z \in [\{x_1, \ldots, x_n\}]_\circ$, associativity enables us to ignore the $\circ$-generation tree and instead search for a word over $\{x_1, \ldots, x_n\}$. Together with commutativity, we just have to guess exponents $k_1, \ldots, k_n$ and test whether $x_1^{k_1} \circ \cdots \circ x_n^{k_n} = z$. The exponentiations are computable in polynomial-time (by squaring and multiplying), which yields the following theorem.

Theorem 7. $\mathrm{GEN}(\circ) \in \mathrm{NP}$ *for all length-monotonic, associative, and commutative polynomial-time computable operations* $\circ : \Sigma^* \times \Sigma^* \to \Sigma^*$.

Again, this upper bound is tight, i.e., there exist associative, commutative, and length-monotonic polynomial-time computable operations whose generation problems are NP-complete. Even the usual addition on natural numbers has this property [8].

Theorem 8. $\mathrm{GEN}(+)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* NP, *where* $+$ *is the addition on* $\mathbb{N}$.

3 Generation Problems for Polynomials

The previous section gave an overview over the complexity of generation problems for polynomial-time computable operations. Now we want to have a look at the more restricted class of generation problems whose operations are polynomials. The Davis-Putnam-Robinson-Matiyasevich theorem [6] states that every recursively enumerable set is range of a polynomial with integer coefficients. Based on this, there are such polynomials where the generation problem is undecidable. To give an idea of this, take a polynomial p with undecidable positive range and replace every variable x by $x_1^2 + x_2^2 + x_3^2 + x_4^2$. Take another polynomial q that is capable to generate all negative numbers. Build a new polynomial out

of p and q with an additional variable y such that for $y = 0$ the value of q is calculated, and for $y \neq 0$ the value of p is calculated. In this way it is possible to generate all negative numbers which in turn allow the generation of the positive range of p. However, to obtain these undecidability results, the polynomials must have negative coefficients and they usually contain a rather large number of variables. Therefore, we concentrate on bivariate polynomials with positive coefficients. These are always length-monotonic and hence, the corresponding generation problem is decidable. We show that many of them are even in NP and all of them belong to NTIME-SPACE($2^{\log^2 n}, n \log n$). So far we have no evidence against the conjecture that all these generation problems belong to NP (see also the discussion in Subsection 3.2). However, we cannot prove this.

3.1 The Main Case

This subsection has two main results: First, we show that if p is not of the form $q(y) + ky$ where q is non-linear and $m \geq 2$, then the corresponding generation problem belongs to NP. Second, we prove NP-completeness for polynomials of the form $x^a y^b c$ where $a, b, c \geq 1$. Let us start our investigation with the obvious case of univariate polynomials p, i.e., $p(x, y) = q(x)$ for a suitable polynomial q.

Theorem 9. *If p is a univariate polynomial, then* GEN(p) *is in P.*

A univariate polynomial $p(x)$ is *linear,* if there are $a, c \in \mathbb{N}$ such that $p(x) = ax + c$.

Theorem 10. *If p is a bivariate polynomial that is not of the form $p(x, y) = kx + q(y)$ or $p(x, y) = q(x) + ky$ where q is non-linear and $k \geq 2$, then* GEN(p) $\in$ NP.

Sketch of proof: We show that p must have one of the following properties:

(1) $p(x, y) = x + q(y)$ or $p(x, y) = q(x) + y$ for some univariate polynomial q,
(2) $p(x, y) = ax + by + c$ for some $a, b, c \in \mathbb{N}$ such that $a, b \geq 2$, and
(3) $p(x, y) \geq x \cdot y$ for all x, y

Assume that the polynomial p has none of the properties (1), (2), and (3). Since p does not fulfill (3) there are univariate polynomials q and r such that $p(x, y) = q(x) + r(y)$. Since $x^2 + y^2 \geq x \cdot y$ at least one of the polynomials q and r is linear. Consequently there exist a univariate polynomial q and an $k \geq 0$ such that $p(x, y) = kx + q(y)$ or $p(x, y) = q(x) + ky$. Since p does not fulfill (2), the polynomial q is not linear. Since p does not fulfill (1), we obtain $k \geq 2$.
If a polynomial p has property (1), it can be proven that

$$[\{a_1, \ldots, a_r\}]_p = \{a_j + \textstyle\sum_{i=1}^{r} \alpha_i \cdot q(a_i) : j \in \{1, \ldots, r\} \text{ and } \alpha_1, \ldots, \alpha_r \in \mathbb{N}\}.$$

This can be decided in NP. For the cases where p is of the form (2) or (3), we have a look at the p-generation tree from a basis B for a destination number z.

These trees can have too large a size to be guessed in polynomial time, but it turns out that in both cases we can overcome this.

If p has form (2), the contribution of each leaf v of a generation tree depends solely on the number of left and right turns on the path from the root to v. It suffices to know these paths in order to retrace the generation of z. That means that we do not have to guess an exact tree, but only how often each tuple $(l, r, b) \in \mathbb{N}^{\leq |z|} \times \mathbb{N}^{\leq |z|} \times B$, appears in the tree.

If p has form (3) then the generation tree has a very special form:

Since every value of a node is multiplied with the value of each other node, almost every node in such a tree is a leaf with a value from $\{0,1\}$ or has exactly one child which is such a leaf. In other words, such a tree is mostly not of a binary form but consists in large parts of long chains of nodes. It turns out, that these chains can in all cases be guessed in one step. This shrinks the size of data to guess to a polynomial size.

So if we consider a polynomial of the form $\sum_{a,b} x^a y^b c_{ab}$ where $a, b \geq 1$, then the generation problem belongs to NP. Here we pick out those polynomials that consist of only one term of the sum. For this special case we can show that $\mathrm{GEN}(x^a y^b c)$ is NP-complete. For $a = 1$ or $b = 1$ this is easy to prove.

Proposition 1. *For $a, c \geq 1$, $\mathrm{GEN}(x^a y c)$ is $\leq_m^p$-complete for* NP.

Now let us consider $\mathrm{GEN}(x^a y^b c)$ for $a, b > 1$. In general, the crucial point in proving hardness for generation problems is to cope with the various different trees that generate a number. In our proofs we force the generation trees to have a specific shape such that the generation is possible only in a predefined way.

Consider an $x^a y^b c$-generation tree. Clearly, the generated number is a product that consists of various multiplicities of c and base elements. As a tool to control these multiplicities we introduce *(a, b)-weighted trees,* where we mark each node with the power its value appears in the root-value. By controlling the marks of the leaves, we can force an $x^a y^b c$-generation tree into the shape of a complete (a, b)-weighted tree.

Definition 1. *Let t be a binary tree. $T = (t, g)$ is called (a, b)-weighted tree, $a, b > 1$, if g is a marking-function $g : \mathrm{Node}(t) \to \mathbb{N}$ such that:*

If $x = \mathrm{root}(t)$, then $g(x) = 1$.
If $x \in \mathrm{Node}(t)$ has a left and a right successor x_l and x_r, then $g(x_l) = a \cdot g(x)$ and $g(x_r) = b \cdot g(x)$.

T is called complete, if $\max_{x \in \mathrm{L}(t)} g(x) < \max(a, b) \cdot \min_{x \in \mathrm{L}(t)} g(x)$.

From this definition it immediately follows that the marks have the desired properties. We obtain the following connection to $\mathrm{GEN}(x^a y^b c)$.

Property 1. Let $a, b > 1$. If $T = (t, g)$ is an (a, b)-weighted tree, where t is an $x^a y^b c$-generation tree with values $I(v)$ for all $v \in \mathrm{L}(t)$, then

$$I(\mathrm{root}(t)) = \prod_{v \in \mathrm{L}(t)} I(v)^{g(v)} \cdot \prod_{v \in \mathrm{Node}(t) - \mathrm{L}(t)} c^{g(v)}.$$

First of all we show that for each $n \geq 1$ there exists a complete (a, b)-weighted tree with nearly n leaves. Furthermore, such a tree is constructable in polynomial time.

Proposition 2. *Let $a, b > 1$. For every $n \geq 1$ there exists a complete (a, b)-weighted tree with at least n and at most $2n - 1$ leaves.*

Proof. For $n = 1$ take the tree that consists only of the root.
For arbitrary $n \geq 1$, let $T = (t, g)$ be a complete (a, b)-weighted tree with at least n and at most $2n - 1$ leaves, given by induction hypothesis. If t has at least $n + 1$ leaves, T proves the claim for $n + 1$. If t has exactly n leaves, let $x_1, \ldots, x_k \in \mathrm{L}(t), 1 \leq k \leq n$ be all leaves with minimal weight, i.e., $g(x_1) = \ldots = g(x_k) = \min_{x \in \mathrm{L}(t)} g(x)$. Define the tree t' by adding in t successors $x_{i,l}$ and $x_{i,r}$ to each leaf x_i, and $g' : \mathrm{Node}(t') \to \mathbb{N}$ by $g'(x) \stackrel{df}{=} g(x)$ for all $x \in \mathrm{Node}(t)$, $g'(x_{i,l}) \stackrel{df}{=} a \cdot g(x_i)$, and $g'(x_{i,r}) \stackrel{df}{=} b \cdot g(x_i)$. This defines an (a, b)-weighted tree $T' \stackrel{df}{=} (t', g')$ with

$$\begin{aligned} \max_{x \in \mathrm{L}(t')} g'(x) &= \max(\max_{x \in \mathrm{L}(t)} g(x), \max\{g'(x_{i,l}), g'(x_{i,r}) : 1 \leq i \leq k\}) \\ &= \max(a, b) \cdot \min_{x \in \mathrm{L}(t)} g(x) < \max(a, b) \cdot \min_{x \in \mathrm{L}(t')} g'(x). \end{aligned}$$

Hence T' is complete, and has $n + k$ leaves, at least $n + 1$ and at most $2n$.

If the generation tree is not the desired complete tree, then at least one leaf-value is taken to a power that is too large.

Proposition 3. *Let $a, b > 1$. Let $T = (t, g)$ be a complete (a, b)-weighted tree with n leaves. If $T' = (t', g')$ is an (a, b)-weighted tree with more than n leaves, then there exists a leaf $y \in \mathrm{L}(t')$ such that $g'(y) > \max_{x \in \mathrm{L}(t)} g(x)$.*

Complete (a, b)-weighted trees have a height which is bounded logarithmically in the number of leaves.

Proposition 4. *Let $a \geq b > 1$. Let $T = (t, g)$ be a complete (a, b)-weighted tree with n leaves. If d denotes the maximal depth of a leaf of t, then*

$$d < \log_b(a) \cdot (1 + \log_2(n)).$$

Theorem 11. *For $a, b, c \geq 1$, $\mathrm{GEN}(x^a y^b c)$ is $\leq_m^p$-complete for* NP.

Proof. By Proposition 1, we can assume $a, b > 1$. Containment in NP follows from Theorem 10. We reduce 1-IN-3-SAT to $\mathrm{GEN}(x^a y^b c)$. Let H be a 3-CNF formula with clauses $C_1, \ldots, C_m$ and variables $x_1, \ldots, x_n$. Let $p_1, p_2, \ldots$ be the prime numbers larger than c. Define $a_i \stackrel{df}{=} p_{m+i} \prod_{x_i \in C_j} p_j$ and $b_i \stackrel{df}{=} p_{m+i} \prod_{\overline{x_i} \in C_j} p_j$. Let $T = (t, g)$ be a complete (a, b)-weighted tree with k leaves where $n \leq k \leq 2n - 1$ and $\mathrm{L}(t) = \{v_1, \ldots, v_k\}$ (such a tree exists by Proposition 2). Furthermore, let d be the maximal depth of a leaf of t. Define $a_i \stackrel{df}{=} p_{m+i}$ for $i = n + 1, \ldots, k$,

$$B \stackrel{df}{=} \left\{a_i' = a_i^{a^d b^d / g(v_i)} : 1 \leq i \leq k\right\} \cup \left\{b_i' = b_i^{a^d b^d / g(v_i)} : 1 \leq i \leq n\right\}, \text{ and}$$

$$z \stackrel{df}{=} \prod_{i=1}^{m+k} p_i^{a^d b^d} \cdot \prod_{v \in \mathrm{Node}(t) - \mathrm{L}(t)} c^{g(v)}.$$

Proposition 4 shows that (B, z) is polynomial-time computable.

If $H \in$ 1-IN-3-SAT, then there is an assignment $I_H : \{x_1, \ldots, x_n\} \to \{0,1\}$ that satisfies exactly one literal in each clause. We consider t as an $x^a y^b c$-generation tree with values

$$I_t(v_i) \stackrel{\text{df}}{=} \begin{cases} a_i' & \text{, if } i = 1, \ldots, n \text{ and } I_H(x_i) = 1 \\ b_i' & \text{, if } i = 1, \ldots, n \text{ and } I_H(x_i) = 0 \\ a_i' & \text{, if } i = n+1, \ldots, k. \end{cases}$$

Using Property 1 we observe $I_t(\text{root}(t)) = z$. Hence $(B, z) \in \text{GEN}(x^a y^b c)$.

Assume $(B, z) \in \text{GEN}(x^a y^b c)$. So there exists an (a, b)-weighted tree $T' = (t', g')$, where t' is an $x^a y^b c$-generation tree from B for z. For each $v \in \text{Node}(t')$ define $I_{t'}(v)$ as the value of node v. Each element of B has exactly one prime factor from $p_{m+1}, \ldots, p_{m+k}$. Since z has all these prime factors at least once, t' must have at least k leaves. Assume t' has more than k leaves. By Proposition 3, there exists $v \in \text{L}(t')$ such that $g'(v) > \max_{x \in \text{L}(t)} g(x)$. $I_{t'}(v)$ has exactly one prime factor from $p_{m+1}, \ldots, p_{m+k}$; say p_{m+i} with exponent $a^d b^d / g(v_i)$. Hence $p_{m+i}^{a^d b^d / g(v_i) \cdot g'(v)}$ is a factor of $I_{t'}(\text{root}(t'))$. From $a^d b^d / g(v_i) \cdot g'(v) > a^d b^d$ it follows that $I_{t'}(\text{root}(t')) \neq z$.
So t' has exactly k leaves. Each prime $p_{m+1}, \ldots, p_{m+k}$ must appear as factor in a value of some leaf. Therefore, for $i = 1, \ldots, n$, either a_i' or b_i' is a value of a leaf (but not both). Define $I_H : \{x_1, \ldots, x_n\} \to \{0,1\}$ such that $I_H(x_i) = 1 \Leftrightarrow_{\text{df}} a_i'$ is a leaf-value of t'. Observe that I_H shows $H \in$ 1-IN-3-SAT.

3.2 The Case $q(x) + ky$, Where q Is Non-linear and $k \geq 2$

So far we do not have upper bounds for generation problems of polynomials $p(x, y) = q(x) + ky$, where q is non-linear and $k \geq 2$. The obvious algorithm guesses and verifies generation trees. How large are these trees? To answer this, observe that the trees are of a special form: When we go from the root to the leaves in y-direction, then in each step, the length of the value decreases by one bit. When we go in x-direction, then in each step, the length is bisected. It follows that the size of such trees can grow faster than any polynomial, but not as fast as $2^{\log^2 n}$. Therefore, $\text{GEN}(p) \in \text{NTIME}(2^{\log^2 n})$. We do not have to guess complete generation trees. If a subtree generates some value b, then it suffices to store b instead of the whole subtree. We need to store a value b every time we go in x-direction. So we need space $O(n \log n)$.

Proposition 5. $\text{GEN}(p) \in \text{NTIME-SPACE}(2^{\log^2 n}, n \log n)$ *for all* $p(x, y) = q(x) + ky$ *where* $k \geq 2$ *and* q *is a non-linear polynomial.*

Because of the special form of a generation tree for such polynomials, the generation problem can be solved by special alternating machines: Some z can be generated via p from A if and only if there exist $z_1, \ldots, z_n \leq z$ such that $n \leq |z|$, $z = z_1$, $z_n \in A$, and for all $1 \leq i < n$, $z_i = p(z_{i+1}, y_i)$ where y_i can be generated via p from A and $|y_i| \leq \frac{1}{2}|z_i|$. An alternating machine can check this predicate

in polynomial time with a logarithmic number of alternations. Furthermore, in existential parts the machine guesses polynomially many bits. In contrast, in universal parts it guesses logarithmically many bits.

This discussion shows that GEN(p) can be solved with quite restricted resources. However, we do not know whether GEN(p) belongs to NP. Standard diagonalizations show that there exist oracles A and B such that $\mathrm{BPP}^A \not\subseteq \mathrm{NTIME}(2^{\log^2 n})^A$ and $\mathrm{coNP}^B \not\subseteq \mathrm{NTIME}(2^{\log^2 n})^B$. Therefore, we should not expect GEN(p) to be hard for any class that contains BPP or coNP. This rules out many reasonable classes above NP to be reducible to GEN(p). We consider this as a hint that GEN(p) could be contained in NP, but we do not have a proof for this. We leave this as an open question.

Nevertheless, in this subsection we prove lower bounds. The main result, Theorem 13, shows that if $p(x, y) = x^c + ky$ where $c, k \geq 1$, then GEN(p) is $\leq_m^p$-hard for NP. The proof is difficult for two reasons which we want to explain for $p(x, y) = x^2 + 2y$.

1. We have to encode NP-computations into generation problems. For this, we need to construct an instance (B, z) of GEN(p) that represents information about a given NP-computation. The elements of B must be chosen in a way so that squaring will not destroy this information. This is difficult, since squaring a number heavily changes its (binary) representation.
2. We construct (B, z) such that if z can be generated, then any time, x must be chosen from B (and is not a generated number). So the generation tree is linear. Because of the factor 2, in any step, the number generated so far is shifted to the left. We have to cope with this shifting.

With regard to item 2, our construction makes sure that the size of the linear generation tree is bounded. So the number of shifts is bounded. For B we choose numbers that are much longer than this bound such that each number is provided with a unique stamp. The stamps make sure that there is at most one possible tree that generates z. In particular, this fixes the sequence of numbers from B that are chosen for x. This keeps the shifting under control.

The problem in item 1 is more complicated and also more interesting. It comes down to prove NP-hardness of the following extended sum-of-subset problem.

$$\mathrm{SOS}_2 \stackrel{df}{=} \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\} (\textstyle\sum_{i \in I} w_i^2 = z)\}$$

(In the proof we use a promise problem related to SOS_2, but for simplicity we argue with SOS_2 in this sketch.) First we reduce 1-IN-3-SAT to SOS and obtain an SOS instance $w = (w_1, \ldots, w_{2n}, z)$. The reduction is such that either $w \notin$ SOS or there is a selection of exactly n weights whose squares sum up to z. We choose a base b larger than z and all w_i^2. So in the system to base b, z and all w_i^2 fit into one digit. For each w_i, define the following 6-digit number in the system to base b.

$$a_i \stackrel{df}{=} [11000 w_i]_b$$
$$r_i \stackrel{df}{=} [10001 w_i]_b$$

The set of all a_i and all r_i build the weights for the SOS_2 instance we want to construct. The intention is to use the weight a_i whenever w_i is used in the sum that yields z, and to use r_i whenever w_i is not used. The squares of a_i and r_i look as follows with respect to base b.

$$a_i^2 \stackrel{df}{=} [\ 1 \quad 2\ 1\ 0\ 0\ 2w_i\ \ 2w_i\ \ 0\ \ 0 \quad\ 0 \quad\ w_i^2]_b$$
$$r_i^2 \stackrel{df}{=} [\ 1 \quad 0\ 0\ 0\ 2\ 2w_i \quad\ 0 \quad\ 0\ 1\ 2w_i\ \ w_i^2]_b$$

Note that a_i^2 and r_i^2 have the same first digit, the same last digit, and the same digit at the middle position. At all other positions, either a_i^2 or r_i^2 has digit 0. In the sum for SOS_2, for every i, either a_i or r_i is used. Therefore, in system b, the last digit of this sum becomes predictable: It must be $\sum_i w_i^2$. This is the most important point in our argumentation. Also, we choose exactly n weights a_i and n weights r_i. With $s_1 \stackrel{df}{=} \sum_i w_i$, $s_2 \stackrel{df}{=} \sum_i w_i^2$, and $\overline{z} \stackrel{df}{=} s_1 - z$ we can easily describe the destination number for the SOS_2 instance.

$$z' \stackrel{df}{=} [2n\ \ 2n\ \ n\ \ 0\ \ 2n\ \ 2s_1 \quad 2z \quad\ 0\ \ n\ \ 2\overline{z} \quad\ s_2]_b$$

We obtain the instance $(a_1, r_1, \ldots, a_{2n}, r_{2n}, z')$ which belongs to the SOS_2 if and only if $(w_1, \ldots, w_{2n}, z) \in \mathrm{SOS}$. This shows NP-hardness for SOS_2 and solves the difficulty mentioned in item 2.

We inductively use this technique to show that for all $c \geq 1$, the following extended sum-of-subset problem is NP-complete.

$$\mathrm{SOS}_c \stackrel{df}{=} \{(w_1, \ldots, w_n, z) : \exists I \subseteq \{1, \ldots, n\} (\textstyle\sum_{i \in I} w_i^c = z)\}.$$

Theorem 12. *For $c \geq 1$, the problem SOS_c is $\leq_m^p$-complete for* NP.

We use this result for SOS_c to prove the main theorem of this subsection.

Theorem 13. *For $c, k \geq 1$, $\mathrm{GEN}(x^c + ky)$ is $\leq_m^p$-hard for* NP.

4 Conclusion and Open Questions

We summarize our results on the complexity of $\mathrm{GEN}(f)$ in the following table. Every lower bound is given by the fact that there exists an f from the considered class of operations whose generation problem is complete for the respective class. All operations are polynomial-time computable.

The gap between NP and $\mathrm{NTIME}(2^{\log^2 n})$ in the last rows of the table below calls the attention to an interesting open question: Does $\mathrm{GEN}(q(x) + ky)$ belong to NP if q is non-linear and $k \geq 2$? Since the generation trees for these polynomials may be of superpolynomial size, the obvious algorithm of guessing and verifying the tree is not applicable. Also, we could not find compacter representations as in Theorem 10. There are generation trees where almost all nodes take different values. Therefore it may be possible that we really have to calculate all

of them. Perhaps there are special polynomials of the form $q(x) + ky$ for which the closure is very regular, as in Theorem 10, case (1)? Another possibility to solve the problem could be to have a closer look at the restricted alternating machines we desribe in subsection 3.2. What are the exact capabilities of these machines?

operation	lower bound	Thm	upper bound	Thm
arbitrary	recursively enumerable	2	recursively enumerable	1
length-monotonic	EXPTIME	4	EXPTIME	3
length-monotonic and commutative	EXPTIME	4	EXPTIME	3
length-monotonic and associative	PSPACE	6	PSPACE	5
length-mon., assoc., and commutative	NP	8	NP	7
all Polynomials $\neq q(x) + ky$	NP	8	NP	10
$x + y$	NP	8	NP	7
$x \cdot y$	NP	1	NP	7
$x^a y^b c$	NP	11	NP	10
all Polynomials $= q(x) + ky$	NP	13	NTIME($2^{\log^2 n}$)	5
$x^c + ky$	NP	13	NTIME($2^{\log^2 n}$)	5

References

1. L Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing,* pages 421–429. ACM Press, 1985.
2. L. Babai, E. Luks, and A. Seress. Permutation groups in NC. In *Proceedings of the nineteenth annual ACM conference on Theory of computing,* pages 409–420. ACM Press, 1987.
3. L Babai and E. Szemerédi. On the complexity of matrix group problems. In *25th Annual Symposium on Foundations of Computer Science,* pages 229–240, 1984.
4. D. M. Barrington, P. Kadau, K. Lange, and P. McKenzie. On the complexity of some problems on groups input as multiplication tables. *Journal of Computer and System Sciences,* 63, 2001.
5. M. Furst, J. Hopcroft, and E. Luks. Polynomial time algorithms for permutation groups. In *21th Annual Symposium on Foundations of Computer Science,* pages 36–41, 1984.
6. Y. V. Matiyasevich. Enumerable sets are diophantine. *Doklady Akad. Nauk SSSR,* 191:279–282, 1970. Translation in Soviet Math. Doklady, 11:354–357, 1970.
7. C. C. Sims. Computational methods in the study of permutation groups. In J. Leech, editor, *Computational problems in abstract algebra,* Proc. Conf. Oxford 1967, pages 169–183, London, 1970. Pergamon.
8. P. van Emde Boas. Complexity of linear problems, pages 117–120, 1979.

One Query Reducibilities Between Partial Information Classes

Sebastian Bab and Arfst Nickelsen

Technische Universität Berlin, Germany
{smarcbab,nicke}@cs.tu-berlin.de

Abstract. A partial information algorithm for A computes for input words $(x_1, \ldots, x_m)$ a set of bitstrings containing $\chi_A(x_1, \ldots, x_m)$. For a family $\mathcal{D}$ of sets of bitstrings of length m, $A \in \mathrm{P}[\mathcal{D}]$ if there is a polynomial time partial information algorithm that always outputs a set from $\mathcal{D}$. For the case $m = 2$ we investigate whether for families $\mathcal{D}_1$ and $\mathcal{D}_2$ the languages in $\mathrm{P}[\mathcal{D}_1]$ are reducible to languages in $\mathrm{P}^X[\mathcal{D}_2]$ for some X in PH or in EXP. Beigel, Fortnow and Pavan, and Tantau already achieved some results in this respect. We achieve results for all remaining non-trivial pairs of classes $\mathrm{P}[\mathcal{D}]$ for $m = 2$. We show: 1. 2-cheatable languages are 1-tt Δ_2^p-reducible to languages in $\Delta_2^p[\mathrm{MIN}_2]$. 2. Languages in $\mathrm{P}[\mathrm{SEL}_2 \cup \{\mathrm{xor}_2\}]$ are 1-tt Δ_2^p-reducible to some Δ_2^p-selective languages. 3. 2-countable languages are 1-tt Δ_3^p-reducible to Δ_3^p strongly 2-membership comparable languages. 4. 2-membership comparable languages are 1-tt EXP-reducible to EXP strongly 2-membership comparable languages. 5. There are easily 2-countable languages not Turing reducible to languages in $\mathrm{EXP}[\mathrm{BOTTOM}_2]$. 6. There are languages in $\mathrm{P}[\mathrm{BOTTOM}_2]$ not 1-tt EXP-reducible to EXP-selective languages.

1 Introduction

If a language A is not decidable in polynomial time, there may still exist a *partial information* algorithm. For a tuple of input words $(x_1, \ldots, x_m)$ such an algorithm computes some partial information on membership of these words in A. More precisely, it computes a set D of possible values for $\chi_A(x_1, \ldots, x_m)$, where χ_A is the characteristic function for A. Such a set $D \subseteq \{0,1\}^m$ is called an m-pool, the set $\mathcal{D}$ of m-pools that may occur as outputs of a specific partial information algorithm is called an m-family. Since we can also consider other resource bounds than polynomial time, three parameters are of interest:

- The number m of input words for the algorithm.
- The type of partial information represented by an m-family.
- The computational resources of the algorithm.

Many types of partial information have been studied, most prominently p-selectivity (due to [25], see [14]), cheatability (due to [5]), and membership comparability (due to [1,11], also known as approximability or non-superterseness, see [9,24]). Other notions are verboseness, strong membership comparability,

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 404–415, 2004.
© Springer-Verlag Berlin Heidelberg 2004

frequency computations, easily countable languages, multi-selectivity and sortability (for detailed definitions see e.g. [1,2,8,9,13,15,16]). The recursion theoretic notion of $\mathcal{D}$-*verboseness* was introduced in [7], where the type of partial information is specified by a family $\mathcal{D}$ of pools. A general theory for $\mathcal{D}$-verboseness was developed in [19,20]. For a recent survey on partial information see [23].

Reduction closures (reductions are always meant to be polynomial time, if not stated otherwise) of partial information classes have been the focus of much interest. Firstly because the class P/poly is the reduction closure of P-sel, the class of p-selective sets [25,17,1] (1. in the list below). Secondly, one can prove results of the following type: If NP is in the reduction closure of some $\mathrm{P}[\mathcal{D}]$ under a suitable reduction, then a collapse of complexity classes occurs (the best result in this respect is 2. [9,24]). Several other results are known (3. to 6.):

1. The closure of P-sel under Turing reductions is P/poly. Since $\mathrm{P}[\mathcal{D}] \subseteq \mathrm{P/poly}$ for nontrivial families $\mathcal{D}$, all languages in $\mathrm{P}[\mathcal{D}]$ are Turing reducible to languages in P-sel.[25,17,1]
2. If, for some m and $\epsilon > 0$, NP is in the closure of the m-approximable languages under $n^{\frac{1}{m-1}-\epsilon}$-tt reductions, then P = NP.[9,24]
3. P-sel is closed under positive Turing reductions [10].
4. The class of m-cheatable languages is closed under Turing reductions [5]. The classes of m-cheatable languages are the only non-trivial classes $\mathrm{P}[\mathcal{D}]$ that are closed under 2-tt reductions [22].
5. The reduction closures of P-sel under k-tt reductions for growing k form a proper hierarchy[12].
6. $\mathrm{P}[\mathcal{D}]$ is closed under 1-tt reductions iff $\mathcal{D}$ is closed under bit-flip [20].

If we want to compare partial information classes for m-families $\mathcal{D}_1, \mathcal{D}_2$ in terms of a reducibility $\leqslant_r$, two questions naturally arise ($R_r(\mathcal{C})$ denotes the closure of a class $\mathcal{C}$ under $\leqslant_r$-reductions):

(a) Is $R_r(\mathrm{P}[\mathcal{D}_1]) \subseteq \mathrm{P}[\mathcal{D}_2]$? (b) Is $\mathrm{P}[\mathcal{D}_1] \subseteq R_r(\mathrm{P}[\mathcal{D}_2])$?

Problem (a) is quite well understood. For bounded query reductions it completely reduces to a combinatorial property of the m-families involved ([26], see [22]):

7. For $m, k \geqslant 1$, an mk-family $\mathcal{D}_1$ and an m-family $\mathcal{D}_2$ (both in normal form): $R^p_{k\text{-tt}}(\mathrm{P}[\mathcal{D}_1]) \subseteq \mathrm{P}[\mathcal{D}_2]$ iff $(\phi_1 \times \cdots \times \phi_m)(D) \in \mathcal{D}_2$ for all $D \in \mathcal{D}_1$ and all k-ary boolean functions $\phi_1, \ldots, \phi_m$.

Not so much has been known about Problem (b), even for the restricted (and maybe most interesting) case of the ten nontrivial partial information classes for $m = 2$. Known results were either trivial, or followed from the fact that some classes are closed under certain reductions or that some classes contain undecidable languages while others do not. The situation changed when, in 2002, [6] came up with two new results for $m = 2$, a negative and a positive one. For the negative result, a 2-membership comparable language A was constructed that is not bounded truth table (btt) reducible to any p-selective set. Inspection of the diagonalization proof given shows that A even is in $\mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$. For the positive result, it was proved that, if P = NP, every 2-cheatable language

is 1-tt reducible to a p-selective set. The assumption P = NP is needed because the reduction uses a Σ_2^p-oracle. Hence the results of [6] can be stated as:

8. There exists an $A \in \mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$ that is not btt reducible to any p-selective language.
9. Every 2-cheatable language is 1-tt Δ_3^p-reducible to a p-selective language.

These results motivated us to develop similar negative or positive results for all remaining pairs of 2-families. Our positive results show the existence of 1-tt reductions. In three cases these reductions require polynomial time with access to an oracle from PH, the polynomial hierarchy. Our results have a drawback compared to [6] insofar as our partial information algorithm must have access to the same oracle as the reduction. In the fourth case the 1-tt reduction and the partial information algorithm constructed even need access to an EXP-oracle.

Our first negative result says that there is an easily 2-countable language that is not reducible to a $B \in \mathrm{EXP}[\mathrm{BOTTOM}_2]$. The second says that there is an $A \in \mathrm{P}[\mathrm{BOTTOM}_2]$ that is not 1-tt EXP-reducible to EXP-selective languages. Note that we allow *exponential time* as resource bound for partial information algorithms for the language we reduce to. We do so, since our positive results require PH-oracles or exponential time, and we want to be on the safe side so that in the negative case such reductions are impossible. We could prove our negative results for even larger resource bounds, but that would give no additional insight. Also note that the negative results are for different reduction types (1-tt and Turing) which reflects the completely different proof techniques.

In *Section 2* we give definitions and basic facts concerning partial information classes. In *Section 3* we study positive results for reductions between partial information classes. Negative results are given in *Section 4*. In *Section 5* we present a figure that gives an overview of the results obtained. Furthermore we give an outlook on open problems and possible future work. Some of the results of this paper already appeared in [3]. Due to lack of space, some of the proofs have to be omitted in this paper. These proofs are available in [4].

2 Basics on Partial Information Classes

Languages, Bitstrings. Languages are subsets of $\Sigma^* = \{0,1\}^*$. The *characteristic function* $\chi_A\colon \Sigma^* \to \{0,1\}$ for a language A is defined by $\chi_A(x) = 1 \Leftrightarrow x \in A$. We extend χ_A to tuples by $\chi_A(x_1,\ldots,x_m) := \chi_A(x_1)\cdots\chi_A(x_m)$. $\overline{A}$ denotes the complement of A. $\#_1(b)$ denoted the number of 1's in a bitstring b, $b[i]$ is the i-th bit of b, and $b[i_1,\ldots,i_k] := b[i_1]\cdots b[i_k]$. We extend this to sets of bitstrings by $D[i_1,\ldots,i_k] := \{b[i_1,\ldots,i_k] \mid b \in D\}$. We use the standard ordering on Σ^* (denoted $x < y$ if x precedes y), i.e. we order by length first, words of the same length are ordered lexicographically.

Complexity Classes. Languages in EXP are decidable by deterministic Turing machines with time bound $O(2^{n^k})$ for some $k \in \mathbb{N}$. Function classes are distinguished by the prefix 'F'. The levels of the Polynomial Hierarchy (PH) are

defined inductively with deterministic and nondeterministic oracle Turing machines: $\Sigma_1^p := \mathrm{NP}$, $\Sigma_{i+1}^p := \mathrm{NP}^{\Sigma_i^p}$, $\Delta_{i+1}^p := \mathrm{P}^{\Sigma_i^p}$. We make use of the characterization of the levels of PH by formulas with alternating quantifiers. We write for short $\exists^{p(n)} y.\ \phi(x,y)$ instead of $\exists y.\ |y| \leqslant p(n) \wedge \phi(x,y)$, and $\exists^{\leqslant x} y.\ \phi(x,y)$ instead of $\exists y.\ y \leqslant x \wedge \phi(x,y)$; and similarly for $\forall$-quantifiers.

Reductions. We distinguish polynomial time adaptive (or Turing) reductions and non-adaptive (or truth table) reductions (in most cases with a bounded number of queries) defined as usual. Moreover, to increase their computational power, our reducing machines have additional (adaptive) access to a further oracle. E.g., a language A is 1-tt Δ_i^p- or 1-tt EXP-reducible if A is 1-tt reducible to B and (besides B) the reducing machine has access to a Δ_i^p- or EXP-complete set.

Partial information classes. We now introduce to the concept of partial information classes. We state facts from [21].

Definition 1 (Pool, family, partial information class).
Let $m \geqslant 1$. A subset $D \subseteq \{0,1\}^m$ is called an m-pool.

1. *A set $\mathcal{D} = \{D_1, \ldots, D_r\}$ of m-pools is called an* m-family *if*
   a) *$\mathcal{D}$ covers $\{0,1\}^m$, that is $\bigcup_{i=1}^r D_i = \{0,1\}^m$, and*
   b) *$\mathcal{D}$ is closed under subsets, that is $D_1 \in \mathcal{D}$ and $D_2 \subseteq D_1$ implies $D_2 \in \mathcal{D}$.*
2. *For an m-family $\mathcal{D}$, a language A is in $\mathrm{P}[\mathcal{D}]$ iff there is an $f \in \mathrm{FP}$ such that $f(x_1, \ldots, x_m) \in \mathcal{D}$ and $\chi_A(x_1, \ldots, x_m) \in f(x_1, \ldots, x_m)$ for all $x_1, \ldots, x_m$.*

This definition can easily be applied to other resource bounds like deterministic exponential time or the Δ_i-levels of PH, yielding classes $\mathrm{EXP}[\mathcal{D}]$ or $\Delta_i^p[\mathcal{D}]$.

Definition 2 (Operations on Bitstrings, Normal Form).

1. *For $b \in \{0,1\}^m$ and a permutation σ on $\{1, \ldots, m\}$ we define $\sigma(b) := b[\sigma(1)] \cdots b[\sigma(m)]$.*
2. *For $b \in \{0,1\}^m$, $i \in \{1, \ldots, m\}$ and $c \in \{0,1\}$ we define* projections π_i^c *by $\pi_i^c(b) := b[1] \cdots b[i-1]\, c\, b[i+1] \cdots b[m]$.*
3. *For $b \in \{0,1\}^m$, $i, j \in \{1, \ldots, m\}$ we define* replacement operations $\rho_{i,j}$ *by $\rho_{i,j}(b) := b'$ where $b'[k] := b[k]$ for $k \neq j$ and $b'[j] := b[i]$.*
4. *For $b \in \{0,1\}^m$ and $i \in \{1, \ldots, m\}$ we define* bit-flip operations flip_i *by $\mathrm{flip}_i(b) := b[1] \cdots b[i-1](1 - b[i]) b[i+1] \cdots b[m]$.*
   Multiple bit-flip operations on positions $i_1, \ldots, i_m$ are denoted $\mathrm{flip}_{i_1,\ldots,i_m}(b)$.
5. *We extend these operations to pools by $\omega(D) := \{\omega(b) \mid b \in D\}$ and to families by $\omega(\mathcal{D}) := \{\omega(D) \mid D \in \mathcal{D}\}$.*
6. *An m-family $\mathcal{D}$ is said to be* closed *under an operation ω if $\omega(\mathcal{D}) \subseteq \mathcal{D}$.*
7. *An m-family $\mathcal{D}$ is in* normal form *if it is closed under all permutations, projections, and replacements defined on strings of length m.*

Families have unique normal forms (Fact 1.1). The inclusion of classes corresponds to the inclusion of families (Fact 1.2). An inspection of the 'only if' part of the proof of Fact 1.2 in [21] shows that one can construct (by diagonalization) languages in $\mathrm{P}[\mathcal{E}]$ that even have no exponential time partial information algorithm yielding pools from $\mathcal{D}$. This yields Fact 1.3.

Fact 1 (Normal Form, Inclusion of Classes and Families, Hierarchy).

1. *For every m-family $\mathcal{D}$ there is a unique m-family $\mathcal{D}'$ in normal form with* $\mathrm{P}[\mathcal{D}] = \mathrm{P}[\mathcal{D}']$.
2. *For all m-families $\mathcal{D}$ and $\mathcal{E}$ in normal form we have* $\mathrm{P}[\mathcal{D}] \subseteq \mathrm{P}[\mathcal{E}]$ *iff* $\mathcal{D} \subseteq \mathcal{E}$.
3. *Let $\mathcal{D}$ and $\mathcal{E}$ be m-families in normal form such that $\mathcal{E} \not\subseteq \mathcal{D}$. Then there are languages in* $\mathrm{P}[\mathcal{E}] \setminus \mathrm{EXP}[\mathcal{D}]$.

Upward translation describes a method to 'translate' m-families to l-families for $l > m$ (Definition 3.1) which preserves normal forms and classes defined (Facts 2.1 and 2.2). A set of m-pools generates an m-family in normal form (Definition 3.2). Bit-flip closure (Definition 3.3) implies 1-tt-closure (Fact 2.3).

Definition 3 (Upward Translation, Generated Family, Closure under bit-flip operations).

1. *Let $\mathcal{D}$ be an m-family and $m < l$. The l-family* $\lceil \mathcal{D} \rceil_l := \{D \subseteq \{0,1\}^l \mid \forall i_1 < \cdots < i_m \colon D[i_1, \ldots, i_m] \in \mathcal{D}\}$ *is called the* upward translation of $\mathcal{D}$ to l.
2. For a set of m-pools $\mathcal{D}$ define $\langle \mathcal{D} \rangle$, the family generated by $\mathcal{D}$, as the minimal m-family $\mathcal{D}'$ in normal form with $\mathcal{D} \subseteq \mathcal{D}'$.
3. For a set of m-pools $\mathcal{D}$ define $\langle \mathcal{D} \rangle_{\mathrm{flip}}$, the bit-flip closure of $\mathcal{D}$, as the minimal family $\mathcal{D}'$ in normal form with $\mathcal{D} \subseteq \mathcal{D}'$ which is closed under bit-flip operations flip_i for every $i \in \{1, \ldots, m\}$.

Fact 2 (Upward Translation, Bit-flip closure implies 1-tt-closure). *For an m-family $\mathcal{D}$ and $m < l$ we have:*

1. *If $\mathcal{D}$ is in normal form, then $\lceil \mathcal{D} \rceil_l$ is in normal form.*
2. $\mathrm{P}[\mathcal{D}] = \mathrm{P}\left[\lceil \mathcal{D} \rceil_l\right]$, $\mathrm{EXP}[\mathcal{D}] = \mathrm{EXP}\left[\lceil \mathcal{D} \rceil_l\right]$, $\Delta_i^p[\mathcal{D}] = \Delta_i^p\left[\lceil \mathcal{D} \rceil_l\right]$ *for all* $i \in \mathbb{N}$.
3. *The classes $\mathrm{P}[\mathcal{D}]$ resp. $\mathrm{EXP}[\mathcal{D}]$ are closed under 1-tt polynomial time resp. exponential time reductions iff $\mathcal{D}$ is closed under bit-flip.*

We next give names and informal descriptions to crucial 2-pools. Then we define all 2-families in normal form as families generated by these pools.

Definition 4.

1. equ_2	$:= \{00, 11\}$	*The words are equivalent wrt. membership in A.*
2. xor_2	$:= \{01, 10\}$	*Exactly one of the words is in A.*
3. sel_2	$:= \{00, 01, 11\}$	*If the first word is in A, then also the second.*
4. co-sel_2	$:= \{00, 10, 11\}$	*If the second word is in A, then also the first.*
5. bottom_2	$:= \{00, 01, 10\}$	*At most one word is in A.*
6. top_2	$:= \{01, 10, 11\}$	*At least one word is in A.*

Definition 5.

1. k-$\textsc{Size}_2$	$:= \{D \mid \lvert D \rvert \leqslant k\}$	6. $\textsc{Sel}_2$	$:= \langle \mathrm{sel}_2 \rangle$
2. $\textsc{Min}_2$	$:= \langle \mathrm{equ}_2 \rangle$	7. $\textsc{Sel}_2 \cup \{\mathrm{xor}_2\}$	$:= \langle \mathrm{sel}_2, \mathrm{xor}_2 \rangle$
3. $\textsc{Bottom}_2$	$:= \langle \mathrm{bottom}_2 \rangle$	8. $\textsc{Sel}_2 \cup \textsc{Bottom}_2$	$:= \langle \mathrm{sel}_2, \mathrm{bottom}_2 \rangle$
4. $\textsc{Top}_2$	$:= \langle \mathrm{top}_2 \rangle$	9. $\textsc{Sel}_2 \cup \textsc{Top}_2$	$:= \langle \mathrm{sel}_2, \mathrm{top}_2 \rangle$
5. 2-$\textsc{Card}_2$	$:= \langle \mathrm{bottom}_2, \mathrm{top}_2 \rangle$		

We call languages in $\mathrm{P}[2\text{-}\mathrm{SIZE}_2]$ 2-cheatable; in [5] they are called 1-cheatable. $\mathrm{P}[\mathrm{SEL}_2]$ equals the class of p-selective sets. Languages in $\mathrm{P}[2\text{-}\mathrm{CARD}_2]$ are called easily 2-countable. Languages in $\mathrm{P}[3\text{-}\mathrm{SIZE}_2]$ are called 2-approximable (or 2-membership comparable). The class $\mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$ contains the strongly 2-membership comparable languages; in [6] they are called 'tree-selective'. Languages in $\mathrm{P}[\mathrm{TOP}_2]$ are complements of languages in $\mathrm{P}[\mathrm{BOTTOM}_2]$; $\mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{TOP}_2]$ contains the complements of languages in $\mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$. Figure 1 shows the inclusion hierarchy of the partial information classes over 2-families.

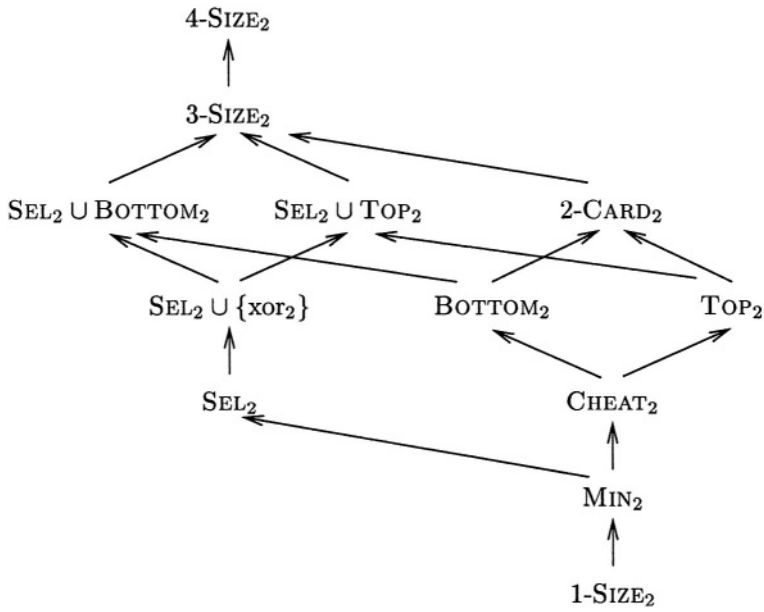


Fig. 1. The lattice of the 12 families for $m = 2$ in normal form.

3 Positive Results

We first present positive results, which means that we prove the existence of reducibilities between pairs of partial information classes. We restrict ourselves to 1-tt reducibilities. We first list the positive results that are already known:

Fact 3 (Known Positive Results).

1. $A \in \mathrm{P}[\mathrm{TOP}_2]$ *iff* $\overline{A} \in \mathrm{P}[\mathrm{BOTTOM}_2]$.
2. $A \in \mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{TOP}_2]$ *iff* $\overline{A} \in \mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$.
3. *Every* $A \in \mathrm{P}[\mathrm{CHEAT}_2]$ *is* 1-*tt* Δ_3^p-*reducible to some* $B \in \mathrm{P}[\mathrm{SEL}_2]$. *[6]*
4. *Every* $A \in \mathrm{P}[\mathrm{CHEAT}_2]$ *is* 1-*tt* Δ_3^p-*reducible to some* $B \in \Delta_3^p[\mathrm{MIN}_2]$. *[26]*

For the two pairs of 2-families for which one class contains the complements of languages of the other class there obiviously is a trivial mutual 1-tt reducibility. We will improve Fact 3.4 to Δ_2^p-reductions in Corollary 1. The following new theorem shows how we can reduce from a class $\mathrm{P}[\mathcal{D}]$ defined by a family $\mathcal{D}$ with $\mathrm{xor}_2 \in \mathcal{D}$ to a class defined by $\mathcal{D} \setminus \{\mathrm{xor}_2\}$.

Theorem 1 (Removal of the xor_2-Pool). *Let $\mathcal{D}$ be a 2-family in normal form containing xor_2 as a maximal pool. Every $A \in \mathrm{P}[\mathcal{D}]$ is* 1-tt *Δ_2^p-reducible to some language $B \in \Delta_2^p[\mathcal{D} \setminus \{\mathrm{xor}_2\}]$, where $B \in \mathrm{P}^{\mathrm{NP},A}$.*

Proof. Suppose $A \in \mathrm{P}[\mathcal{D}]$ via a function $f \in \mathrm{FP}$. For every $x \in \Sigma^*$ define a set $R(x)$ of words below x to which, informally, x is connected by a chain of xor's of length at most two; define r_x as the minimum of $R(x)$:

- $R_1(x) := \{y \mid y \leqslant x \wedge f(x,y) = \mathrm{xor}_2\}$.
- $R_2(x) := \{y \mid y \leqslant x \wedge \exists z, z \leqslant x.\ f(x,z) = \mathrm{xor}_2 \wedge f(z,y) = \mathrm{xor}_2\}$.
- $R(x) := \{x\} \cup R_1(x) \cup R_2(x)$ and $r_x := \min R(x)$.

The properties "$y \in R(x)$" and "$\exists r, r \leqslant z.\ r \in R(x)$" are NP-properties. Hence, on input x, r_x can be computed via binary search in polynomial time with access to an NP oracle. Define a language B by $\chi_B(x) := \chi_A(r_x))$. Because r_x can be computed with the help of an NP oracle, we have $B \in \mathrm{P}^{\mathrm{NP},A}$.

Why is A 1-tt reducible to B? We show how to compute $\chi_A(x)$: First compute r_x using the NP oracle. Check, which of the cases $x = r_x$, $x \in R_1(x)$, or $x \in R_2(x)$ holds. Compute $\chi_B(x)$ by querying B. If $x = r_x$ or $x \in R_2(x)$, then $\chi_A(x) = \chi_A(r_x) = \chi_B(x)$. If $x \in R_1(x)$, then $\chi_A(x) = 1 - \chi_A(r_x) = 1 - \chi_B(x)$.

Why is $B \in \Delta_2^p[\mathcal{D} \setminus \{\mathrm{xor}_2\}]$? We exhibit an algorithm that computes a function f_B for B. On input (x,y) a pool $D = f_B(x,y)$ has to be produced such that $\chi_B(x,y) \in D$ and $D \in \mathcal{D}$ but $D \neq \mathrm{xor}_2$. By definition of B, $\chi_B(x,y) = \chi_A(r_x, r_y)$. Hence a pool for (r_x, r_y) with respect to A has to be found. First, compute r_x and r_y with the help of an NP oracle. If $r_x = r_y$, output the equivalence pool equ_2. This pool is in $\mathcal{D}$ since $\mathcal{D}$ contains xor_2, and equ_2 is in every 2-family in normal form that contains some pool of size 2. If $r_x \neq r_y$, w.l.o.g. say $r_x < r_y$, compute $f(r_x, r_y)$. If $f(r_x, r_y) \neq \mathrm{xor}_2$, just take this pool as output. If $f(r_x, r_y) = \mathrm{xor}_2$, then it must be the case that $r_y \in R_2(y)$, because otherwise r_x would be in $R(y)$ and r_y would not be minimal in $R(y)$. Hence $\chi_A(y) = \chi_A(r_y)$. Now compute $f(r_x, y)$ and output this pool. $f(r_x, y)$ is a pool for (r_x, r_y) and $f(r_x, y) \neq \mathrm{xor}_2$, because otherwise r_x would be in $R(y)$, again a contradiction.

Because $f \in \mathrm{FP}$, the algorithm runs in polynomial time except for the use of the NP oracle to compute r_x and r_y. □

Corollary 1 (Removal of the xor_2-Pool).

1. *Every $A \in \mathrm{P}[\mathrm{CHEAT}_2]$ is* 1-tt *Δ_2^p-reducible to some $B \in \Delta_2^p[\mathrm{MIN}_2]$, where $B \in \mathrm{P}^{\mathrm{NP},A}$.*
2. *Every $A \in \mathrm{P}[\mathrm{SEL}_2 \cup \{\mathrm{xor}_2\}]$ is* 1-tt *Δ_2^p-reducible to some $B \in \Delta_2^p[\mathrm{SEL}_2]$, where $B \in \mathrm{P}^{\mathrm{NP},A}$.*

Theorem 2 (Reduction from 2-CARD$_2$ to SEL$_2$ $\cup$ BOTTOM$_2$). *Every $A \in \mathrm{P}[2\text{-CARD}_2]$ is 1-tt Δ_3^p-reducible to some $B \in \Delta_3^p[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$, where $B \in \mathrm{P}^{\Sigma_2^p, A}$.*

Proof. Let $A \in \mathrm{P}[2\text{-CARD}_2]$ via $f \in \mathrm{FP}$, such that f only outputs maximal pools from 2-CARD$_2$, i.e. equ$_2$, bottom$_2$ or top$_2$. A word $x \in \Sigma^*$ is called *easy* for a length n if there are $y_1, y_2 \in \Sigma^{\leqslant n}$ such that $|f(x, y_1, y_2)[1]| = 1$. Otherwise x is called *hard* for n. For a length n and $i \in \{0, 1\}$ we define the following sets:

- $\mathrm{EASY}_i^n := \{x \mid \exists y_1, y_2 \in \Sigma^{\leqslant n}. f(x, y_1, y_2)[1] = \{i\}\}$.
- $\mathrm{EASY}^n := \mathrm{EASY}_0^n \cup \mathrm{EASY}_1^n$ and $\mathrm{HARD}^n := \Sigma^{\leqslant n} \setminus \mathrm{EASY}^n$.

The properties "$x \in \mathrm{EASY}_i^n$" and "$x \in \mathrm{EASY}^n$" are NP-properties. For any given n we can determine the membership of x to HARD^n with one query to a co-NP oracle in time polynomial in n. We distinguish two types of word lengths: A length n is called a *top-length* if there are $x_1, x_2 \in \mathrm{HARD}^n$ with $f(x_1, x_2) = \mathrm{top}_2$, and is called a *bottom-length* otherwise. Using a Σ_2^p oracle we can determine the type of any given length n. Define the language B as follows:

$$\chi_B(x) := \begin{cases} 1 - \chi_A(x), & \text{if } x \in \mathrm{HARD}^{|x|} \text{ and } |x| \text{ is a top-length} \\ \chi_A(x), & \text{otherwise.} \end{cases}$$

We have $B \in \mathrm{P}^{\Sigma_2^p, A}$. We show how to compute $\chi_A(x)$ with one query to B: If x is easy we can compute $\chi_A(x)$ using an NP oracle. If x is hard, determine whether $|x|$ is a top-length or a bottom-length using a Σ_2^p oracle. Query x to B. If $|x|$ is a bottom-length we get $\chi_A(x) = \chi_B(x)$, otherwise $\chi_A(x) = 1 - \chi_B(x)$.

Next we show $B \in \Delta_3^p[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$. We exhibit an algorithm that computes a function f_B for B. On input (x, y) with $|x| = m$ and $|y| = n$ (w.l.o.g. $m \leqslant n$) first determine the type of the word lengths m and n by querying a Σ_2^p oracle. If x or y is easy for length n, $\chi_A(x)$ or $\chi_A(y)$ can easily be computed. If $x \in \mathrm{EASY}^n$, we get $\chi_B(x) = 1 - \chi_A(x)$ or $\chi_B(x) = \chi_A(x)$ depending on the type of length m. If $y \in \mathrm{EASY}^n$, obviously $\chi_B(y) = \chi_A(y)$. One can now output either $\{\chi_B(x)0, \chi_B(x)1\} \in \mathrm{SEL}_2$ or $\{0\chi_B(y), 1\chi_B(y)\} \in \mathrm{SEL}_2$ as a pool for x and y. Now suppose $x, y \in \mathrm{HARD}^n$. Compute $f(x, y)$. There are three different cases:

1. If both m and n are bottom-lengths, we have $f(x, y) \neq \mathrm{top}_2$ and $\chi_A(x, y) = \chi_B(x, y)$. Hence we may output $f(x, y)$ as a pool for $\chi_B(x, y)$.
2. If m is bottom- and n is top-length, we output $\mathrm{flip}_2(f(x, y))$ which is a pool for $\chi_B(x, y)$ by the definition of B. If m is top- and n is bottom-length we output $\mathrm{flip}_1(f(x, y))$ as a pool for $\chi_B(x, y)$ which is allowed, because $\mathrm{flip}_i(\mathrm{top}_2) \in \mathrm{SEL}_2$, $\mathrm{flip}_i(\mathrm{bottom}_2) \in \mathrm{SEL}_2$ and $\mathrm{flip}_i(\mathrm{equ}_2) = \mathrm{xor}_2 \in \mathrm{BOTTOM}_2$ for $i \in \{1, 2\}$.
3. Suppose both m and n are top-lengths. In case $f(x, y) \neq \mathrm{bottom}_2$ we output $\mathrm{flip}_{1,2}(f(x, y)) \in \mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2$. In case $f(x, y) = \mathrm{bottom}_2$ we output bottom_2. We have to argue why this is correct. This means we have to show that $\chi_A(x, y) \in \mathrm{top}_2$. We consider the output of f on words from HARD^n. We have sort of a transitivity property for equ$_2$: If $z_1, z_2, z_3 \in \mathrm{HARD}^n$ and

$f(z_1, z_2) = f(z_2, z_3) = \mathrm{equ}_2$, then also $f(z_1, z_3) = \mathrm{equ}_2$. Otherwise at least one of the z_i would be easy. We also have two non-mixing properties for bottom_2 and top_2: First, if $f(z_1, z_2) \in \{\mathrm{bottom}_2, \mathrm{top}_2\}$ and $f(z_2, z_3) = \mathrm{equ}_2$, then $f(z_1, z_2) = f(z_1, z_3)$. Secondly, for every three $z_1, z_2, z_3 \in \mathrm{HARD}^n$ with $f(z_i, z_j) \neq \mathrm{equ}_2$, i.e. $f(z_i, z_j) \in \{\mathrm{bottom}_2, \mathrm{top}_2\}$, for each pair it must hold $f(z_1, z_2) = f(z_2, z_3) = f(z_1, z_3)$. For, for any mix of bottom_2 and top_2, pools at least one z_i would turn out to be easy. Remember that because n is a top-length, there are $x_1, x_2 \in \mathrm{HARD}^n$ with $f(x_1, x_2) = \mathrm{top}_2$.

a) Suppose there are hard words z_1, z_2, z_3 with $f(z_i, z_j) \in \{\mathrm{bottom}_2, \mathrm{top}_2\}$. Because of $f(x_1, x_2) = \mathrm{top}_2$ and the non-mixing properties we must get $f(x, y) = \mathrm{top}_2$, a contradiction.

b) Suppose there are *no* z_1, z_2, z_3 with $f(z_i, z_j) \in \{\mathrm{bottom}_2, \mathrm{top}_2\}$. We have $f(x_1, x_2) = \mathrm{top}_2$ and $f(x, y) = \mathrm{bottom}_2$. Then either $f(x, x_1) = f(y, x_2) = \mathrm{equ}_2$ or $f(x, x_2) = f(y, x_1) = \mathrm{equ}_2$, and hence $\chi_A(x, y) \in \mathrm{xor}_2 \subseteq \mathrm{top}_2$. □

The proof of the following result is also based on the idea of avoiding the top_2-pool by flipping membership for a word x, if necessary, when changing from A to B. But to decide wether one has to flip or not, an exponential time oracle is needed. [The proof is omitted due to lack of space.]

Theorem 3 (Reduction from 3-SIZE$_2$ to SEL$_2$∪BOTTOM$_2$). *Every* $A \in \mathrm{P}[3\text{-}\mathrm{SIZE}_2]$ *is* 1-*tt* $\mathrm{P}^{\mathrm{EXP}}$-*reducible to some* $B \in \mathrm{EXP}[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$. *Moreover* B *is* 1-*tt* EXP-*reducible to* A.

4 Negative Results

We now present negative results. This means that we prove the non-existence of reducibilities between pairs of partial information classes. We first list the negative results that are already known:

Fact 4 (Known Negative Results).

1. *There is an* $A \in \mathrm{P}[\mathrm{MIN}_2]$ *which is not Turing* $\mathrm{P}^{\mathrm{EXP}}$-*reducible to any* $B \in \mathrm{EXP}$. *[20]*
2. *There is an* $A \in \mathrm{P}[\mathrm{BOTTOM}_2]$ *which is not Turing* $\mathrm{P}^{\mathrm{EXP}}$-*reducible to any* $B \in \mathrm{EXP}[\mathrm{CHEAT}_2]$. *[5]*
3. *There is an* $A \in \mathrm{P}[\mathrm{SEL}_2]$ *which is not Turing* $\mathrm{P}^{\mathrm{EXP}}$-*reducible to any* $B \in \mathrm{EXP}[2\text{-}\mathrm{CARD}_2]$. *[25,18]*
4. *There is an* $A \in \mathrm{P}[4\text{-}\mathrm{SIZE}_2]$ *which is not Turing* $\mathrm{P}^{\mathrm{EXP}}$-*reducible to any* $B \in \mathrm{EXP}[3\text{-}\mathrm{SIZE}_m]$. *[2]*
5. *There is an* $A \in \mathrm{P}[\mathrm{SEL}_2 \cup \mathrm{BOTTOM}_2]$ *which is not bounded truth table* $\mathrm{P}^{\mathrm{EXP}}$-*reducible to any* $B \in \mathrm{EXP}[\mathrm{SEL}_2]$. *[6]*

Next we will show that there is no reducibility from 2-CARD_2 to BOTTOM_2. For this we use the structural property of bi-immunity. For a complexity class $\mathcal{C}$ a language A is $\mathcal{C}$-bi-immune if neither A nor $\overline{A}$ contain an infinite subset

$B \in \mathcal{C}$. In [20] it is shown that $\mathrm{P}[2\text{-}\mathrm{CARD}_2]$ contains P-bi-immune languages. The diagonalisation technique used to prove this also works to construct a language that is EXP-bi-immune. Hence we get:

Fact 5 (Bi-Immunity). $\mathrm{P}[2\text{-}\mathrm{CARD}_2]$ *contains EXP-bi-immune languages.*

In [20] it is also shown that $\mathrm{P}[\mathrm{BOTTOM}_2]$ does not contain P-bi-immune languages. We extend this by proving that the reduction closure of $\mathrm{EXP}[\mathrm{BOTTOM}_2]$ under Turing $\mathrm{P}^{\mathrm{EXP}}$-reductions does not contain EXP-bi-immune languages. The proof technique for Theorem 4 is an extension of a technique applied in [20]. [The proof is omitted due to lack of space.] Fact 6 is shown in [8]. They even show this collapse for all easily countable languages.

Theorem 4 (Truth table reductions, BOTTOM_2, and bi-immunity). *Let A and B be languages, X some oracle language. If A is truth table P^X-reducible to B and $B \in \mathrm{P}^X[\mathrm{BOTTOM}_2]$, then A is not P^X-bi-immune.*

Fact 6 (Turing implies truth table reducibility for BOTTOM_2). *Let A, B, X be languages. If A is Turing P^X-reducible to B and B is in $\mathrm{P}^X[\mathrm{BOTTOM}_2]$, then A ist truth table P^X-reducible to B.*

Theorem 5 (No Reduction from 2-CARD_2 to BOTTOM_2). *There is an $A \in \mathrm{P}[2\text{-}\mathrm{CARD}_2]$ that is not Turing $\mathrm{P}^{\mathrm{EXP}}$-reducible to any $B \in \mathrm{EXP}[\mathrm{BOTTOM}_2]$.*

Proof. By Fact 5, $\mathrm{P}[2\text{-}\mathrm{CARD}_2]$ contains EXP-bi-immune languages. But by combining Theorem 4 and Fact 6 the closure of $\mathrm{EXP}[\mathrm{BOTTOM}_2]$ under Turing $\mathrm{P}^{\mathrm{EXP}}$-reductions does not contain EXP-bi-immune languages. □

Now we turn to negative results for the families BOTTOM_2 and SEL_2. Lemma 1 provides a general tool to show one query non-reducibility results. Proposition 1 states that this tool is applicable to BOTTOM_2 and SEL_2. This implies Theorem 6. [The proof of Proposition 1 is omitted due to lack of space.]

Lemma 1 (Upward translation and closure under bit-flip). *Let $\mathcal{D}_1$, $\mathcal{D}_2$ be k-families in normal form, $m \geqslant k$, and $\lceil \mathcal{D}_1 \rceil_m \not\subseteq \langle \lceil \mathcal{D}_2 \rceil_m \rangle_{\mathrm{flip}}$. Then there is an $A \in \mathrm{P}[\mathcal{D}_1]$ which is not 1-tt $\mathrm{P}^{\mathrm{EXP}}$-reducible to any $B \in \mathrm{EXP}[\mathcal{D}_2]$.*

Proof. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be given as above. Due to Fact 2.2 we have $\mathrm{P}[\lceil \mathcal{D}_1 \rceil_m] = \mathrm{P}[\mathcal{D}_1]$ and $\mathrm{EXP}[\lceil \mathcal{D}_2 \rceil_m] = \mathrm{EXP}[\mathcal{D}_2]$. We know that $\mathrm{EXP}[\langle \lceil \mathcal{D}_2 \rceil_m \rangle_{\mathrm{flip}}]$ is closed under 1-tt EXP-reduction (Fact 2.3).
Because $\lceil \mathcal{D}_1 \rceil_m \not\subseteq \langle \lceil \mathcal{D}_2 \rceil_m \rangle_{\mathrm{flip}}$, by Fact 1.3 there exists an $A \in \mathrm{P}[\lceil \mathcal{D}_1 \rceil_m]$ which is not a member of $\mathrm{EXP}[\langle \lceil \mathcal{D}_2 \rceil_m \rangle_{\mathrm{flip}}]$. It follows that A is not 1-tt $\mathrm{P}^{\mathrm{EXP}}$-reducible to any member of $\mathrm{EXP}[\langle \lceil \mathcal{D}_2 \rceil_m \rangle_{\mathrm{flip}}]$. □

Proposition 1. *Let $\mathrm{BOTTOM}_3 := \lceil \mathrm{BOTTOM}_2 \rceil_3$ and $\mathrm{SEL}_3 := \lceil \mathrm{SEL}_2 \rceil_2$. Then BOTTOM_3 is not contained in $\langle \mathrm{SEL}_3 \rangle_{\mathrm{flip}}$.*

Theorem 6 (No reduction from BOTTOM_2 to $\mathrm{SEL}_2 \cup \{\mathrm{xor}_2\}$). *There are languages in $\mathrm{P}[\mathrm{BOTTOM}_2]$ which are not 1-tt $\mathrm{P}^{\mathrm{EXP}}$-reducible to any set in $\mathrm{EXP}[\mathrm{SEL}_2 \cup \{\mathrm{xor}_2\}]$.*

5 Conclusion and Open Problems

We have achieved results for all pairs of 2-families, for which there was no reducibility or non-reducibility result known up to now. Our positive results all are in terms of 1-tt reductions, but of different quality due to the resource bounds used for the partial information algorithm and the reduction. Figure 2 gives an overview on the known and new results. It would be nice, if for the reduction from 3-SIZE$_2$ to SEL$_2$ ∪ BOTTOM$_2$ one could do with an PH oracle. In light of the result of [6] it may be possible for some positive results to get rid of the PH oracles, at least for the partial information algorithm for B. In case of the negative results we sometimes can only state the non-existence of one query truth-table reductions. It would be nice to extend such results to more general reductions.

Unfortunately, the sufficient condition given in Lemma 1 is not necessary. The families 2-CARD$_2$ and BOTTOM$_2$ provide a counterexample. Therefore, the search for (hopefully easy) combinatorial criteria to decide the existence or non-existence of reducibilities between partial information classes has to go on.

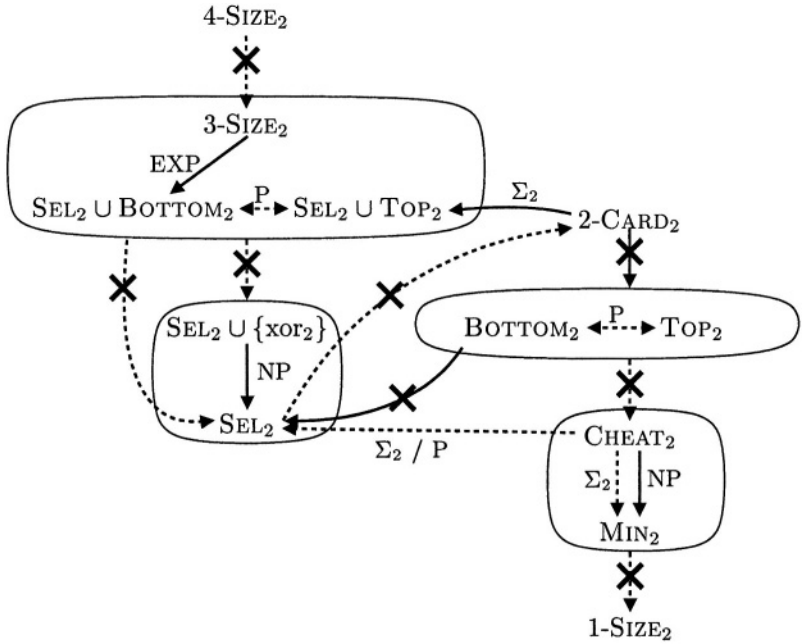


Fig. 2. If two families $\mathcal{D}_1$, $\mathcal{D}_2$ are grouped together by an encircling line, the languages in EXP$[\mathcal{D}_1]$ are reducible to languages in EXP$[\mathcal{D}_2]$, and vice versa. An arrow marked with $\mathcal{C}$ from $\mathcal{D}_1$ to $\mathcal{D}_2$ means that languages in P$[\mathcal{D}_1]$ are 1-tt P$^{\mathcal{C}}$-reducible to languages in P$^{\mathcal{C}}[\mathcal{D}_2]$. A crossed out arrow from $\mathcal{D}_1$ to $\mathcal{D}_2$ means that there are languages in P$[\mathcal{D}_1]$ which are not 1-tt P$^{\mathrm{EXP}}$-reducible to languages in EXP$[\mathcal{D}_2]$. See Section 3 to find out in which cases 1-tt can be improved to btt or Turing reductions. Continuous arrows stand for new, dotted arrows for previously known results.

References

1. A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query classes and non-uniform complexity. In *Proc. 5th Struct. in Complexity Th.,* 1990.
2. A. Amir and W. Gasarch. Polynomial terse sets. *Inf. and Computation,* 77, 1988.
3. S. Bab. Reduktionsbeziehungen zwischen Teilinformationsklassen bei beschränkter Fragenzahl. Master's thesis, Technical University Berlin, Germany, 2003.
4. S. Bab and A. Nickelsen. Reducibilities between polynomial time partial information classes over 2-families. Technical Report 2004/07, TU Berlin, 2004.
5. R. Beigel. *Query-Limited Reducibilities.* PhD thesis, Stanford University, 1987.
6. R. Beigel, L. Fortnow, and A. Pavan. Membership comparable and p-selective sets. Technical Report 2002-006N, NEC Research Institute, 2002.
7. R. Beigel, W. Gasarch, and E. Kinber. Frequency computation and bounded queries. In *Proc. 10th Structure in Complexity Theory,* 1995.
8. R. Beigel, M. Kummer, and F. Stephan. Quantifying the amount of verboseness. In *Proc. Logical Found. of Comput. Sci.,* volume 620 of *LNCS.* Springer, 1992.
9. R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Inf. and Computation,* 120(2), 1995.
10. H. Buhrman, L. Torenvliet, and P. van Emde Boas. Twenty questions to a P-selector. *Information Processing Letters,* 4(4):201–204, 1993.
11. W. Gasarch. Bounded queries in recursion theory: A survey. In *Proc. 6th Structure in Complexity Theory,* 1991.
12. L. Hemaspaandra, A. Hoene, and M. Ogihara. Reducibility classes of p-selective sets. *Theoretical Computer Science,* 155:447–457, 1996.
13. L. Hemaspaandra, Z. Jiang, J. Rothe, and O. Watanabe. Polynomial-time multi-selectivity. *J. of Universal Comput. Sci.,* 3(3), 1997.
14. L. Hemaspaandra and T. Torenvliet. *Theory of semi-feasible Algorithms.* Springer, 2002.
15. M. Hinrichs and G. Wechsung. Time bounded frequency computations. In *Proc. 12th Conf. on Computational Complexity,* 1997.
16. A. Hoene and A. Nickelsen. Counting, selecting, and sorting by query-bounded machines. In *Proc. STACS 93,* volume 665 of *LNCS.* Springer, 1993.
17. K. Ko. On self-reducibility and weak p-selectivity. *Journal of Computer and System Sciences,* 26:209–221, 1983.
18. M. Kummer. A proof of Beigel's cardinality conject. *J. of Symb. Log.,* 57(2), 1992.
19. A. Nickelsen. On polynomially $\mathcal{D}$-verbose sets. In *Proc. STACS 97,* volume 1200 of *LNCS,* pages 307–318. Springer, 1997.
20. A. Nickelsen. Partial information and special case algorithms. In *Proc. MFCS 01,* pages 573–584. Springer LNCS 2136, 2001.
21. A. Nickelsen. *Polynomial Time Partial Information Classes.* W&T Verlag, 2001. Dissertation, TU Berlin, 1999.
22. A. Nickelsen and T. Tantau. Closure of polynomial time partial information classes under polynomial time reductions. In *Proc. FCT 01,* volume 2138 of *LNCS,* pages 299–310. Springer, 2001.
23. A. Nickelsen and T. Tantau. Partial information classes. Complexity Theory Column,*SIGACT News,* 34, 2003.
24. M. Ogihara. Polynomial-time membership comparable sets. *SIAM Journal on Computing,* 24(5): 1068–1081, 1995.
25. A. Selman. P-selective sets, tally languages and the behaviour of polynomial time reducibilities on NP. *Math. Systems Theory,* 13:55–65, 1979.
26. T. Tantau. Combinatorial representations of partial information classes and their truth-table closures. Master's thesis, TU Berlin, Germany, 1999.

A New Dimension Sensitive Property for Cellular Automata

Vincent Bernardi[1], Bruno Durand[1], Enrico Formenti[2], and Jarkko Kari[3]

[1] LIF, CNRS - Universit de Provence, 39 rue F. Joliot Curie 13453 Marseille Cedex 13, France
{Vincent.Bernardi,Bruno.Durand}@lif.univ-mrs.fr
[2] I3S, UNSA - CNRS, 2000, route des lucioles, Les Algorithmes - bat Euclide B, BP 121, 06903 Sophia Antipolis Cedex, France
enrico.formenti@unice.fr
[3] Mathematics Department FIN-20014 University of Turku, Finland
jjkari@cs.uiowa.edu

Abstract. In this paper we study number-decreasing cellular automata. They form a super-class of standard number-conserving cellular automata. It is well-known that the property of being number-conserving is decidable in quasi-linear time. In this paper we prove that being number-decreasing is dimension sensitive *i.e.* it is decidable for one-dimensional cellular automata and undecidable for dimension 2 or greater. There are only few known examples of dimension sensitive properties for cellular automata and this denotes some rich panel of phenomena in this class.

Introduction

Number-conserving cellular automata (NCA) were introduced in [13] as an alternative to differential equations for modeling highway car traffic.

The huge literature published since then witnesses the great interest they received from the scientific community gathering together physicians, computer scientists and mathematicians. In fact, the NCA model applies to all phenomena governed by conservation laws of mass or energy.

In [2], the authors proved that the property of being number-conserving is decidable in all dimensions. This is quite surprising since almost all non-trivial properties on cellular automata are known to be undecidable [12,15,16,8,14].

In this paper we introduce *number-decreasing* cellular automata, a variant of NCA that can be used for modeling systems in which the total mass or energy do not increase during the evolution.

Our main results (Theorems 1 and 4) prove that the property of being number-decreasing is dimension sensitive: it is decidable in dimension 1 and undecidable in greater dimensions. This is the only known property on cellular automata with such a peculiarity if we except pure set properties like surjectivity and injectivity (see for example [14]).

The proof technique of Theorem 5 is of some interest in its own since we use a subtle construction for reducing a problem on cellular automata to a problem of finite tiling of the plane that might be of help in other situations.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 416–426, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Finally, several combinatorial results relate surjectivity and balance (see page 5) to the property of being number-decreasing.

1 Basic Definitions

In this paper, $\mathbb{N}^*$ is the set of positive integers; a *neighborhood* is any finite ordered subset of $\mathbb{Z}^d$ ordered by the lexicographic order; a *principal neighborhood* is a neighborhood whose smallest element is $(0,\ldots,0)$.

Formally, a *cellular automaton* $\mathcal{A}$ is a tuple $(d, S, \mathcal{V}, f)$ where d is a positive integer, S a finite set, $\mathcal{V}$ a principal neighborhood and f a map from $S^{\mathcal{V}}$ to S. S is called the *set of states* of the automaton $\mathcal{A}$, d its *dimension* and f its *local evolution rule*. If $\mathcal{V} = \{\mathbf{v_0},\ldots,\mathbf{v_{n-1}}\}$ with $\forall i, \mathbf{v_i} < \mathbf{v_{i+1}}$, the conjugate function of f is the map $\hat{f}$ from $S^{\mathrm{card}(\mathcal{V})}$ to S defined by $\hat{f}(u_0,\ldots,u_{n-1}) = f(u)$ where u is the map from $\mathcal{V}$ to S defined by $u(\mathbf{v_0}) = u_0,\ldots,u(\mathbf{v_{n-1}}) = u_{n-1}$. To simplify notations, we omit the distinction between f and $\hat{f}$ and use the symbol f for both. In the rest of this article, unless otherwise mentioned, we always assume that the cellular automaton has the same set of states which is always noted S.

A *configuration* is a map from $\mathbb{Z}^d$ to S, where S is the set of states. The set of all configurations is noted C. A configuration c in C is called *spatially periodic* iff there exists a vector $\mathbf{\Pi}$ in $\mathbb{N}^{*d}$ such that $\forall i \in \{1,\ldots,d\}, \forall \mathbf{x} \in \mathbb{Z}^d, c(\mathbf{x}+\mathbf{\Pi_i}) = c(\mathbf{x})$, where $\mathbf{\Pi_i}$ is the orthogonal projection of $\mathbf{\Pi}$ on the i-th axis. The vector $\mathbf{\Pi}$ is called a *period* of c. The set of configurations which are spatially periodic for a given vector $\mathbf{\Pi}$ is noted $C_{\mathbf{\Pi}}$. The set of all periodic configurations is noted C_p. A configuration c is called *s-finite*, for $s \in S$ iff c is almost everywhere equal to s (*i.e.* except on a finite subset of $\mathbb{Z}^d$).

The *global rule* of a CA $\mathcal{A} = (S, d, \mathcal{V}, f)$ where $\mathcal{V} = (\mathbf{v_0}, \mathbf{v_1},\ldots,\mathbf{v_{n-1}})$ is the map A from C onto itself defined as follows

$$\forall c \in C, \forall \mathbf{x} \in \mathbb{Z}^d, A(c)(\mathbf{x}) = f(c(\mathbf{x}+\mathbf{v_0}), c(\mathbf{x}+\mathbf{v_1}),\ldots,c(\mathbf{x}+\mathbf{v_{n-1}}))$$

In the sequel, a CA is said *surjective* if its global rule is surjective.

In literature, there are several equivalent definitions for the concept of number-conserving CA (see [2]). One of these is concerned with the action of the automaton over periodic configurations, another with its action over finite configurations and another with the asymptotic properties of every configuration. In the present paper, we will only concern ourselves with the first two definitions.

Definition 1 (Number-Conserving Cellular Automata). *Let $\mathcal{A}$ be a cellular automaton over the set of states $\{0,\ldots,q-1\}$. Then $\mathcal{A}$ is* number-conserving *iff either of these two equivalent definitions holds:*

- *For any 0-finite configuration c, $\sum_{\mathbf{x}\in\mathbb{Z}^d} A(c)(\mathbf{x})$ exists and equals $\sum_{\mathbf{x}\in\mathbb{Z}^d} c(\mathbf{x})$.*
- *For any periodic configuration $\mathbf{c}$ of period $\mathbf{\Pi} = (\pi_1,\ldots,\pi_d)$, we have:*

$$\sum_{\substack{0\le x_1<\pi_1 \\ \cdots \\ 0\le x_d<\pi_d}} A(c)(\boldsymbol{x}) = \sum_{\substack{0\le x_1<\pi_1 \\ \cdots \\ 0\le x_d<\pi_d}} c(\boldsymbol{x})$$

Let us denote NCA the class of number-conserving CA.

Definition 2 (Finite Number-Decreasing Cellular Automata). *Consider a CA* $\mathcal{A}$ *with* $S = \{0, \ldots, n-1\}$. *Then* $\mathcal{A}$ *is* finite number-decreasing *if and only if 0 is quiescent for* $\mathcal{A}$ *and for any 0-finite configuration* $\mathbf{c}$, *it holds*

$$\sum_{\mathbf{x} \in \mathbb{Z}^d} A(c)(\mathbf{x}) \leq \sum_{\mathbf{x} \in \mathbb{Z}^d} c(\mathbf{x}).$$

Definition 3 (Periodic Number-Decreasing Cellular Automata). *Consider a CA* $\mathcal{A}$ *with* $S = \{0, \ldots, n-1\}$. *Then* $\mathcal{A}$ *is* finite number-decreasing *if and only if for any periodic configuration* $\mathbf{c}$ *of period* $\mathbf{\Pi} = (\pi_1, \ldots, \pi_d)$, *we have:*

$$\sum_{\substack{0 \leq x_1 < \pi_1 \\ \cdots \\ 0 \leq x_d < \pi_d}} A(c)(\mathbf{x}) \leq \sum_{\substack{0 \leq x_1 < \pi_1 \\ \cdots \\ 0 \leq x_d < \pi_d}} c(\mathbf{x})$$

The proof of equivalence for NCA in [2] can be easily adapted to prove the equivalence of Definitions 2 and 3. For this reason in the sequel we will contextually use the most convenient one. Let us denote NDCA the class of number-decreasing CA.

2 Decidability

In [1], the decidability of the number conserving property is proved. Later in [2], it has been proved that the number-conserving property is decidable in all dimensions in quasi-linear time with respect to the size of the transition table.

Definition 4 (De Bruijn Graph). *The* De Bruijn graph *(V,E) of a one-dimensional CA* $(1, S, \mathcal{V}, f)$ *is the labeled directed graph in which* $V = S^{\mathrm{card}(\mathcal{V})-1}$ *and whose set of edges is defined by:*

$$((a_0, \ldots, a_{n-2}), (b_0, \ldots, b_{n-2})) \in E \Leftrightarrow \begin{cases} a_1 = b_0 \\ a_2 = b_1 \\ \cdots \\ a_{n-2} = b_{n-3} \end{cases}$$

An edge $((a, \alpha_1, \ldots, \alpha_{n-2}), (\alpha_1, \ldots, \alpha_{n-2}, b))$ *is labeled* $f(a, \alpha_1, \ldots, \alpha_{n-2}, b)$. *The* value *of any vertex* $(a_0, \ldots, a_{n-2})$ *is by definition* a_0 *and the* value *of any edge* $((a, \alpha_1, \ldots, \alpha_{n-2}), (\alpha_1, \ldots, \alpha_{n-2}, b))$ *is by definition its label* $f(a, \alpha_1, \ldots, \alpha_{n-2}, b)$.

Examples of De Bruijn graphs are given in Figures 1 and 2.

Theorem 1 (Decidability). *The number-decreasing property is decidable in dimension* 1.

Proof. Let $\mathcal{A}$ be a one-dimensional number-conserving (respectively number-decreasing) CA. Let $\tilde{c}$ be a circuit in the De Bruijn graph of $\mathcal{A}$. By taking the successive values of the vertices of $\tilde{c}$, we create a periodic 1-dimensional configuration c. $\mathcal{A}(c)$ is then by design the periodic configuration obtained by taking the successive values of the edges of $\tilde{c}$. Since $\mathcal{A}$ is number-conserving (resp. number-decreasing), the sum of the values of the vertices is equal to (resp. greater than) the sum of the values of the edges. Conversely, using again the bijection between periodic configurations and cycles of the De Bruijn graph, it is clear that if on each circuit of the graph the sum of the values of the vertices is equal to (resp. greater than) the sum of the values of the edges then the CA is number-conserving (resp. number-decreasing).

Consider now the De Bruijn graph G of a one-dimensional CA. It is clear that the number-decreasing property is verified on all hamiltonian circuits of G then it is verified on all circuits of G, since the property is additive. Since there are only a finite number of hamiltonian circuits in a given graph, the number-conserving (resp. number-decreasing) property is decidable in dimension 1. □

3 Undecidability

In this section we prove our main undecidability result and several combinatorial properties that give semi-decidable procedures for the property of being number-decreasing in dimensions greater than one.

A CA $\mathcal{A} = (d, S, \mathcal{V}, f)$ is *k-balanced* if all neighborhoods of size k have the same number of preimages by $\mathcal{A}$. In [11], surjectivity has been reformulated in term of k-balance: a CA is surjective if and only if it is k-balanced for all $k \in \mathbb{N}^*$. Theorem 3 and the following corollaries correlate balance and surjectivity with the property of being number-decreasing.

Theorem 2. *A two-state number-conserving CA* $\mathcal{A} = (d, \{0,1\}, \mathcal{V}, f)$ *is* 1*-balanced,* i.e. $\mathrm{card}(f^{-1}(\{0\})) = \mathrm{card}(f^{-1}(\{1\}))$. *A two-state number-decreasing CA* $\mathcal{A} = (d, \{0,1\}, \mathcal{V}, f)$ *verifies* $\mathrm{card}(f^{-1}(\{0\})) \geq \mathrm{card}(f^{-1}(\{1\}))$.

Proof. In this proof, to simplify notations, let $\mathcal{P}_d$ be the set $\{0, \ldots, n-1\}^d$. Let us call $|0|$ the cardinality of $f^{-1}(\{0\})$ and $|1|$ the cardinality of $f^{-1}(\{1\})$. We assume without loss of generality that the principal neighborhood of $\mathcal{A}$ is an hypercube of side n. f is then a function from $S^{\mathcal{P}_d}$ to S. For a given μ in $S^{\mathcal{P}_d}$, we note c_μ the periodic configuration in $C_{\{n,\ldots,n\}}$ which matches μ (*i.e.* c_μ is the periodic configuration obtained by tiling the space with μ). Let us call τ the map from $S^{\mathcal{P}_d} \times \{0, \ldots, n-1\}^d$ to $S^{\mathcal{P}_d}$ such that for all μ in $S^{\mathcal{P}_d}$ and for all vector $\mathbf{N}$ in $\{0, \ldots, n-1\}^d$, $\tau(\mu, \mathbf{N})(\mathbf{x}) = \mu(\mathbf{x} + \mathbf{N})$ (*i.e.* τ translates the hypercube on c_μ by the vector $\mathbf{N}$). Each element μ' of $S^{\mathcal{P}_d}$ has then exactly n^d preimages by τ. By applying the number-conserving property to a configuration c_μ, we get

$$\forall \mu \in S^{\mathcal{P}_d}, \sum_{\mathbf{V} \in \mathcal{P}_d} c_\mu(\mathbf{V}) = \sum_{\mathbf{V} \in \mathcal{P}_d} \mathcal{A}(c_\mu)(\mathbf{V})$$

and therefore

$$\sum_{\mu \in S^{\mathcal{P}_d}} \sum_{\mathbf{V} \in \mathcal{P}_d} c_\mu(\mathbf{V}) = \sum_{\mu \in S^{\mathcal{P}_d}} \sum_{\mathbf{V} \in \mathcal{P}_d} \mathcal{A}(c_\mu)(\mathbf{V})$$

By the definition of the global transition rule, we have $\mathcal{A}(c_\mu)(\mathbf{V}) = f(\tau(\mu, \mathbf{V}))$, and so:

$$\sum_{\mu \in S^{\mathcal{P}_d}} \sum_{\mathbf{V} \in \mathcal{P}_d} c_\mu(\mathbf{V}) = \sum_{\mu \in S^{\mathcal{P}_d}} \sum_{\mathbf{V} \in \mathcal{P}_d} f(\tau(\mu, \mathbf{V})) = n^d * |1|$$

It is obvious that $\sum_{\mu \in S^{\mathcal{P}_d}} \sum_{\mathbf{V} \in \mathcal{P}_d} c_\mu(\mathbf{V}) = \frac{1}{2} n^d * 2^{n^d}$, hence $|1| = \frac{1}{2} 2^{n^d}$. Since $|0| + |1| = 2^{n^d}$, we must have $|0| = |1|$.

The exact same proof is valid for a number-decreasing CA by replacing the $=$ sign with a $\geq$ sign where necessary. □

Figure 1 shows that the converse of Theorem 2 is false. The automaton is 1-balanced, since there are as many arrows labeled with 1 as there are labeled with 0, but it is not number-conserving, since the image of the periodic configuration of period 011 is the periodic configuration everywhere equal to 1.

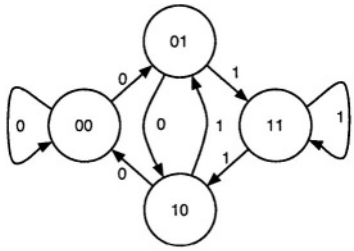


Fig. 1. De Bruijn graph of a 1-balanced but not number-conserving CA

Theorem 3. *Consider a NDCA $\mathcal{A}$. If $\mathcal{A}$ is 1-balanced then it is number-conserving.*

Proof. Assume that $\mathcal{A}$ is number-decreasing and not number-conserving. Let us modify the hypothesis of the proof of Theorem 2 as follows:

$$\begin{cases} \forall \mu \in S^{\mathcal{P}_d}, \sum_{\mathbf{V} \in \mathcal{P}_d} c_\mu(\mathbf{V}) \geq \sum_{\mathbf{V} \in \mathcal{P}_d} \mathcal{A}(c_\mu)(\mathbf{V}) \\ \exists \mu \in S^{\mathcal{P}_d}, \sum_{\mathbf{V} \in \mathcal{P}_d} c_\mu(\mathbf{V}) > \sum_{\mathbf{V} \in \mathcal{P}_d} \mathcal{A}(c_\mu)(\mathbf{V}) \end{cases}$$

Then the same proof shows that $|1| < \frac{1}{2} 2^{n^d}$ and therefore $\mathcal{A}$ is not 1-balanced. □

Corollary 1. *The global rule of a number-decreasing CA which is not number-conserving is not surjective.*

Proof. Theorem 3 implies that such an automaton is not 1-balanced. It is well-known that a CA is surjective iff it is k-balanced for all k [5]. □

Given two CA $\mathcal{A} = (d, S, \mathcal{V}, f_a)$ and $\mathcal{B} = (d, S, \mathcal{V}, f_b)$, define $\mathcal{A} \leqslant \mathcal{B}$ if and only if $\forall K \in S^{\mathcal{V}}, f_a(K) \leq f_b(K)$. If $\mathcal{A} \leqslant \mathcal{B}$, then we say that $\mathcal{B}$ is an upgraded version of $\mathcal{A}$.

Corollary 2. *If $\mathcal{A} \leqslant \mathcal{B}$ and if $\mathcal{B}$ is number-conserving, then $\mathcal{A}$ is number-decreasing.*

Proof. Going from $\mathcal{B}$ to $\mathcal{A}$ consists in decreasing the value of the local rule; therefore, if the sum of the states of a periodic configuration c is equal to the sum of the states of its image $\mathcal{B}(c)$, it is greater than the sum of the states of $\mathcal{A}(c)$. Therefore $\mathcal{A}$ is number-decreasing. □

The preceding result might lead one to wonder if NDCA $\leqslant$ NCA *i.e.* any NDCA is a kind of perturbed version of a NCA. Figure 2 invalidates this conjecture, since any increase of the value of the local rule creates an increasing cycle of length two and therefore an increasing periodic configuration of period two.

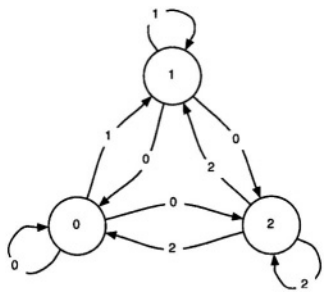


Fig. 2. NDCA that cannot be upgraded to any NCA.

Although a number-decreasing CA may not be upgradable to a NCA, it can be partially upgraded to a NCA with a larger set of states. This new automaton cannot be algorithmically extended to a complete NCA since the property of being number-conserving is decidable and, by Theorem 4, being number-decreasing is undecidable.

Theorem 4. *The number-decreasing property is not decidable in dimension 2 and greater.*

Proof. The idea of the proof is to reduce our problem to the classical halting problem for Turing machines. The proof consists in two main steps:

1. we reduce the finite tiling problem to the halting problem for Turing machines on empty input;
2. we reduce the NDCA problem to the finite tiling problem.

As a reminder, the finite tiling problem is the following: given a finite number of square tiles of size 1 with colored borders (introduced by Wang in 1961 [17]), and a *blank* tile of the same size with a given *blank* color on each border, does there exist a tiling of the plane with only a finite number of non-*blank* tiles and such that two borders facing each other always share the same color?

Step 1 is achieved by using a very interesting construction introduced in [8, 9]. We go into details of the tiling construction of [8,9], since the mechanisms of construction are essential to understand how the final part of our proof works.

Consider a Turing Machine $\mathcal{M}$. Assume that $\mathcal{M}$ is defined over the alphabet $\{0,1\}$ and has state set T.

Define a set of colors C associated with $\mathcal{M}$ as follows:

- A *blank* color B and two *inside* colors, *left* and *right*, C_l and C_r.
- Two *initialization* colors I_l and I_r.
- Two *halting* colors H_l and H_r.
- A *left border* color L and a *right border* color R.
- A *state* color t for each state $t \in T$.
- Two symbols colors 0 and 1.
- A *state-symbol* color $t.a$ for each $t \in T$ and $a \in \{0,1\}$.

Our tiling set $\mathcal{T}_\mathcal{M}$ is a subset of C^4 defined as follows:

- The *blank* tile (B, B, B, B).
- The *initialization* tiles (Fig 3)
  $(L, I_l, B, B), (0, I_l, B, I_l), (t_i.0, I_r, B, I_l), (0, I_r, B, I_r), (R, B, B, I_r)$
  where t_i is the initial state of $\mathcal{M}$.
- The *static* tiles (Fig 4)
  $(L, C_l, L, B), (0, C_l, 0, C_l), (1, C_l, 1, C_l), (0, C_r, 0, C_r), (1, C_r, 1, C_r)$,
  (R, B, R, C_r)
- *Computation* tiles (Fig 5)
  - $(a', C_r, t.a, t')$ for each transition $(t, a) \rightarrow (t', a', left)$.
  - $(a', t', t.a, C_l)$ for each transition $(t, a) \rightarrow (t', a', right)$.
- *Merging* tiles (Fig 5)
  - $(t'.a, t', a, C_l)$ for each $t' \in T$ and $a \in \{0,1\}$.
  - $(t'.a, C_r, a, t')$ for each $t' \in T$ and $a \in \{0,1\}$.
  - The *halting* tiles (Fig 6)
    $(B, H_l, L, B), (B, H_l, a, H_l), (B, H_r, a, H_r), (B, H_r, t_h.a, H_l)$,
    (B, B, R, H_r) for each $a \in \{0,1\}$.

To give a more visual intuition of the space-time evolution diagram of the Turing machine we have marked tiles in the figures with several types of arrows. Double-headed arrows represent the head of the TM and single-headed arrows show the evolutions of the state of a given cell of the tape. Tiles marked with three lines represent the external limits of the calculus of the TM. All these properties are enforced by the colors on the sides of the tiles.

This construction is known to have the following property: $\mathcal{M}$ halts when run on a blank tape if and only if $\mathcal{T}_\mathcal{M}$ admits a finite tiling of the plane (*i.e.* a

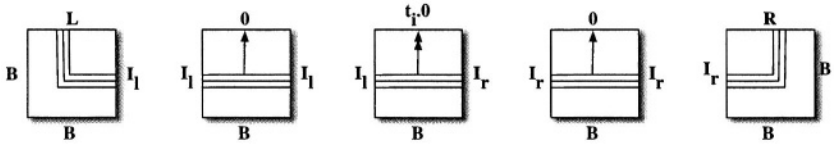


Fig. 3. Initialization tiles

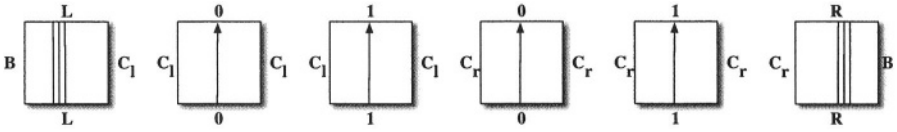


Fig. 4. Static tiles

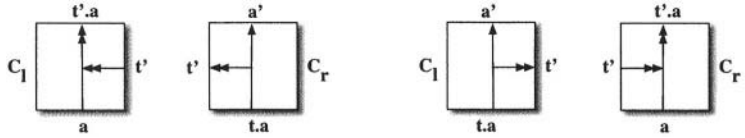


Fig. 5. Left and right computing and merging tiles

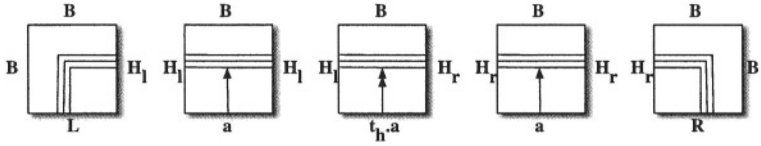


Fig. 6. Halting tiles

tiling made of a finite number of non-blank tiles). For a proof of this property, see [10].

We now proceed to construct a 2-dimensional cellular automaton $\mathcal{A}_\mathcal{M}$ in the following manner: the set of states S is $\{0, \ldots, n-1\}$ where n is the number of tiles in $\mathcal{T}_\mathcal{M}$. The neighborhood of $\mathcal{A}_\mathcal{M}$ is the standard Von Neumann neighborhood of radius 1. Let ϕ be a one-to-one mapping from S to $\mathcal{T}_\mathcal{M}$ verifying: $\phi(0) = (B, B, B, B)$, the blank tile; $\phi(1) = (t_i.0, I_r, B, I_l)$, the initialization tile with the TM head. Using ϕ, we identify S and $\mathcal{T}_\mathcal{M}$.

The local rule of the automaton is defined as follows: if a non-blank (non-zero) cell sees a tiling error on one of its borders (*i.e.* two different colors on the

same edge), it decreases its value by one; if a cell in state 1 does not see any tiling error, it increases its value by one. Otherwise, the state of the cell is left unchanged.

Assume that $\mathcal{M}$ halts when run on a blank tape. Then $\mathcal{T}_{\mathcal{M}}$ admits a valid finite tiling α of the plane with the TM-head initialization tile appearing. Using ϕ^{-1}, α maps to a finite configuration of our automaton $\mathcal{A}_{\mathcal{M}}$. Since α is valid, no cell is bordered by a tiling error, and therefore, no cell will decrease its value. Since at least one cell is in the state 1 and no tiling error occurs, this cell will increase its value. Therefore this configuration breaks the number-decreasing property and $\mathcal{A}_{\mathcal{M}}$ is not number-decreasing.

Assume that $\mathcal{M}$ does not halt when run on a blank tape. Let us prove that the number of tiles having at least one tiling error on their borders in any $\mathcal{T}_{\mathcal{M}}$ finite tiling of the plane is at least the number of head initialization tiles in the same tiling.

Let c be a finite configuration of $\mathcal{A}_{\mathcal{M}}$. In the following, we do not make the distinction between a cell and its state in c. For each cell $\mathbf{x}$ in state 1, we define $D_{\mathbf{x}}$, the domain of $\mathbf{x}$, as a subset of $\mathbb{Z}^d$ as follows:

- $\mathbf{x} \in D_{\mathbf{x}}$.
- If $\mathbf{y}$ is one of the tiles in Fig 3 which has I_l but not I_r as one of its colors, and $\mathbf{y}$'s right neighbor is in $D_{\mathbf{x}}$, then $\mathbf{y} \in D_{\mathbf{x}}$.
- Conversely, if $\mathbf{y}$ is one of the tiles in Fig 3 which has I_r but not I_l as one of its colors, and $\mathbf{y}$'s left neighbor is in $D_{\mathbf{x}}$, then $\mathbf{y} \in D_{\mathbf{x}}$.
- If $\mathbf{y}$ is one of the tiles in Fig 4, 5 or 6 and the tile below $\mathbf{y}$ is in $D_{\mathbf{x}}$, then $\mathbf{y} \in D_{\mathbf{x}}$.
- In any other case, $\mathbf{y} \notin D_{\mathbf{x}}$.

It is easy to see that no $D_{\mathbf{x}}$ intersects any other. We prove that any domain $D_{\mathbf{x}}$ contains at least a tile that sees a tiling error. Let us take $\mathbf{x}$ such that no tile in $D_{\mathbf{x}}$ sees any tiling error. Then from the tile $\mathbf{x}$ and following the valid region enclosed by the borders starting in $\mathbf{x}$, we visit a valid (since it is necessarily enclosed in $D_{\mathbf{x}}$) part of the configuration which represents a finite (since c is finite) calculus of $\mathcal{M}$. Since by hypothesis no such calculus exists, it means that $D_{\mathbf{x}}$ contains at least a tile that sees a tiling error.

Therefore, every domain $D_{\mathbf{x}}$ contains exactly one cell in state 1 ($\mathbf{x}$) and at least one cell that sees a tiling error. Furthermore, every cell $\mathbf{x}$ in state 1 is in exactly one domain ($D_{\mathbf{x}}$). It implies that the number of cell which see a tiling error is greater than the number of cell in state 1. We conclude that $\mathcal{A}_{\mathcal{M}}$ is decreasing on c. Since this property is true for any finite configuration of $\mathcal{A}_{\mathcal{M}}$, $\mathcal{A}_{\mathcal{M}}$ is number-decreasing.

We have reduced the number-decreasing problem to the halting problem; that implies that the number-decreasing problem is undecidable. □

Conclusion

In this paper we have introduced a new class of discrete dynamical systems: number-decreasing CA. It is designed to ease the simulation of physical or bio-

logical systems in which a quantity like total mass or energy does not increase with time. Our main results show that the property of being number-decreasing is dimension sensitive: it is decidable in dimension 1 and undecidable in greater dimensions.

Surprisingly the complexity character of this property differs completely from the similar property in which only strict conservation is allowed; indeed, the number-conserving property is decidable in all dimensions (see [2]).

These matters are convincing us that the study of dynamical behavior of the class of NDCA might reveal as difficult as the whole class of general cellular automata. It seems then logical to try to further restrict the class of NDCA in the hope that the newly added constraints can help to shape out the dynamical behavior. For example, one can define the class of strongly NDCA as the class of NDCA that are nilpotent (*i.e.* that for each configuration c there exists $n \in \mathbb{N}^*$ such that $f^n(c) = \underline{0}$, where $\underline{0}$ is the configuration everywhere equal to 0). Hopefully this new constraint makes the class of strongly NDCA decidable? Can we precisely describe the dynamics of these systems?

References

1. N. Boccara and H. Fukś, *Number-conserving cellular automaton rules,* Fundamenta Informaticae, IOS Press, 2002.
2. B. Durand, E. Formenti, Z. Rka, *Number conserving cellular automata I: decidability,* TCS 299(1-3), 523-535, 2003.
3. E. Formenti, and A. Grange, *Number conserving cellular automata II: dynamics,* Theoretical Computer Science, 304, 269-290, 2003.
4. K. Fujita, K. Imai, C. Iwamoto, and K. Morita, *Self-Reproduction and Logical Universality in Number-Conserving CA,* Seventh International Workshop on Cellular Automata, Hyeres, France, September 2001.
5. A. Maruoka and M. Kimura, *Condition for Injectivity of Global Maps for Tessellation Automata,* Information and Control, 32, 158-162, 1976.
6. K. Morita and K. Imai, *Number-Conserving Reversible Cellular Automata and their Computation- Universality,* Proceedings of the MFCS 98 Satellite Workshop on CA.
7. S. Takesue, *Staggered invariants in cellular automata,* Complex Systems, 9, 149-168, 1995.
8. J. Kari, *Reversibility and Surjectivity Problems of Cellular Automata,* Journal of Computer and System Sciences, 48, 149-182, 1994.
9. B. Durand, *Inversion of 2D cellular automata: some complexity results,* Theoretical Computer Science, 134, 387-401, 1994.
10. B. Durand, *The Surjectivity Problem for 2D Cellular Automata,* Journal of Computer and System Sciences, 49, 718-725, 1994.
11. A. Maruoka and M. Kimura, *Injectivity and Surjectivity of parallel maps for cellular automata,* Journal of Computer and System Sciences, 18, 158-162, 1979.
12. J. Kari, *Rice's theorem for the limit set of cellular automata,* Theoretical Computer Science, 127, 229-254, 1994.
13. K. Nagel and M. Schreckenberg, *A cellular automaton for freeway traffic,* Journal Physics I, 2, 2221-2229, 1992.

14. B. DURAND, *Global properties of cellular automata,* Cellular Automata and Complex Systems, 1998.
15. K. CULIK and S. YU, *Undecidability of CA Classification Schemes,* Complex Systems, 2, 177-190, 1988.
16. K. CULIK, J. PACHL and S. YU, *On the Limit Sets of Cellular Automata,* SIAM Journal on Computing, 18, 831-842, 1989.
17. H. Wang, Bell System Tech. Journal 40, pp. 1-42, 1961.

Captive Cellular Automata

Guillaume Theyssier

LIP (UMR CNRS, ENS Lyon, INRIA, Univ. Claude Bernard Lyon 1),
École Normale Supérieure de Lyon,
46 allée d'Italie 69364 LYON cedex 07 FRANCE
Guillaume.Theyssier@ens-lyon.fr

Abstract. We introduce a natural class of cellular automata characterised by a property of the local transition law without any assumption on the states set. We investigate some algebraic properties of the class and show that it contains intrinsically universal cellular automata. In addition we show that Rice's theorem for limit sets is no longer true for that class, although infinitely many properties of limit sets are still undecidable.

Cellular automata (CA for short) are discrete dynamical systems capable of producing a wide class of different behaviours. They consist of a large collection of simple identical components (the cells) with uniform local interactions. As such they provide an idealistic model to study complex systems observed in nature. Despite the simplicity of the model, most of the richness of behaviours they exhibit is still to be understood. Moreover, many interesting and natural properties are undecidable To that extent it is meaningful to consider classes of CA obtained by structural assumptions on the local transition law with the hope that these local assumptions are sufficiently handleable to express global dynamical properties of CA from this class.

To our knowledge, the main attempt in that sense appearing in literature is the class of additive CA first suggested by O. Martin, A. M. Odlyzko and S. Wolfram in [10]. Additive CA are those which are linear with respect to some commutative ring structure on the state set. Thanks to classical algebraic tools, such CA are now well understood on several aspects. Unfortunately, they don't reveal the richness of CA both dynamically and algorithmically (in particular, they cannot be intrinsically universal). Notice that other classes, like threshold CA studied in [5], were defined using an interpretation of the states as weights and are now well-understood.

In this paper, we define a new class of CA, namely *captive cellular automata* (CCA for short), which relies on a characterisation of their local transition law without endowing the state set with any external structure. This characterisation is based on a natural notion of sub-dynamical system in the context of CA, the notion of sub-automaton, which has often been considered and, particularly, play a central role in the algebraic classifications of CA introduced by J. Mazoyer and I. Rapaport in [11] and generalised by N. Ollinger in [13]. CCA are CA following a canonical form with respect to that notion: precisely those for

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 427–438, 2004.
© Springer-Verlag Berlin Heidelberg 2004

which the transition law is stable for any set of states. By means of this strong characterisation, we investigate properties of CCA which show that this class makes sense on several aspects.

First we prove that it possesses interesting closure properties (especially reversibility) and we fully characterise CCA which are also additive or permutive. Then, we construct for any CA a CCA capable of simulating it, what shows that there exists intrinsically universal CCA. We also show that CCA are non-trivially distributed with respect to the algebraic classification mentioned above. Finally, we consider Rice's theorem for limit sets (established by J. Kari in [8]) and show that it does not hold for the class of CCA, although an unavoidable set of infinitely many properties are still undecidable due to the ability of CCA to uniformly simulate CA.

1 Definitions and Notations

Formally, a cellular automaton $\mathcal{A}$ of dimension d (d-CA) is given by a state set A, a tuple of neighbourhood vectors $(\vec{v}_1, \ldots, \vec{v}_n)$ and a local transition map f from A^n to A. A cell is indexed by a position $\vec{z}$ of the lattice $\mathbb{Z}^d$. A configuration of $\mathcal{A}$ is a mapping from the lattice $\mathbb{Z}^d$ to the state set A giving a state to each cell. When the dimension is fixed, $A^{\mathbb{Z}^d}$ will be called the set of A-configurations. On each discrete time step, the cells alter their states synchronously according to f. This defines a mapping $\mathcal{A}$ from configurations to configurations as follows:

$$\forall c \in A^{\mathbb{Z}^d}, \forall \vec{z} \in \mathbb{Z}^d : \big(\mathcal{A}(c)\big)(\vec{z}) = f\big(c(\vec{z} + \vec{v}_1), \ldots, c(\vec{z} + \vec{v}_n)\big).$$

Global maps on configurations which are actual global maps of CA have been characterised topologically by M. L. Curtis, G. A. Hedlund and R. C. Lyndon in [6] (they give the characterisation for dimension 1, and D. Richardson extends the result to any dimension in [15]). The set of configurations endowed with the natural topology is a compact space. Then the global maps of CA are precisely the continuous maps which commute with the translations (for any $\vec{t} \in \mathbb{Z}^d$, the translation $\sigma_{\vec{t}}$ is defined by $\sigma_{\vec{t}}(c)(\vec{i} + \vec{t}) = c(\vec{i})$). Notice that this characterisation justifies the fact that the inverse map of a CA global map is itself a CA global map.

Without loss of generality we consider only CA whose neighbourhood is a ball, for the infinite norm on $\mathbb{Z}^d$, centred on (0,... ,0). When considering a CA $\mathcal{A}$, the notation $\mathcal{A}$ denotes at the same time the local transition map and the global map on configurations, A denotes the state set and $r_{\mathcal{A}}$ the radius of the ball neighbourhood of $\mathcal{A}$.

The limit set $\Omega_{\mathcal{A}}$ of a CA $\mathcal{A}$ is the set of configurations that may appear at any time step in the evolution of $\mathcal{A}$ (for a more complete study of limit sets, see [3]). Formally,

$$\Omega_{\mathcal{A}} = \bigcap_{t \in \mathbb{N}} \mathcal{A}^t\big(A^{\mathbb{Z}^d}\big).$$

It is always non-empty but it can be a singleton. It has also the important property that for any configuration $c \in \Omega_{\mathcal{A}}$, there is $d \in \Omega_{\mathcal{A}}$ such that $\mathcal{A}(d) = c$.

When $\Omega_{\mathcal{A}}$ is reduced to a single configuration, $\mathcal{A}$ is said to be *nilpotent.* The nilpotency problem is undecidable in any dimension (the first proof for dimension $d \geq 2$ appears in [3], later a proof for any dimension was given in [7]).

Let $\mathcal{A}$ be a CA and $A' \subseteq A$. The fact that A' is stable under the action of $\mathcal{A}$ (namely $\mathcal{A}((A')^{\mathbb{Z}^d}) \subseteq (A')^{\mathbb{Z}^d}$) is denoted by $A' \sqsubseteq \mathcal{A}$. The restriction of $\mathcal{A}$ to A' is then called a *sub-automaton* of $\mathcal{A}$. More generally, a CA $\mathcal{B}$ is a *sub-automaton* of $\mathcal{A}$ (also denoted by $\mathcal{B} \sqsubseteq \mathcal{A}$) if there is some $A' \sqsubseteq \mathcal{A}$ such that $\mathcal{B}$ is isomorphic to the restriction of $\mathcal{A}$ to A'.

Definition 1. *$\mathcal{A}$ is a* captive cellular automaton *(*CCA *for short) if it satisfies the following equivalent conditions:*

1. $\forall B \subseteq A : B \sqsubseteq \mathcal{A}$;
2. $\forall B \subseteq A : |B| = |A| - 1 \Rightarrow B \sqsubseteq \mathcal{A}$;
3. $\forall a_1, \ldots, a_{(2r_{\mathcal{A}}+1)^d} \in A : \mathcal{A}(a_1, \ldots, a_{(2r_{\mathcal{A}}+1)^d}) \in \{a_1, \ldots, a_{(2r_{\mathcal{A}}+1)^d}\}$.

Those equivalent conditions will often be referred to as "the jail property". It is clear that this is a locally checkable property. Notice that the identity map and more generally any shift map is trivially a CCA. Besides, a 2-state CA is a CCA if and only if each of its states is quiescent (q is quiescent if the uniform configuration $\overline{q}$ is a fixed point).

To end this section, we assume from this point on and without explicit mention that any CA considered will be one-dimensional and with at least 2 states. To simplify notations when considering a CA $\mathcal{A}$ of radius r, $\mathcal{A}$ will denote not only the local and the global map but also the action of the CA over the set of words over the alphabet with length at least $2r_{\mathcal{A}} + 1$. A word of length $2r_{\mathcal{A}} + 1$ will be called a *neighbourhood word* for $\mathcal{A}$. We denote by $L(c)$ the set of words appearing in the configuration c. Finally, for any word w let $\Sigma(w)$ denote the set of letters appearing in it.

2 First Properties

We will now present some simple properties implied by the strong structure of the local transition map of a CCA.

First it is straightforward to verify that CCA are closed under composition and sub-automata (any sub-automaton of a CCA is itself a CCA) but not by Cartesian product. An interesting and less immediate closure property is that of inversion. We give here a one-dimensional proof for clarity in notations but it is straightforward to extend it to any dimension.

Proposition 1. *Let $\mathcal{A}$ be a* CCA *which is reversible and denote by $\mathcal{A}^{-1}$ its inverse* CA. *Then $\mathcal{A}^{-1}$ is a* CCA.

Proof. Let r be the radius of $\mathcal{A}^{-1}$ and consider any $(2r+1)$-tuple of states $(x_{-r}, \ldots, x_r)$. Let c be a periodic configuration of period $x_{-r} \ldots x_r$. The sequence of configurations $\left(\mathcal{A}^n(c)\right)_{n \in \mathbb{N}}$ is periodic (the sequence cannot be only ultimately

periodic because $\mathcal{A}$ is bijective). Then there is $n \in \mathbb{N}$ such that $\mathcal{A}^n(c) = c$. Therefore $\mathcal{A}^{-1}(x_{-r} \dots x_r) \in \{x_{-r}, \dots, x_r\}$ because $\mathcal{A}^{n-1}(c) = \mathcal{A}^{-1}(c)$ and $\mathcal{A}^{n-1}$ has the jail property. □

We now give two illustrations of other constraints induced by the jail property. We characterise CCA which are respectively additive and permutive. First, for the additive case, we consider the following definition which clearly contains the linear CA of [10].

Proposition 2. *Let $\mathcal{A}$ be a* CCA. *If we can endow the states set A with a group law denoted by $+$ with neutral element 0 such that*

$$\forall c, c' \in A^{\mathbb{Z}},\ \mathcal{A}(c \overline{+} c') = \mathcal{A}(c) \overline{+} \mathcal{A}(c')$$

(where $\overline{+}$ is the uniform extension of $+$ to configurations), then either $\mathcal{A}$ is a shift map or A has only 2 states.

Proof. Let $k = 2r_{\mathcal{A}} + 1$ and for $1 \leq i \leq k$ denote by $f_i : A \to A$ the map

$$x \mapsto \mathcal{A}(\underbrace{0, \dots, 0}_{i-1}, x, \underbrace{0, \dots, 0}_{k-i}).$$

By hypothesis, $\forall x_1, \dots, x_k \in A$, $\mathcal{A}(x_1, \dots, x_k) = \sum_{i=1}^{k} f_i(x_i)$. Moreover, $\mathcal{A}$ being a CCA, $\forall x$ and $\forall i$, either $f_i(x) = 0$ or $f_i(x) = x$. Let i be fixed and consider x such that $f_i(x) = x$ and y such that $f_i(y) = 0$. Then

$$f_i(x + y) = 0 \Rightarrow x = 0 \text{ and } f_i(x + y) = x + y \Rightarrow y = 0.$$

So either $f_i = Id$ or $f_i = 0$. Let now suppose that the group A has an element a such that $a + a \neq a$ and $a + a \neq 0$. Then there is at most one i such that $f_i = Id$. Indeed, if $f_i = f_j = Id$ with $i \neq j$, $\mathcal{A}$ cannot satisfy the jail property on the input $x_1, \dots, x_k$ with $x_i = x_j = a$ and $x_m = 0$ for $m \neq i$, $m \neq j$. In this case, either $\mathcal{A}$ is a shift, or the constant map equal to 0, but a CCA with a least 2 states cannot be constant.

Otherwise, each element $a \neq 0$ of A is of order 2. Then if $\mathcal{A}$ is neither a shift nor the constant map equal to 0 (*i.e.* if there is $i \neq j$ such that $f_i = f_j = Id$), A cannot contain 2 distinct elements a and b different from 0 because $\mathcal{A}$ would not satisfy the jail property on the entry $x_1, \dots, x_k$ with $x_i = a$, $x_j = b$ and $x_m = 0$ for $m \neq i$, $m \neq j$: $a + b$ can neither be 0 (otherwise $a = -b = b$), nor a (otherwise $b = 0$), nor b (otherwise $a = 0$). Thus A has only 2 elements. □

The notion of permutivity was essentially studied through LR-permutive CA which are simple examples of expansive CA (see [4] for example). A proposition similar to proposition 2 for permutivity can be stated as follows.

Proposition 3. *A* CA $\mathcal{A}$ *is said to be* permutive at position i *$(1 \leq i \leq 2r + 1)$ if for all $u \in A^{i-1}$ and all $v \in A^{2r+1-i}$ the map $\pi_{i,u,v} : x \mapsto \mathcal{A}(uxv)$ is bijective from A to A. If $\mathcal{A}$ is a* CCA *which is permutive at 2 positions $i \neq j$, then $\mathcal{A}$ has only 2 states.*

Proof. Without loss of generality one can suppose $i < j$. Consider $u \in A^{i-1}$, $v \in A^{2r+1-i}$ and $P = \Sigma(uv)$. Let I denote the set $\pi^{-1}_{i,u,v}(P)$. The map $\pi^{-1}_{i,u,v}$ is bijective by hypothesis so $|P| = |I|$. Consider $x \in I \setminus P$. We have $x \notin P$ and $\pi^{-1}_{i,u,v}(x) \neq x$ (because $\pi_{i,u,v}(x) \in P$), then $x = \mathcal{A}\big(u\pi^{-1}_{i,u,v}(x)v\big) \notin \Sigma\big(u\pi^{-1}_{i,u,v}(x)v\big)$ which contradicts the jail property. Thus $I \setminus P = \emptyset$ and hence $I = P$. Therefore for any $a \notin P$, $\mathcal{A}(uav) \notin P$ because $a \notin I$, so $\mathcal{A}(uav) = a$ by the jail property. The same is true at position j.

Suppose now a, b, c are 3 different elements in A; according to what was shown above $a = \mathcal{A}(b^{i-1}ab^{j-i-1}cb^{2r+1-j}) = c$: contradiction. So $|A| = 2$. □

3 Simulations and Universality

From previous section, CCA appear to be very constrained and could constitute a marginal class of CA (as additive CA). In this section we will show that it is not the case: CCA actually exhibit as much richness and complexity as CA in general. To give a precise sense to the latter assertion, we rely on the algebraic framework introduced by J. Mazoyer and I. Rapaport ([11]) and extended by N. Ollinger ([14]). It consists in a quasi-order over CA interpreted as a simulation relation: a CA can simulate another if the second is a sub-automaton of the first, up to "rescaling" of both. It naturally induces an equivalence relation and an order on equivalence classes which provides an interesting formal tool to compare CA. The generalised notion of rescaling introduced by N. Ollinger leads to an order with a global maximum which correspond to the set of intrinsically universal CA, that is CA which can simulate any CA. Notice that this notion captures intuitive ideas already present in the work of E. R. Banks (see [2]) and formalised for the first time by J. Albert and K. Čulik in [1]. We refer the reader to [13] (in French) or [12] for a deeper study of such ideas.

The notion of rescaling can be uniformly formalised in any dimension but for clarity we give only the definition for one-dimensional CA. For any CA $\mathcal{A}$ and any positive integer m denote by o^m the "packing" map from A-configurations to A^m-configurations:

$$\forall c \in A^{\mathbb{Z}}, \forall z \in \mathbb{Z} : \big(o^m(c)\big)(z) = \big(c(mz), c(mz+1), \ldots, c(m(z+1)-1)\big).$$

For any CA $\mathcal{A}$ and any $k \in \mathbb{Z}$, $m > 0$ and $n > 0$ let $\mathcal{A}^{<m,n,k>}$ denote the (m,n,k)-rescaling of $\mathcal{A}$ defined as follows: $\mathcal{A}^{<m,n,k>} = \sigma^k \circ o^m \circ \mathcal{A}^n \circ (o^m)^{-1}$. Then $\mathcal{A}$ can simulate $\mathcal{B}$ (denoted by $\mathcal{A} \preceq \mathcal{B}$) if there exist $k, k' \in \mathbb{Z}$ and $m, m', n, n' > 0$ such that $\mathcal{B}^{<m,n,k>} \sqsubseteq \mathcal{A}^{<m',n',k'>}$. Let us now show how any CA can be simulated in that precise sense by an appropriate CCA.

Definition 2. *Let $\mathcal{A}$ be a* CA *of radius r and state set $A = \{a_1, \ldots, a_n\}$ and let $\#$ be an additional state not in A. Let $w = w_1 w_2 \ldots w_{n+2}$ denote the word $\#a_1a_2\ldots a_n\#$. Then $\kappa_{\#,A}$ denote the following injective coding map from $A^{\mathbb{Z}}$ to $(A \cup \{\#\})^{\mathbb{Z}}$:*

$$\big(\kappa_{\#,A}(c)\big)(z) = \begin{cases} c(k) & \text{if } z = k(n+3) \text{ for } k \in \mathbb{Z}, \\ w_i & \text{otherwise where } i = z \bmod (n+3). \end{cases}$$

We define $\rho_{\#}(\mathcal{A})$, *a* CA *of radius* $r(n+3)+n+2$ *and state set* $A \cup \{\#\}$. *It acts as follows on any neighbourhood word* $u = u_{-r(n+3)-n-2} \cdots u_{r(n+3)+n+2}$:

$$\big(\rho_{\#}(\mathcal{A})\big)(u) = \begin{cases} \mathcal{A}(c_{-r} \dots c_0 \dots c_r) & \text{if } u = wc_{-r}w \dots wc_iw \dots wc_rw, \\ \# & \text{if } u \notin L\big(\kappa_{\#,A}(A^{\mathbb{Z}})\big) \text{ and } \# \in \Sigma(u) \\ u_0 & \text{otherwise.} \end{cases}$$

Informally, $\kappa_{\#,A}(c)$ consist in the bi-infinite concatenation of macro-cells of the form $m_c((i)) = c(i)\#a_1 \cdots a_n\#$, where $m_c(i)$ begins at position $i(n+3)$ in $\kappa_{\#,A}(c)$ and encodes the cell at position i in the configuration c.

A word for $\rho_{\#}(\mathcal{A})$ is said to be *valid* if it is a sub-word of $\kappa_{\#,A}(c)$ for some $c \in A^{\mathbb{Z}}$. In such a word, a letter is said to be *informative* if it is the first letter of a macro-cell. A valid neighbourhood word is *active* if its central cell is informative. Here is an illustration of those notions for $n = 2$ and $r = 1$:

$$\underbrace{\#a_1a_2\#a_1\#a_1a_2\#a_2\#\overbrace{a_1a_2\#a_2\#a_1a_2\#}^{\text{valid but not active}}a_1\#a_1a_2\#a_1\#a_1a_2\#a_2}_{\text{active}}\#a_1a_2\underbrace{\#a_2\#a_2}_{\text{non-valid}}$$

Then $\rho_{\#}(\mathcal{A})$ acts as follows on a neighbourhood word u:

- if u is active then $\rho_{\#}(\mathcal{A})$ mimics $\mathcal{A}$ using informative letters present in u;
- if u is not valid and contains # then the cell state becomes #;
- no change is done in other cases.

It is thus clear that $\rho_{\#}(\mathcal{A})$ is a CCA: updates of the cell state are done in such a way that the new cell state was already present in the neighbourhood word. The construction gives raise to simulation of any CA by a CCA (and it can be straightforwardly extend to higher dimensions).

Proposition 4. *For any* CA $\mathcal{A}$, *we have* $\mathcal{A} \preceq \rho_{\#}(\mathcal{A})$.

Proof. We actually show that $\mathcal{A} \sqsubseteq \rho_{\#}(\mathcal{A})^{<n+3,1,0>}$. By construction, we have the following commutative diagram:

$$\begin{array}{ccccc} A^{\mathbb{Z}} & \xrightarrow{\kappa_{\#,A}} & \kappa_{\#,A}(A^{\mathbb{Z}}) & \xrightarrow{\sigma^{n+3}} & M^{\mathbb{Z}} \\ \downarrow{\scriptstyle \mathcal{A}} & & \downarrow{\scriptstyle \rho_{\#}(\mathcal{A})} & & \downarrow{\scriptstyle \rho_{\#}(\mathcal{A})^{<n+3,1,0>}} \\ A^{\mathbb{Z}} & \xrightarrow{\kappa_{\#,A}} & \kappa_{\#,A}(A^{\mathbb{Z}}) & \xrightarrow{\sigma^{n+3}} & M^{\mathbb{Z}} \end{array}$$

where M denotes the set of all macro-cells in the sense of the coding map $\kappa_{\#,A}$. The construction guaranties that $M \sqsubseteq \rho_{\#}(\mathcal{A})^{<n+3,1,0>}$. □

We deduce from proposition 4 that CCA reach the top of the order on CA, that is there exists intrinsically universal CCA. We can also show that the order, restricted to the classes containing CCA, is still rich. More precisely, it contains infinite chains and admits any finite tree as a sub-order. These two facts are straightforward corollaries of the following proposition.

Proposition 5. *There exists a family* $(\mathcal{B}_{m,n})_{m,n\in\mathbb{N},n\geq m}$ *of* CCA *such that:*

- $(m \leq m') \wedge (n \leq n') \Rightarrow \mathcal{B}_{m,n} \preceq \mathcal{B}_{m',n'}$;
- $(n < n') \vee (m < m') \Rightarrow \mathcal{B}_{m',n'} \not\preceq \mathcal{B}_{m,n}$

Proof. For each m, n let $A_{m,n} = \{1, \ldots, m+n\}$ and $\mathcal{A}_{m,n}$ be a CCA of radius 5 on the state set $A_{m,n} \cup \{\#\}$ defined hereafter. Let $w = w_1w_2w_3w_4$ be the word $\#m\#\#$. Among configurations of $\mathcal{A}_{m,n}$ we distinguishes the following set of configurations: $S^0_{m,n} = \{c \in (A_{m,n} \cup \{\#\})^{\mathbb{Z}} : c(i) = w_{i \bmod 4} \text{ if } i \neq 0 \bmod 4\}$. Let $S_{m,n} = \bigcup_{i\in\mathbb{Z}} \sigma^i(S^0_{m,n})$. Then $\mathcal{A}_{m,n}$ is defined by

$$\mathcal{A}_{m,n}(u_{-5}\ldots u_5) = \begin{cases} \max(x,y,z) & \text{if } u = xwywz \text{ and } \max(x,y,z) \geq m \\ x & \text{if } u = xwxwx \text{ and } x \leq m \\ m & \text{otherwise if } u = xwywz \text{ and } x,y,z \leq m \\ u_0 & \text{otherwise if } u \in L(S_{m,n}) \\ \# & \text{if } u \notin L(S_{m,n}) \text{ and if } \# \in u \\ \min(u_i) & \text{in any other case.} \end{cases}$$

Now, for any CA $\mathcal{A}$, let $Y(\mathcal{A})$ be the set of configurations of $\mathcal{A}$ belonging to a cyclic orbit, and let $U(\mathcal{A})$ be the set of configurations of $Y(\mathcal{A})$ with a unique predecessor.

We claim that $\left|Y(\mathcal{A}^{<a,b,c>}_{m,n})\right| = 6(m+n)+1$ and $\left|U(\mathcal{A}^{<a,b,c>}_{m,n})\right| = 5(m-1)$ for any rescaling parameters a, b and c. To complete the proof from that claim it is sufficient to notice that:

1. $m \leq m'$ and $n \leq n' \Rightarrow \mathcal{A}_{m,n} \sqsubseteq \mathcal{A}_{m',n'}$;
2. for any CA $\mathcal{A}$ and $\mathcal{B}$: $\mathcal{A} \sqsubseteq \mathcal{B} \Rightarrow \begin{cases} |Y(\mathcal{A})| \leq |Y(\mathcal{B})| \text{ and} \\ |Y(\mathcal{A}) \setminus U(\mathcal{A})| \leq |Y(\mathcal{B}) \setminus U(\mathcal{B})| \end{cases}$.

Then define $\mathcal{B}_{m,n} = \mathcal{A}_{m,n-m}$.

To prove the claim, first notice that rescaling a CA $\mathcal{A}$ does not change the cardinalities of the sets Y and U. Thus the proof is brought down to enumerating $Y(\mathcal{A}_{m,n})$ and $U(\mathcal{A}_{m,n})$. This is a straightforward case study from the definition of $\mathcal{A}_{m,n}$. The reader can verify that configurations from the sets $S_{m,n}$, $A^{\mathbb{Z}}_{m,n}$ and $X = \{c : c \notin S_{m,n} \cup A_{m,n}\}$ have disjoint orbits. Thus, considering $\mathcal{A}_{m,n}$ on each of these sets separately, we can easily infer :

- $Y(\mathcal{A}_{m,n}) \cap S_{m,n} = \{c : c \text{ is periodic of period } xw \text{ with } x \in A_{m,n}\}$;
- $U(\mathcal{A}_{m,n}) \cap S_{m,n} = \{c : c \text{ is periodic of period } xw \text{ with } x \in A_{m,n}, x < m\}$;
- $Y(\mathcal{A}_{m,n}) \cap A^{\mathbb{Z}}_{m,n} = \{\overline{x} : x \in A_{m,n}\}$;
- $Y(\mathcal{A}_{m,n}) \cap X = \{\overline{\#}\}$;
- $U(\mathcal{A}_{m,n}) \cap A^{\mathbb{Z}}_{m,n} = U(\mathcal{A}_{m,n}) \cap X = \emptyset$.

To conclude the proof, just notice that for any $x \in A_{m,n}$ there are exactly 5 periodic configurations of period xw. □

4 Decidability

We will now present some decidability results. The situation concerning CCA is somewhat balanced because on one hand many undecidable properties for CA are preserved under the coding schemes introduced in section 3, and on the other hand the nilpotency problem used in many proofs by reduction becomes trivial in the context of CCA (a CCA $\mathcal{A}$ has at least 2 configurations in its limit set because for any state s the uniform configuration $\overline{s}$ is a fixed point.) We now give precise decidability results.

We will focus on an analog of Rice's theorem for CA limit sets established by J. Kari in [8]. It states that, in any dimension, any non-trivial property of limit sets is undecidable. More precisely J. Kari shows that the nilpotency problem can be reduced to any decision problem concerning a non-trivial property of limit sets. As said before, there are no nilpotent CCA. Thus it is natural to ask what Kari's theorem becomes when restricted to CCA. We are going to show that there is as many undecidable properties for CCA as undecidable properties for CA in general. This result actually relies on the properties of the maps $\kappa_{\#,\mathcal{A}}$ and $\rho_{\#}$, while the jail property is not directly used in the proof. So we present it in the general framework of any set of CA. Before stating the theorem let us formalise the notion of *property of limit sets.*

To avoid irrelevant set theoretical problems (due to renaming of states), we will temporally assume like in [8] that any CA state set comes from a countably infinite set of states $S = \{s_0, s_1, \ldots\}$. For any finite set $T \subseteq S$ let P_T denote the set of all subsets of $T^{\mathbb{Z}}$ (P_T contains all possible limit sets involving only states from T). Finally let U be the infinite union of sets P_T for all finite T. A property $\mathcal{P}$ of limit sets is a subset of U and a CA $\mathcal{A}$ has the property $\mathcal{P}$ (denoted by $\mathcal{A} \vdash \mathcal{P}$) if $\Omega_{\mathcal{A}} \in \mathcal{P}$.

Theorem 1. *Let E be a set of* CA *such that there are maps ρ and $\kappa_{\mathcal{A}}$ with the following properties:*

- *ρ is a computable map from the set of* CA *to E ;*
- *for any* CA *$\mathcal{A}$, $\kappa_{\mathcal{A}}$ maps the configurations of $\mathcal{A}$ to configurations of $\rho(\mathcal{A})$ and the following diagram commutes:*

$$\begin{array}{ccc} A^{\mathbb{Z}} & \xrightarrow{\kappa_{\mathcal{A}}} & \kappa_{\mathcal{A}}(A^{\mathbb{Z}}) \\ \downarrow{\scriptstyle \mathcal{A}} & & \downarrow{\scriptstyle \rho(\mathcal{A})} \\ A^{\mathbb{Z}} & \xrightarrow{\kappa_{\mathcal{A}}} & \kappa_{\mathcal{A}}(A^{\mathbb{Z}}) \end{array}$$

- *κ maps are overall injective: $\kappa_{\mathcal{A}}(c) = \kappa_{\mathcal{B}}(d) \Rightarrow \kappa_{\mathcal{A}} = \kappa_{\mathcal{B}}$ and $c = d$;*
- *κ maps are honest: if $\rho(\mathcal{A})(d) = c$ then $c \in \kappa_{\mathcal{B}}(B^{\mathbb{Z}}) \Rightarrow d \in \kappa_{\mathcal{A}}(A^{\mathbb{Z}})$.*

Then there is an injective map Φ on U such that, for any non-trivial property $\mathcal{P}$, $\Phi(\mathcal{P})$ is an undecidable property when restricted to the set E of input CA.

Proof. The map Φ is defined as follows:

$$\Phi(\mathcal{P}) = \bigcup_{\mathcal{A}} \{P_+ \in U : P_+ \cap \kappa_{\mathcal{A}}(A^{\mathbb{Z}}) \in \{\kappa_{\mathcal{A}}(P) : P \in \mathcal{P}, P \neq \emptyset\}\}.$$

First notice that Φ is injective because if $\mathcal{P}_a \neq \mathcal{P}_b$ in U, there is $P_a \in \mathcal{P}_a$, $P_a \notin \mathcal{P}_b$ (the proof is the same if $\mathcal{P}_b \setminus \mathcal{P}_a \neq \emptyset$). Let $\mathcal{A}$ be a CA such that $P_a \subseteq A^{\mathbb{Z}}$ and let $K = \kappa_{\mathcal{A}}(P_a)$. Clearly $K \in \Phi(\mathcal{P}_a)$. Suppose $K \in \Phi(\mathcal{P}_b)$. Then there are $P \in \mathcal{P}_b$, $P \neq \emptyset$ and a CA $\mathcal{B}$ such that $K \cap \kappa_{\mathcal{B}}(B^{\mathbb{Z}}) = \kappa_{\mathcal{B}}(P)$. By overall injectivity of κ maps, this implies $\mathcal{A} = \mathcal{B}$. Therefore $\kappa_{\mathcal{A}}(P) = K$ and hence, by injectivity of $\kappa_{\mathcal{A}}$, $P_a = P \in \mathcal{P}_b$: contradiction.

Let us now show that $\mathcal{A} \vdash \mathcal{P}$ if and only if $\rho(\mathcal{A}) \vdash \Phi(\mathcal{P})$ (what is sufficient to complete the proof by Kari's theorem since ρ is computable). First suppose $\Omega_{\mathcal{A}} \in \mathcal{P}$. We have by the commutative diagram $\kappa_{\mathcal{A}}(\Omega_{\mathcal{A}}) \subseteq \Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{A}}(A^{\mathbb{Z}})$. Conversely for any $c \in \Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{A}}(A^{\mathbb{Z}})$ there is $d \in \Omega_{\rho(\mathcal{A})}$ such that $\big(\rho(\mathcal{A})\big)(d) = c$ (by the property of limit sets mentioned in section 1). Therefore by the honesty property $d \in \kappa_{\mathcal{A}}(A^{\mathbb{Z}})$. We can repeat this argument to construct an infinite history of configurations in $\kappa_{\mathcal{A}}(A^{\mathbb{Z}})$ leading to c by $\rho(\mathcal{A})$. By the commutative diagram, we deduce that $c \in \kappa_{\mathcal{A}}(\Omega_{\mathcal{A}})$. Thus $\Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{A}}(A^{\mathbb{Z}}) \subseteq \kappa_{\mathcal{A}}(\Omega_{\mathcal{A}})$ and hence, $\Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{A}}(A^{\mathbb{Z}}) = \kappa_{\mathcal{A}}(\Omega_{\mathcal{A}})$. Therefore $\Omega_{\rho(\mathcal{A})} \in \Phi(\mathcal{P})$ by definition of Φ.

Suppose now that $\Omega_{\rho(\mathcal{A})} \in \Phi(\mathcal{P})$. Then there are $P \in \mathcal{P}$, $P \neq \emptyset$ and a CA $\mathcal{B}$ such that $\Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{B}}(B^{\mathbb{Z}}) = \kappa_{\mathcal{B}}(P)$. As above, for any $c \in \Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{B}}(B^{\mathbb{Z}})$, we have $d \in \Omega_{\rho(\mathcal{A})}$ such that $\big(\rho(\mathcal{A})\big)(d) = c$. By the honesty property, this implies $d \in \kappa_{\mathcal{A}}(A^{\mathbb{Z}})$ and $c \in \kappa_{\mathcal{A}}(A^{\mathbb{Z}})$. Thus, by overall injectivity, $\kappa_{\mathcal{A}} = \kappa_{\mathcal{B}}$ (and therefore $A = B$). We have shown above that $\Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{A}}(A^{\mathbb{Z}}) = \kappa_{\mathcal{A}}(\Omega_{\mathcal{A}})$. So we have the following:

$$\kappa_{\mathcal{A}}(\Omega_{\mathcal{A}}) = \Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{A}}(A^{\mathbb{Z}}) = \Omega_{\rho(\mathcal{A})} \cap \kappa_{\mathcal{B}}(B^{\mathbb{Z}}) = \kappa_{\mathcal{B}}(P) = \kappa_{\mathcal{A}}(P).$$

By injectivity of $\kappa_{\mathcal{A}}$, we deduce that $\Omega_{\mathcal{A}} = P \in \mathcal{P}$. □

Let us now show how to construct κ and ρ maps for CCA. Some technical points must be made more precise. First, the macro-cells used in the construction of $\kappa_{\#,\mathcal{A}}$ rely on an enumeration of the state set A. Given that any CA considered here take its states from S we will always use enumerations following the order of elements in S. We will also shift the states to free s_0 which will play the role of the special state #. Formally, let s denote the successor map on S ($s(s_i) = s_{i+1}$) and $\overline{s}$ its extension to configurations. ρ and κ maps are then defined as follows

$$\rho(\mathcal{A}) = \rho_{s_0}\big(\overline{s} \circ \mathcal{A} \circ \overline{s}^{-1}\big), \qquad \kappa_{\mathcal{A}} = \kappa_{s_0,\mathcal{A}} \circ \overline{s}.$$

Lemma 1. *The maps defined above fill the hypotheses of theorem 1 for the set of* CCA.

Proof. First, it is clear that ρ is computable and proposition 4 shows that the diagram in the hypotheses commutes. Let us show that κ maps are overall injective and honest. Let $\mathcal{A}$ and $\mathcal{B}$ be two CA on state sets $A = \{a_1, \ldots, a_k\}$ and $B = \{b_1, \ldots, b_l\}$ enumerated in increasing order according to S.

Suppose that $c \in A^{\mathbb{Z}}$ and $d \in B^{\mathbb{Z}}$ are such that $\kappa_{\mathcal{A}}(c) = \kappa_{\mathcal{B}}(d)$. Necessarily $k = l$ because if $k < l$ (the other case is symmetric) then $\big(\kappa_{\mathcal{A}}(c)\big)(k+2) = s_0$ and $\big(\kappa_{\mathcal{B}}(d)\big)(k+2) = b_{k+1}$, but $s_0 \neq b_{k+1}$ because $b_{k+1} > b_1 \geq s_0$. Then equality of configurations $\kappa_{\mathcal{A}}(c)$ and $\kappa_{\mathcal{B}}(d)$ at positions 2 to $k+1$ implies $A = B$ and therefore $\kappa_{\mathcal{A}} = \kappa_{\mathcal{B}}$. Finally, $c = d$ because $\kappa_{\mathcal{A}}(c)$ and $\kappa_{\mathcal{B}}(d)$ are equal at position $z(k+3)$ for any $z \in \mathbb{Z}$. This shows overall injectivity. For honesty, suppose that $c \in \kappa_{\mathcal{B}}(B^{\mathbb{Z}})$ and $d \in \big(s(A) \cup \{s_0\}\big)^{\mathbb{Z}}$ are such that $\rho(\mathcal{A})\big(d\big) = c$. Let $\mathcal{R} = \rho(\mathcal{A})$. First $\mathcal{R}$ has the following property :

$$(\mathcal{R}(x)(i-1) \neq s_0) \wedge (\mathcal{R}(x)(i) = s_0) \wedge (\mathcal{R}(x)(i+1) \neq s_0) \Rightarrow x(i) = s_0.$$

Indeed, if $x(i) \neq s_0$ and $\mathcal{R}(x)(i) = s_0$ the the neighbourhood of cell i in x is non-valid and contains a s_0. Then either cell $i-1$ or cell $i+1$ has a non-valid neighbourhood which contains s_0 (it is straightforward to verify): thus either $\mathcal{R}(x)(i-1) = s_0$ or $\mathcal{R}(x)(i+1) = s_0$. We deduce that $c(i) = s_0 \Rightarrow d(i) = s_0$ because $s_0 s_0 \notin L(c)$ (c is a valid configuration for $\kappa_{\mathcal{B}}$). In particular, any neighbourhood word for $\mathcal{R}$ in d contains s_0. Let us suppose that $d \notin \kappa_{\mathcal{A}}(A^{\mathbb{Z}})$. Then there is $z \in \mathbb{Z}$ such that position z in d is non-valid and contains s_0 in its neighbourhood. Therefore $\mathcal{R}(d)(z) = c(z) = s_0$ and, according to what was shown above, $d(z) = s_0$. Either position $z-1$ or $z+1$ is non-valid and has s_0 in its neighbourhood (precisely at position z): in both case we have exhibited an occurrence of the word $s_0 s_0$ in c which contradicts its validity. □

This lemma together with theorem 1 shows that there is an infinite number of undecidable properties of limit sets for CCA. However, the analog of Rice's theorem is no longer true for CCA. For instance, the property "an even number of different states appears in the limit set" is clearly decidable for CCA although non-trivial, simply because each state of a CCA appears in its limit set.

Let us now present a natural problem on limit sets which is undecidable for CA but decidable for CCA. We no longer follows the hypothesis of states being taken in S. Let $\mathcal{A}$ be a fixed CA of radius r.

Proposition 6. *Consider the following decision problems* :

($\mathcal{P}_1$) *Input : a* CA $\mathcal{B}$ *of radius* r. *Question* : $\Omega_{\mathcal{A}} = \Omega_{\mathcal{B}}$?
($\mathcal{P}_2$) *Input : a* CCA $\mathcal{B}$ *of radius* r. *Question* : $\Omega_{\mathcal{A}} = \Omega_{\mathcal{B}}$?

$\mathcal{P}_1$ *is undecidable whereas* $\mathcal{P}_2$ *is decidable.*

Proof. First let X denote the set of states appearing in $\Omega_{\mathcal{A}}$. Notice that a CCA $\mathcal{B}$ is such that $\Omega_{\mathcal{B}} = \Omega_{\mathcal{A}}$ only if $B = X$. Thus there is only a finite number of possible CCA having the same limit set as $\mathcal{A}$ and the problem $\mathcal{P}_2$ is trivially decidable. Now we prove that $\mathcal{P}_1$ is undecidable. We proceed by a reduction from the nilpotency problem which is proven undecidable in [7] even if we restrict ourselves to CA of radius r with a spreading state (a state q is spreading for $\mathcal{N}$ if $\mathcal{N}(u) = q$ for any neighbourhood word u containing an occurrence of q). Now given any CA $\mathcal{N}$ of radius r with a spreading state q, we can algorithmically construct a CA $\mathcal{D}$ as follows. First (up to renaming) we can suppose $A \cap N = \emptyset$.

Let $D = A \cup (N \setminus \{q\})$ *be* the state set of $\mathcal{D}$. Let f and g denote the following maps from D to N and form N to D (respectively), where $a_0 \in A$ is fixed:

$$f : x \mapsto \begin{cases} q & \text{if } x \in A, \\ x & \text{otherwise,} \end{cases} \qquad g : y \mapsto \begin{cases} a_0 & \text{if } y = q, \\ y & \text{otherwise.} \end{cases}$$

Then $\mathcal{D}$ is defined by

$$\mathcal{D}(x_{-r}, \ldots, x_r) = \begin{cases} \mathcal{A}(x_{-r}, \ldots, x_r) & \text{if } x_i \in A, -r \leq i \leq r, \\ g(\mathcal{N}(f(x_{-r}), \ldots, f(x_r))) & \text{otherwise.} \end{cases}$$

Then $\Omega_{\mathcal{D}} = \Omega_{\mathcal{A}}$ if and only if $\mathcal{N}$ is nilpotent. Indeed, if $\mathcal{N}$ is nilpotent there is n_0 such that for all c, $\mathcal{N}^{n_0}(c) = \overline{q}$. Therefore $\mathcal{D}^{n_0}(D^{\mathbb{Z}}) \subseteq A^{\mathbb{Z}}$ and $\Omega_{\mathcal{D}} \subseteq \Omega_{\mathcal{A}}$. So $\Omega_{\mathcal{D}} = \Omega_{\mathcal{A}}$ because $\mathcal{D}$ is equal to $\mathcal{A}$ on $A^{\mathbb{Z}}$. Conversely, if $\mathcal{N}$ is not nilpotent, then there is $c \neq \overline{q}$ in $\Omega_{\mathcal{N}}$. Then for any $t \in \mathbb{N}$ there is $c_t \in N^{\mathbb{Z}}$ such that $\mathcal{N}^t(c_t) = c$. Let $d_t = \overline{g}(c_t)$ (where $\overline{g}$ is the uniform extension of g to configurations). Clearly for any $i \in \mathbb{Z}$ such that $c(i) \neq q$, we have $\left(\mathcal{D}^t(d_t)\right)(i) \notin A$. By compactness, there is $d \in D^{\mathbb{Z}}$ with infinite history for $\mathcal{D}$ and such that $d \notin A^{\mathbb{Z}}$: thus $\Omega_{\mathcal{D}}$ cannot be equal to $\Omega_{\mathcal{A}}$. □

5 Conclusion and Perspectives

The model of CA is widely admitted as a relevant framework to study questions raised by the paradigm of complex systems. Unfortunately, most of the interesting properties concerning their behaviours are undecidable. It is then natural to consider sub-classes of CA with the hope that classical problems and generally behaviours classification will be easier when restricted to that class. CCA introduced here constitute such a class, with both the property that some undecidable problems become decidable when restricted to the class (section 4) and that it somehow preserves the complexity of CA (section 3).

We defined CCA by a formal property of the local transition law. A remarkable fact is that reversible CCA satisfy the property both forwards and backwards. Is there a topological proof of the stability by inversion of the jail property? And more generally is there a topological characterisation of CCA?

Besides, we emphasise the non-closure of CCA by Cartesian product and a noticeable consequence: the classical simulation of any CA by a one-way CA cannot be directly transfered to CCA. More generally, the role played by the radius and the number of states are highly non-symmetric in the context of CCA, contrary to the general case. Is there a hierarchy of complexity for CCA according to the radius?

We also showed that undecidability remains highly present in CCA. The frontier between decidability and undecidability is thus still to be made more precise. Particularly, are classical properties like that of reaching the limit set in finite time or being reversible in dimension 2 still undecidable for CCA?

Finally, experiments[1] on randomly chosen CCA show particular dynamics (for example, equicontinuity points appears very often) and propositions 2 and 3 suggests that "chaotic-like" behaviours are extremely constrained for CCA. Can we give a more precise sense to this assertion? For instance, can we explicit the intersections of CCA with Kůrka's classes (introduced in [9])?

Acknowledgement. I would like to thank M. Delorme and J. Mazoyer for helpful discussions and carefull reading of the paper.

References

1. J. Albert and K. Čulik, II. A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems,* 1:1–16, 1987.
2. E. R. Banks. Universality in cellular automata. In *Eleventh Annual Symposium on Switching and Automata Theory,* Santa Monica, California, 1970. IEEE.
3. Karel Čulik, II, Jan Pachl, and Sheng Yu. On the limit sets of cellular automata. *SIAM Journal on Computing,* 18(4):831–842, August 1989.
4. F. Fagnani and Luciano Margara. Expansivity, permutivity, and chaos for cellular automata. *Theory of Computing Systems,* 31(6):663–677, 1998.
5. E. Goles and J. Olivos. The convergence of symmetric threshold automata. *Information and Control,* 51:98–104, 1981.
6. G. A. Hedlund. Endomorphisms and Automorphisms of the Shift Dynamical Systems. *Mathematical Systems Theory,* 3(4):320–375, 1969.
7. J. Kari. The Nilpotency Problem of One-dimensional Cellular Automata. *SIAM Journal on Computing,* 21:571–586, 1992.
8. J. Kari. Rice's theorem for the limit sets of cellular automata. *Theoretical Computer Science,* 127:229–254, 1994.
9. P. Kůrka. Languages, equicontinuity and attractors in cellular automata. *Ergodic Theory and Dynamical Systems,* 17:417–433, 1997.
10. O. Martin, A. Odlyzko, and S. Wolfram. Algebraic properties of cellular automata. *Communications in Mathematical Physics,* 93:219, 1984.
11. J. Mazoyer and I. Rapaport. Inducing an Order on Cellular Automata by a Grouping Operation. In *Symposium on Theoretical Aspects of Computer Science.* Lecture Notes in Computer Science, 1998.
12. N. Ollinger. Toward an algorithmic classification of cellular automata dynamics. Reasearch report 2001-10, Laboratoire de l'Informatique du Parallèlisme, January 2001.
13. N. Ollinger. *Automates Cellulaires : structures.* PhD thesis, École Normale Supérieure de Lyon, décembre 2002.
14. Nicolas Ollinger. The quest for small universal cellular automata. In *International Colloquium on Automata, Languages and Programming,* pages 318–330. Lecture Notes in Computer Science, 2002.
15. D. Richardson. Tesselations with local tranformations. *Journal of Computer and System Sciences,* 5:373–388, 1972.

[1] See `http://captive.ca.free.fr` for random space-time diagrams of CCA

Simulating 3D Cellular Automata with 2D Cellular Automata

Victor Poupet

LIP (UMR CNRS, ENS Lyon, INRIA, Univ. Claude Bernard Lyon 1),
École Normale Supérieure de Lyon,
46 allée d'Italie 69364 LYON cedex 07 FRANCE

Abstract. The purpose of this article is to describe a way to simulate any 3-dimensional cellular automaton with a 2-dimensional cellular automaton. We will present the problems that arise when changing the dimension, propose solutions and discuss the main properties of the obtained simulation.

1 Introduction

Cellular automata are a widely studied and massively parallel computing model. They are composed of *cells,* arranged on $\mathbb{Z}^n$ (where n is the *dimension* of the automaton). The cells can be in different *states,* the set of states being finite and common to all cells. They *evolve* synchronously in a deterministic way at discrete times. All cells evolve according to the same local rule, and their new state depends only on their neighbors' states. From this local behavior, we can define a global evolution, from any configuration of the automaton into another, by having each cell change its state according to the local rule.

Despite their apparent simplicity cellular automata can have very complex behaviors and are commonly used in order to modelize various phenomena. The underlying graph is usually $\mathbb{Z}^2$ or $\mathbb{Z}^3$ (depending on the space of the modelized reality). Computer networks however are often massively parallel computers working on $\mathbb{Z}$ or $\mathbb{Z}^2$ in which $\mathbb{Z}^3$ cannot be embeded easily. Various attempts have been made to simulate a $\mathbb{Z}^3$ space on a $\mathbb{Z}^2$ network: for example using hexagonal networks (see [1]) but in this case a degree of liberty is lost [2]. These simulations also lead to a problem of representation of the usual $\mathbb{Z}^3$ shapes (lines, spheres, planes, cones etc.) in $\mathbb{Z}^2$ without loss of information.

The subject matter of this article is to show how it is possible, given a 3-dimensional cellular automaton (3DCA) $\mathcal{A}_3$, to construct a 2-dimensional cellular automaton (2DCA) $\mathcal{A}_2$ that will *mimic* its behavior, in a sense that we will define. To do so, we want to associate to each cell of $\mathcal{A}_3$ a unique cell of $\mathcal{A}_2$ with a one-to-one function t, so that $t(c)$ will simulate the evolution of c. Considering the relative sizes of the spheres of radius k in $\mathbb{Z}^3$ and $\mathbb{Z}^2$, it is obvious that there will be cells c_1 and c_2 that are very close in $\mathbb{Z}^3$ and such that $t(c_1)$ and $t(c_2)$ are arbitrarily far in $\mathbb{Z}^2$ (see [3]). For this reason, it won't be possible to execute the simulation in linear time, but we will show that it is possible to have $\mathcal{A}_2$ compute k generations of $\mathcal{A}_3$ in polynomial time.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 439–450, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In this article, for matters of simplicity, we will consider that $\mathcal{A}_3$ works on the 3D Von Neumann neighborhood $\mathcal{V}_3 = \{\vec{0}, \pm\vec{x}, \pm\vec{y}, \pm\vec{z}\}$ and that $\mathcal{A}_2$ works on the Moore neighborhood $\mathcal{V}_2$ (9 neighbors including itself). There is no real loss of generality since simulating different neighborhoods in linear time is a well known technique (see [4]).

Now if we assume that all cells in $t(\mathbb{Z}^3)$ have been marked on $\mathcal{A}_2$ by a particular state and also that for each $c \in \mathbb{Z}^3$, $t(c)$ "knows" the state of c in $\mathcal{A}_3$, then $t(c)$ will send signals indicating this state in the 6 directions *up, up-right, right, down, down-left* and *left.*

If we can make t so that the first cells in $t(\mathbb{Z}^3)$ that these signals meet are precisely the images of c's neighbors by t (that we will call the t-neighbors of $t(c)$), $t(c+\vec{x})$ receiving the signal going *right,* $t(c+\vec{y})$ the one going *up,* $t(c+\vec{z})$ the one going *up-right* and symmetrically for their opposites, then each cell in $t(\mathbb{Z}^3)$ gets the state of all its t-neighbors and can apply the rule of $\mathcal{A}_3$, and send its state again...

2 The Projection t from $\mathbb{Z}^3$ to $\mathbb{Z}^2$

In order to be able to transmit the current state of a cell of $t(\mathbb{Z}^3)$ to all its t-neighbors as explained previously, it is necessary that the function t is such that for all $(a,b,c) \in \mathbb{Z}^3$, all the points $t(a+k,b,c)_{k\in\mathbb{Z}}$ are on the same horizontal line and ordered according to k. Moreover, if $(b,c) \neq (b',c')$ then $t(a,b,c)$ and $t(a,b',c')$ must be on different horizontal lines. Similarly, all the points $t(a,b+k,c)_{k\in\mathbb{Z}}$ must be on the same column (one for each couple (a,c)), and $t(a,b,c+k)_{k\in\mathbb{Z}}$ on the same diagonal (one for each (a,b)).

We prove easily that these requirements are equivalent to the existence of f, g, and h three increasing functions from $\mathbb{Z}$ into $\mathbb{Z}$ such that

$$\forall (a,b,c) \in \mathbb{Z}^3, \qquad t(a,b,c) = (f(a)+h(c), g(b)+h(c))$$

$$\left.\begin{array}{l}(x,z) \mapsto f(x)+h(z)\\ (y,z) \mapsto g(y)+h(z)\\ (x,y) \mapsto g(y)-f(x)\end{array}\right\} \text{ are one-to-one.} \tag{1}$$

There are many functions that verify the equations (1), for example

$$f,g,h : \mathbb{Z} \to \mathbb{Z} \quad \begin{cases} f(k) = \mathrm{sgn}(k).2^{3|k|} \\ g(k) = \mathrm{sgn}(k).2^{3|k|+1} \\ h(k) = \mathrm{sgn}(k).2^{3|k|+2} \end{cases} \text{ (where } \mathrm{sgn}(x) \text{ is the sign of } x)$$

However, we will focus on a solution where the functions grow polynomially. We have the following theorem:

Theorem 1. *There exist three increasing functions f, g and h in $\mathbb{Z} \to \mathbb{Z}$, bounded by polynomials of degree 3 and satisfying the equations (1).*

Proof. Consider the three functions f, g and h defined by induction as follows:

- All three functions are odd ($f(0) = 0$ and $\forall x, f(-x) = -f(x)$).
- For all $n \in \mathbb{N}$, $f(n+1)$ is defined as the smallest positive integer not in $\{f(i) + g(j) + g(k), \quad f(i) + h(j) + h(k) \quad | \quad |i|, |j|, |k| \leq n\}$
- For all $n \in \mathbb{N}$, $g(n+1)$ is defined as the smallest positive integer not in $\{g(i) + h(j) + h(k), g(i) + f(j') + f(k') \quad | \quad |i|, |j|, |k| \leq n, |j'|, |k'| \leq n+1\}$
- For all $n \in \mathbb{N}$, $h(n+1)$ is defined as the smallest positive integer not in $\{h(i) + f(j) + f(k), \quad h(i) + g(j) + g(k) \quad | \quad |i|, |j|, |k| \leq n+1\}$

We can easily see that these functions verify the equations (1). Moreover,

$$\forall n \in \mathbb{N}, \quad \begin{cases} \text{Card}(\{f(i) + g(j) + g(k)| - n \leq i, j, k \leq n\}) \leq (2n+1)^3 \\ \text{Card}(\{f(i) + h(j) + h(k)| - n \leq i, j, k \leq n\}) \leq (2n+1)^3 \end{cases}$$

Since both sets contain 0, their union is of cardinal lower than $2(2n+1)^3$. From the definitions of f, g and h, we get

$$\forall n \in \mathbb{N}, \quad \begin{cases} f(n+1) \leq 2(2n+1)^3 & \leq 2(2n+3)^3 = O(n^3) \\ g(n+1) \leq 2(2n+1)(2n+3)^2 & \leq 2(2n+3)^3 = O(n^3) \\ h(n+1) \leq 2(2n+1)(2n+3)^2 & \leq 2(2n+3)^3 = O(n^3) \end{cases} \quad (2)$$

□

From now on, we will consider that the functions f, g and h are the ones defined in the proof of the theorem 1, and the function t is the one defined by (1).

The table 1 gives the first 10 values of the f, g and h functions. We can already see that these functions are very irregular. Although we have a polynomial upper bound (2), the functions don't seem to increase regularly: $g(5)$ and $g(6)$ are extremely close for example. Moreover, from the definitions, it seems that f will be inferior to g and h because its value is chosen before the other two. However, we have $g(7) >> f(7)$ and $h(8) >> f(8)$. For these reasons, it seems very hard to give a good lower bound of these functions.

3 Construction of $t(\mathbb{Z}^3)$ by a 2DCA

In this section, we want to show that $t(\mathbb{Z}^3)$ can be constructed by a cellular automaton in polynomial time, meaning that there exists a polynomial $P \in \mathbb{Z}[X]$ such that for all $p \in \mathbb{Z}^3$, $t(p)$ is constructed at time at most $P(\|p\|)$. We have the following theorem:

Theorem 2. *There exists a 2DCA working on the Moore neighborhood that, starting from a configuration where all cells are in the quiescent state except the origin, will* mark *by a particular state all cells that are in $t(\mathbb{Z}^3)$ in polynomial time.*

Table 1. The first values of the functions f, g and h

n	$f(n)$	$g(n)$	$h(n)$
0	0	0	0
1	1	3	4
2	10	15	17
3	26	38	46
4	57	75	65
5	84	116	128
6	143	119	176
7	152	223	193
8	270	276	340
9	327	380	386
10	510	553	579

3.1 Computations on the Axis

The first step of the computation is to mark all the points on the horizontal axis that correspond to the image of $\mathbb{N}$ by f, g and h. Given the recursive definitions of the functions, the automaton will successively mark $f(1)$, $g(1)$, $h(1)$, $f(2)$ etc.

The general idea is to mark, after $f(k)$ is constructed, all the points of the form $f(k)\pm g(i)\pm g(j)$, $f(k)\pm h(i)\pm h(j)$, $g(i)\pm f(j)\pm f(k)$ and $h(i)\pm f(j)\pm f(k)$ for all $i, j < k$, and symmetrically after marking $g(k)$ or $h(k)$. If we inductively assume that all the necessary markings have been done, then after marking $f(k+1)$ and all the corresponding points, a signal can move to the right from $g(k)$ until it reaches the first cell that is not "forbidden" by the definition of g and mark it as being $g(k+1)$.

Also, from the start of the construction, the horizontal axis ($y = 0$), the vertical axis ($x = 0$) and the "diagonal axis" ($x = y$) will be marked by signals going at maximum speed.

We will only focus on the construction of the functions on the positive half axes, but the symmetric construction will be made on the negative half axes.

Addition and Substraction on the Axes. Let us consider two integers a and b such that $0 < a \leq b$. We assume that both cells $(a, 0)$ and $(b, 0)$ have been marked on the axis at times $\tau(a)$ and $\tau(b)$ respectively. Then, using constructions as illustrated in figures 1 and 2, it is possible to mark the points $(a+b, 0)$ and $(b-a, 0)$ at times

$$\begin{aligned} &\tau(a+b) = \max(\tau(a)+a+b, \tau(b)+2a) \quad \text{and} \\ &\tau(a-b) = \max(\tau(a)+b, \tau(b)+b)\ . \end{aligned} \tag{3}$$

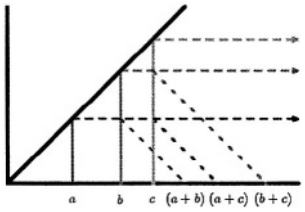


Fig. 1. Additions on a 2DCA

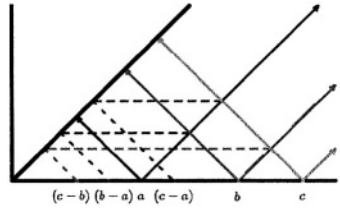


Fig. 2. Substractions on a 2DCA

3.2 Computing the Functions on the *X*-Axis

In the following section, we will identify the x-axis of $\mathbb{Z}^2$ with $\mathbb{Z}$. In other words, we will refer to the cell $(n, 0)$ simply as n.

To construct the sets $f(\mathbb{N})$, $g(\mathbb{N})$ and $h(\mathbb{N})$ we will use many different states. Let's assume that at time $\tau(n)$ all the following cell sets are marked on the x-axis (some cells might be in several sets in which case it is marked with a "combination" of states):

$$F = \{f(k), k \leq n\},$$
$$2F = \{f(k) \pm f(k') | k, k' \leq n\} = F \pm F \text{ and}$$
$$\overline{F} = (\pm F \pm 2G \cup \pm F \pm 2H) \cap \mathbb{N} .$$

and the corresponding sets after permutation of f, g and h.

Also, we assume that $f(n)$, $g(n)$ and $h(n)$ are in particular states that indicate they are the greatest known points of $f(\mathbb{N})$, $g(\mathbb{N})$ and $h(\mathbb{N})$ and that every point marked on the x-axis (as being part of any of the previous sets) has sent signals diagonally to his right and his left in order to be able to perform additions and substractions as illustrated in the previous paragraph.

From here, constructing $f(n+1)$ is simply a matter of moving a signal from $f(n)$ to the right until it reaches a cell that is not marked as being in $\overline{F}$. $f(n)$ "forgets" that it is the last value of F when the signal starts searching for $f(n+1)$ and when $f(n+1)$ is marked, it "knows" that it is the last known value of F. $f(n+1)$ is also marked as being part of the sets F, $2F$, $\overline{F}$, $\overline{G}$ and $\overline{H}$. New diagonal signals are sent to the right and the left, that will interact with the existing ones (sent by previously marked points) to mark the cells of the following sets:

$$f(n+1) \pm F \subseteq 2F,$$
$$2f(n+1) \in 2F,$$
$$\pm f(n+1) \pm 2G \text{ and}$$
$$\pm f(n+1) \pm 2H \subseteq \overline{F} .$$

and then the points in $2F + G \subseteq \overline{G}$ and $2F + H \subseteq \overline{H}$ since we have changed $2F$.

In order for the point $g(n+1)$ to be marked by the same method without error, we have to make sure that the set $\overline{G}$ has been completely marked. It is easy to see that the last point of this set to be constructed will be $2f(n+1) + 2g(n)$. Using the inequations from (3), we see that it'll be constructed at most at time

$$\begin{aligned}&\big[\tau(f(n+1)) + 2f(n+1)\big] + 2\big[f(n+1) + 2g(n)\big]\\ &= \tau(f(n+1)) + 4(f(n+1) + g(n))\ .\end{aligned}$$

where $\tau(f(n+1))$ is the time when $(f(n+1), 0)$ is marked as being in F.

Therefore, we have to wait $4f(n+1) + 4g(n)$ generations before we start looking for $g(n+1)$. To do so, the cell $f(n+1)$ will send two signals as soon as it is marked. The first one will move to the left at speed 1/4 while the second one will move to the right at speed 1. Their evolution will only be affected by the cells $f(n+1)$, $g(n)$ and the origin $(0, 0)$. We are in one of these two possibilities (both cases are illustrated by figures 3 and 4):

- $0 < g(n) < f(n+1)$, in this case the right signal won't encounter any important cell and thus will continue to the right forever. On the other hand, the left signal will reach $g(n)$ and remember it (by changing its state), and continue to the origin still at speed 1/4. After reaching the origin, it will turn around and return to $g(n)$ at speed 1/4.
  When the right signal arrives at $g(n)$ for the second time, it has travelled during $4f(n+1) + 4g(n)$ generations.
- $0 < f(n+1) < g(n)$, in this case, the left signal will go directly to the origin without ever seeing $g(n)$ and will disappear. The right signal however will reach $g(n)$, turn around and go back to $f(n+1)$ at speed 1, then continue to the origin at speed 1/4 where it turns around and goes to $f(n+1)$ for the second time at speed 1/4, and finally goes to $g(n)$ at speed 1/2.
  When it reaches $g(n)$ for the second time, the right signal has travelled during $4f(n+1) + 4g(n)$ generations.

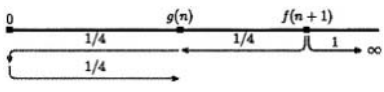


Fig. 3. $0 < g(n) < f(n+1)$

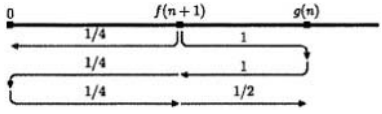


Fig. 4. $0 < f(n+1) < g(n)$

In both cases, we see that one of the signals reaches $g(n)$ exactly when the last point of $\overline{G}$ is constructed on the axis, and so $g(n)$ can send a signal to the right to mark $g(n+1)$. After marking $g(n+1)$, two signals are sent to wait exactly $4g(n+1) + 4h(n)$ generations while the set $\overline{H}$ is being updated. Then $h(n+1)$ is constructed, and after waiting long enough, a signal reaches $f(n+1)$ when $\overline{F}$ is up to date. At this point we are in the same conditions that the ones we assumed to start constructing $f(n+1)$ and so the inductive construction of $f(\mathbb{N})$, $g(\mathbb{N})$ and $h(\mathbb{N})$ can continue.

We have the following inequalities:

$$\begin{aligned}&\tau(g(n+1)) \geq \tau(f(n+1)) + 4f(n+1) + 4g(n)\\ &\tau(h(n+1)) \geq \tau(g(n+1)) + 4g(n+1) + 4h(n))\\ &\tau(f(n+2)) \geq \tau(h(n+1)) + 4h(n+1) + 4f(n+1))\end{aligned} \tag{4}$$

Polynomial Time Bounds. We will now give a rough upper bound of the time needed to construct the functions f, g and h on the x-axis. We assume that $\tau(f(0)) = \tau(g(0)) = \tau(h(0)) = 0$ since the state of the origin is given in the initial configuration. After marking $f(n+1)$, we wait exactly $4f(n+1) + 4g(n)$ generations before the signal starts propagating to the right from $g(n)$ and $g(n+1) - g(n)$ before it reaches its destination.

Therefore we have, using the inequalities from (2), for all $n \in \mathbb{N}$

$$\tau(g(n+1)) = \tau(f(n+1)) + O(n^3)$$

And using the corresponding result on f and h, for all $n \in \mathbb{N}$

$$\tau(g(n+1)) = \tau(g(n)) + O(n^3) = \sum_{k=0}^{n} O(k^3) = O(n^4) \tag{5}$$

We have the same polynomial bound on $\tau(f(n))$ and $\tau(h(n))$.

Construction on the Other Axes. When the cell $(g(n), 0)$ is marked as being part of G, a signal is sent in the direction $(-1, 1)$ and it reaches the y-axis on $(0, g(n))$ after $g(n)$ generations and marks it. Similarly, when $h(n)$ is marked, a signal is sent in the direction $(0, 1)$ that reaches the diagonal axis $(x = y)$ on $(h(n), h(n))$ after $h(n)$ generations.

3.3 Computation of the Rest of $t(\mathbb{Z}^3)$

We will consider here the norm 1 on $\mathbb{Z}^3$ ($\|(a,b,c)\| = |a| + |b| + |c|$). The sphere of radius k for this norm will be denoted as S_k.

Let $p = (x, y, z) \in \mathbb{Z}^3$, with $\|p\| = k$. We will say that $p' = (x', y', z')$ is a *superior neighbor* of p (and that p is an *inferior neighbor* of p') if $p' \in S_{k+1}$ and $\|p' - p\| = 1$. In $\mathbb{Z}^2$ we will say that $t(p')$ is a *superior t-neighbor* of $t(p)$. The number and the position (relative) of the superior neighbors of p only depend on the signs of p's coordinates. Also, every point that is not on one of the axes (in $\mathbb{Z}^3$) has at least two inferior neighbors (one for each non-zero coordinate).

Inductive Construction. We will see here how it is possible to inductively construct the images by t of all the spheres S_k. Since it is not possible to give each cell in $t(\mathbb{Z}^3)$ the number of the sphere it is in (this would require an infinite number of states) we will consider two cell sets S_{+0} and S_{+1} that will represent, for each step k of the induction, the current sphere (S_k) and the next one (S_{k+1}). During step k we will construct S_{k+2}.

For $k_0 \in \mathbb{N}$, let's assume we are in a configuration such that:

- For all $p \in S_{k_0}$, the cell $t(p)$ is in S_{+0} and knows the sign of each of p's coordinates (and so knows the direction in which each of its superior t-neighbors is). Reciprocally, every cell in S_{+0} is the image of a point of S_{k_0}.

– For all $p \in S_{k_0+1}$, the cell $t(p)$ is in S_{+1} and knows the sign of each of p's coordinates (it knows where its inferior t-neighbors are). Also, every cell in S_{+1} is the image of a point of S_{k_0+1}.

Such a situtation is illustrated on the figure 5 for $k_0 = 1$.

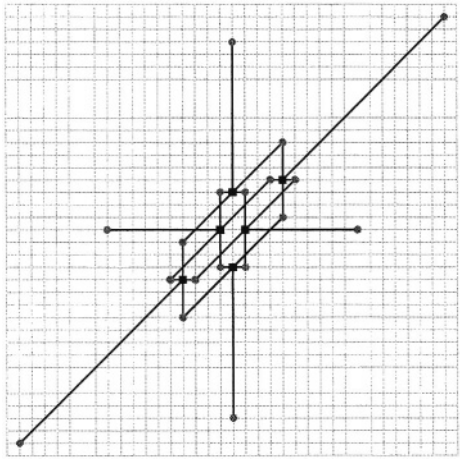


Fig. 5. The spheres S_1 *(black squares)* and S_2 *(grey circles)*

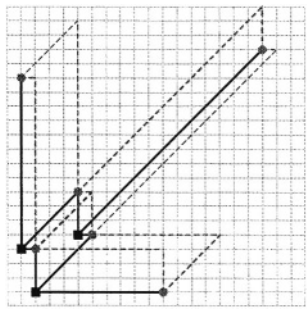


Fig. 6. The signals s_1 *(black)* and s_2 *(dashed grey)* during step 1

In this situation, a signal s_0 is sent from the origin (we will explain later what causes this signal to appear) in all directions and spreads at maximum speed forming a square around the origin. When a cell in S_{+0} receives this signal, it sends a signal s_1 to each of its superior t-neighbors and, after doing so, is no longer in S_{+0}. When a cell in S_{+1} receives a signal s_1, the signal disappears and the cell sends a signal s_2 to each of its superior t-neighbors except the one that is the direction of the received signal (this is illustrated in figure 6 for the upper right quarter of $\mathbb{Z}^2$). The upper t-neighbors of these cells are not yet constructed but the cells know in which direction they are because they know the coordinates of their corresponding point in $\mathbb{Z}^3$. When a cell in S_{+1} has received a signal s_1 from all its inferior t-neighbors it becomes a cell of S_{+0} and is no longer in S_{+1}. Since the signal s_0 spreads at maximum speed we are sure that it cannot interact with these cells (but it will during the next step of course).

When two s_2 signals meet on a cell, they both disappear and the cell is marked as being in S_{+1} (again, this cell won't do anything before the next step since it won't receive any s_1 signal). Also, the direction of the s_2 signals that arrive to a cell indicate the position of its inferior t-neighbors so when all signals have arrived, the cell knows where all its superior t-neighbors are. Note that a given cell can (and most of the time will) receive more than one couple of s_2 signals. Each time, the signals disappear, and the cell gets more information from the inferior t-neighbors.

This construction marks all cells in $t(S_{k_0+2})$ that have at least two inferior t-neighbors. Only the ones on the axes are left, and they are constructed independently as explained earlier.

Correctness of the Construction. The correctness of this construction is based on two observations. First, there cannot be any conflict between the s_1 and s_2 signals because between a cell of $t(\mathbb{Z}^3)$ and one of its superior t-neighbors there is no other cell of $t(\mathbb{Z}^3)$. Second, if we consider a point $p \in S_{k_0+2}$ having at least two inferior neighbors p_1 and p_2, then these two points share a common inferior t-neighbor p' because p_1 and p_2 can be obtained from p by reducing one of the coordinates (a different one for each), and so p' can be obtained by reducing both coordinates from p. Moreover, the basic properties of t ensure that the points $t(p)$, $t(p_1)$, $t(p_2)$ and $t(p')$ form a parallelogram. So the paths $p' \to p_1 \to p$ and $p' \to p_2 \to p$ have same length and so the two s_2 signals will meet on p (see figure 6).

Every cell in $t(S_{k_0+1})$ will therefore receive a couple of coinciding s_2 signals for each couple of its inferior neighbors, and so will be able to deduce the signs of all coordinates of the point in $\mathbb{Z}^3$ it is the image of.

Synchronisation. The previously explained construction relies on a signal s_0 that will initiate it. This signal must appear after all points of S_{k_0} and S_{k_0+1} are marked as being in S_{+0} and S_{+1} respectively. Since the construction is inductive, it is obvious that S_{k_0+1} will be marked after S_{k_0}. Therefore, it is sufficient to ensure that the signal s_0 (corresponding to step k_0) appears after S_{k_0+1} is fully marked.

To do so, let's create a signal from $(h(k_0+1), 0)$ when the cell is marked as being in $h(\mathbb{N})$ (at time $\tau(h(n))$) that will move left at speed 1, and will trigger the s_0 signal when reaching the origin.

Let's prove the construction by induction. For some $k_0 \in \mathbb{N}$, let's assume that all points of S_{k_0+1} are marked at time $\tau_s(k_0) = \tau(h(k_0+1)) + h(k_0+1)$ (when the signal s_0 from step k_0 appears). Then s_0 reaches all points in S_{k_0} by time at most

$$\tau_s(k_0) + \max(f(k_0), g(k_0)) + h(k_0)$$

and the propagation of signals s_1 and s_2 lasts at most

$$2\max(f(k_0+1) - f(k_0), g(k_0+1) - g(k_0), h(k_0+1) - h(k_0))$$

In the end, all points of S_{k_0+1} that are marked by these signals are marked at time at most

$$\tau(h(k_0+1)) + h(k_0+1) + \max\begin{cases} f(k_0) + h(k_0) \\ g(k_0) + h(k_0) \end{cases} + 2\max\begin{cases} f(k_0+1) - f(k_0) \\ g(k_0+1) - g(k_0) \\ h(k_0+1) - h(k_0) \end{cases}$$

It is easy to see that this time is lower than

$$\tau(h(k_0+1)) + h(k_0+1) + 2f(k_0+1) + 2g(k_0+1) + 2h(k_0+1)$$

And if we use the inequalities from (4) we get

$$\begin{aligned}\tau_s(k_0+1) &= t(h(k_0+2)) + h(k_0+2)\\ &\geq \tau(h(k_0+1)) + 4h(k_0+1) + 4f(k_0+2) + 4g(k_0+2) + h(k_0+2)\\ &\geq \tau(h(k_0+1)) + h(k_0+1) + 2f(k_0+1) + 2g(k_0+1) + 2h(k_0+1)\end{aligned}$$

Also, the only points in $t(S_{k_0+2})$ that are not constructed by the signals s_2 are the ones on the axes. There are six of them: $\pm(f(k_0+2),0)$, $\pm(0,g(k_0+2))$ and $\pm(h(k_0+2),h(k_0+2))$, and they are marked at times $\tau(f(k_0+2))$, $\tau(g(k_0+2))+g(k_0+2)$ and $\tau(h(k_0+2))+h(k_0+2)$ respectively, all lower than or equal to $\tau_s(k_0+1)$.

Therefore, when the signal s_0 from step k_0+1 is triggered all points of S_{k_0+1} and S_{k_0+2} are marked and the construction can be done correctly.

3.4 End of the Construction

The last thing to do now is to "activate" the cells in $t(\mathbb{Z}^3)$ when all of their t-neighbors have been constructed (we will need this information later when simulating $\mathcal{A}_3$). To do so, since every point of $t(\mathbb{Z}^3)$ knows the direction where all its inferior t-neighbors are when it is constructed, it will send a special "activation" signal to them as soon as it is marked. When a given cell has received an "activation" signal from all its superior t-neighbors it goes into a new "activated" state.

We have therefore seen in this section how it is possible to construct, using a 2DCA working on Moore's neighborhood starting on the configuration where all cells are in a quiescent state except the origin, the image of $\mathbb{Z}^3$ by t. This construction is done in polynomial time, meaning that the image of the sphere S_k, which is of radius $O(k^3)$ as seen in (2) is constructed in time $O(k^4)$ (according to (5) and the arguments of induction).

This ends the proof of theorem 2.

4 Description of $\mathcal{A}_2$

The final 2DCA $\mathcal{A}_2$ is a superposition of two 2DCA working in parallel. The *lower layer* is the one that will mark $t(\mathbb{Z}^3)$ during the simulation as explained previously. We will represent the evolution of this construction by colors. A cell that has been marked as part of $t(\mathbb{Z}^3)$ but has not yet been *activated* (its t-neighbors are still being constructed) will be represented as *grey*. A cell marked and activated will be either *black* or *red* (we will see later why we need two colors) and a cell that hasn't been marked is *white*. We will say that a cell is *colored* if it is either *grey, black* or *red* (as opposed to *white*). The *upper layer* is the one that will really do the simulation.

Colored cells all correspond to a cell of $\mathcal{A}_3$. Their state will be a product of information. What we will call their central state is the state of their corresponding cell that they are currently simulating. A *red* cell will send signals

containing its central state in the directions of its t-neighbors. The next generation, it becomes *black* so that the central state is sent only once. These signals are transmitted by *white* cells. When a *black* cell receives one of these signals, the signal disappears and the cell memorizes the state of the corresponding t-neighbor (which neighbor depends on the direction the signal came from). When it knows all the central states of its t-neighbors, the *black* cell applies the transition function of $\mathcal{A}_3$, changes its central state and becomes *red* (to transmit the new central state)...

Given a configuration $\mathcal{C}_3 \in Q_3^{\mathbb{Z}^3}$, the initial configuration of $\mathcal{A}_2$ is as follows. The lower layer is initialized on the origin only (all other cells are in the blank state) to construct $t(\mathbb{Z}^3)$ as we have seen. If $\mathcal{C}_3$ is finite (only a finite number of cells are not in the blank state q_0), for all $c \in \mathbb{Z}^3$ such that $\mathcal{C}_3(c) \neq q_0$, $t(c)$ receives $\mathcal{C}_3(c)$ as its central state. All other cells are left blank, so the initial configuration of $\mathcal{A}_2$ is finite too. If $\mathcal{C}_3$ is infinite, all cells in $t(\mathbb{Z}^3)$ receive the state of their preimage.

As the construction of $t(\mathbb{Z}^3)$ by the lower layer goes on, cells become *grey* (marked but not activated) and then they are activated and become *red.* A new *red* cell either has a central state given initially, or it is initialized as q_0 when the cell is activated. Note that *grey* cells gather informations about their t-neighbors like *black* cells do but don't send their central state because some of their t-neighbors aren't marked yet and the signals might be lost.

It is important to see that the simulation of $\mathcal{A}_3$ by $\mathcal{A}_2$ is therefore asynchronous because the colored cells apply the transition function of $\mathcal{A}_3$ at different times. This means that the number of $\mathcal{A}_3$ generations computed by the colored cells isn't the same for all. However, there is a dependency between a cell and its t-neighbors that ensures that, at a given time τ, each colored cell $t(c)$ has computed at most one generation more than any of its t-neighbors. This gives a sort of *partial-synchronicity* and proves that a *black* or *grey* cell can never receive more than two consecutive signals from any of its t-neighbors before it can process them, and so each cell can have only a finite memory.

This automaton simulates $\mathcal{A}_3$ in the sense that for a given configuration $\mathcal{C}_3$, for any cell $c \in \mathbb{Z}^3$, the sequence of states of c in the evolution of $\mathcal{A}_3$ starting on $\mathcal{C}_3$ is exactly the sequence of central states of $t(c)$ restricted to the times when $t(c)$ is *red.*

The time needed for a given cell $t(c)$ to compute k generations of its corresponding cell c is at most the time needed to activate all the image by t of the sphere of radius k centered on c, plus the radius of the image of this sphere (it's the time needed for the furthest information to reach $t(c)$). We have seen that these spheres grow polynomially, and that they are constructed in polynomial time so for every cell $c \in \mathbb{Z}^3$ there exists a polynomial $P_c \in \mathbb{Z}[X]$ of degree at most 4 such that $t(c)$ has computed k generations of c in time at most $P_c(k)$. Note that the polynomial depends on c but considering the relative sizes of the spheres of radius k in $\mathbb{Z}^3$ and $\mathbb{Z}^2$ it is easy to prove that we cannot simulate k generations of $\mathcal{A}_3$ on $\mathbb{Z}^2$ in a time bounded by a global polynomial (it is in fact impossible to bound it by any function).

5 Conclusion

What we have done here can easily be done to simulate any n-dimensional CA on a 2DCA. However doing such a construction to simulate a 2DCA with a 1DCA doesn't work (mainly because we cannot embed as we did the 2D neighborhood in a 1D neighborhood and use a similar construction without important conflicts).

Many things can now be studied about this simulation. For example, the f, g and h functions used here are bounded by polynomials of degree 3. Considering the size of the spheres, we see that it is not possible to have polynomials of degree lower than 3/2 but where exactly is the real lower bound of the simulation? Also the irregularity of the functions that we use here make them hard to compute with a 2DCA, but having a better understanding of their growth (a good lower bound) or finding simpler functions (still bounded polynomially) could highly improve the construction in terms of time and number of necessary states.

The automaton constructs a representation of $\mathbb{Z}^3$ in $\mathbb{Z}^2$. It would be interesting to study how usual 3-dimensional objects are represented, and if natural 3-dimensional cellular automata configurations can be identified easily after projection.

We have seen here how to encode finite configurations of $\mathcal{A}_3$ as finite configurations of $\mathcal{A}_2$. Periodic $\mathcal{A}_3$ configurations can also be simulated by a 2DCA using a method similar to the one described if we encode them as finite configurations of $\mathcal{A}_2$ using specific state markers for the borderline and sending the signals from one border to the other (as on a torus).

References

1. Frisch, U., d'Humières, D., Hasslacher, B., Lallemand, P., Pomeau, Y., Rivet, J.P.: Lattice gas hydrodynamics in two and three dimension. Complex Systems **1** (1987) 649–707 Reprinted in *Lattice Gas Methods for Partial Differential Equations,* ed. G. Doolen, p.77, Addison-Wesley, 1990.
2. Róka, Z.: Simulations between cellular automata on Cayley graphs. Theoretical Computer Science **225** (1999) 81–111
3. Martin, B.: A geometrical hierarchy on graphs via cellular automata. Fundamenta Informaticae **52** (2002) 157–181
4. Cole, S.N.: Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Transactions on Computers **C-18** (1969) 349–365

Graph Exploration by a Finite Automaton

Pierre Fraigniaud[1], David Ilcinkas[1], Guy Peer[2], and Andrzej Pelc[3], and David Peleg[2]

[1] CNRS, LRI, Université Paris-Sud, France
[2] Dept. of Computer Science, Weizmann Institute, Israel
[3] Dép. d'informatique, Univ. du Québec en Outaouais, Canada

Abstract. A finite automaton, simply referred to as a *robot,* has to explore a graph whose nodes are unlabeled and whose edge ports are locally labeled at each node. The robot has no a priori knowledge of the topology of the graph or of its size. Its task is to traverse all the edges of the graph. We first show that, for any K-state robot and any $d \geq 3$, there exists a planar graph of maximum degree d with at most $K + 1$ nodes that the robot cannot explore. This bound improves all previous bounds in the literature. More interestingly, we show that, in order to explore all graphs of diameter D and maximum degree d, a robot needs $\Omega(D \log d)$ memory bits, even if we restrict the exploration to planar graphs. This latter bound is tight. Indeed, a simple DFS at depth $D + 1$ enables a robot to explore any graph of diameter D and maximum degree d using a memory of size $O(D \log d)$ bits. We thus prove that the worst case space complexity of graph exploration is $\Theta(D \log d)$ bits.

1 Introduction

1.1 Background and Motivation

A mobile entity, e.g., a software agent or a robot, has to *explore* an undirected graph by visiting all its nodes and traversing all its edges, without any a priori knowledge of the topology of the graph or of its size. The task of visiting all nodes is fundamental in searching for data stored at unknown nodes of a network, and traversing all edges is often required in network maintenance and when looking for defective components. More precisely, we consider the task of "perpetual" exploration in which the robot has to traverse all edges of the graph but is not required to stop. That is, the robot moves from node to node, traversing edges, so that eventually all edges have been traversed. Perpetual exploration is of practical interest, e.g., if regular control of a network for the presence of faults is required, and all edges must be periodically traversed over long periods of time.

If nodes and edges have unique labels, exploration can be easily achieved (e.g., by depth-first search). However, in some navigation problems in unknown environments, such unique labeling may not be available, or limited sensory capabilities of the robot may prevent it from perceiving such labels. Hence it is important to be able to program the robot to explore *anonymous* graphs, i.e., graphs without unique labeling of nodes or edges. Clearly, the robot has

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 451–462, 2004.
© Springer-Verlag Berlin Heidelberg 2004

to be able to locally distinguish ports at a node: otherwise it is impossible to explore even the star with 3 leaves (after visiting the second leaf, the robot cannot distinguish the port leading to the first visited leaf from that leading to the unvisited one). Hence we make a natural assumption that all ports at a node are locally labeled $1, \ldots, d$, where d is the degree of the node. No consistency between those local labelings is assumed.

In many applications, robots and mobile agents are meant to be simple, often small and inexpensive devices. This limits the amount of memory with which they can be equipped. As opposed to numerous papers that imposed no restrictions on the memory of the robot and sought exploration algorithms minimizing time, i.e., the number of edge traversals, we investigate the minimum memory size of the robot that allows exploration of graphs of given (unknown) size, regardless of the time of exploration. That is, we want to find an algorithm for a robot performing exploration, using as little memory as possible.

A robot with a k-bit memory is modeled as a finite automaton. The first known finite automaton algorithm designed for graph exploration was introduced by Shannon [35] in 1951. Since then several papers have been dedicated to the graph exploration problem. In 1967, during his talk at Berkeley, Rabin [32] presented a proof that no finite automaton with a finite number of pebbles can explore all graphs (a pebble is a marker that can be dropped at and removed from nodes). In 1971, Müller [29] gave some formal arguments to support Rabin's claim, in the restricted case of a robot without pebble. In 1977, Coy [16] presented another proof, but some parts of it are fuzzy. The first formal proof of Rabin's claim is generally attributed to Budach [15], in 1978, for a robot without pebble. Actually, the long and technical paper by Budach is concerned with labyrinths. A *labyrinth* is a two-dimensional obstructed chess-board (i.e., $\mathbb{Z}^2$ with forbidden cells). The forbidden cells in $\mathbb{Z}^2$ are described by a set L. If L (resp., $\mathbb{Z}^2 \setminus L$) is finite, then the labyrinth is called finite (resp., co-finite). Exploring a finite labyrinth means that the automaton is able to go arbitrarily far away from its starting position, for any starting position. The edges of the labyrinth are consistently labeled North, South, East, West. (Budach's result applies also to graphs because a co-finite labyrinth is a finite graph.) The same year, Blum and Kozen [10] improved Budach's result by proving that three finite automata cannot cooperatively perform exploration of all graphs. In 1979, Kozen [27] proved that four cooperative robots cannot explore all graphs. Finally, in 1980, Rollik [34] gave a complete proof of Rabin's claim. More precisely, Rollik proved that no finite set of finite automata can cooperatively perform exploration of all cubic planar graphs. Since a finite automaton is more powerful than a pebble, Rabin's claim is a corollary of Rollik's theorem. In all proofs, including the one by Budach and the one by Rollik, the size of the smallest *trap* for an automaton with no pebble (i.e., the smallest graph that an automaton with no pebble cannot explore) is large. One of the objectives of the current paper is to revisit Rabin's claim in the case of a robot with no pebble, specifically for improving the size of traps, and for designing traps with specific topological properties.

1.2 Our Results

Our first result is the design of a trap with at most $K+1$ vertices for any K-state automaton. More precisely, we prove that, for any $d \geq 3$ and for any K-state automaton, there exists a planar graph of $K + 1$ nodes and maximum degree d that the automaton cannot explore. (We assume $d \geq 3$ since, obviously, all connected graphs of maximum degree $d \leq 2$ can be explored by a robot with a constant memory size.) This construction improves –in terms of size– the best bound known so far, i.e., $2K$, due to Rollik.

More importantly, our construction methodology is quite generic and can be adapted for the minimization of other graph parameters. In particular, we prove that, for any $d \geq 3$ and for any K-state automaton, there exists a planar graph of $O(K)$ nodes, maximum degree d, and diameter $O(\frac{\log K}{\log d})$ that the automaton cannot explore. This latter result has an important corollary, namely that for any $d \geq 3$ and any D, a robot requires $\Omega(D \log d)$ memory bits to explore all graphs of maximum degree d and diameter D. This bound is tight. Indeed, a simple DFS at depth $D + 1$ enables a robot with $O(D \log d)$ memory bits to explore all graphs of maximum degree d and diameter D.

To summarize, we prove that the worst case space complexity of graph exploration is $\Theta(D \log d)$ bits.

1.3 Related Work

Exploration and navigation problems for robots in an unknown environment have been extensively studied in the literature (cf. [25,33]). There are two groups of models for these problems. In one of them a particular geometric setting is assumed (see, e.g., [5,9,17]). Another approach is to model the environment as a graph, assuming that the robot may only move along its edges. The graph setting can be further specified in two different ways. In [1,8,7,18,24] the robot explores strongly connected directed graphs and it can move only in the head-to-tail direction of an edge, not vice-versa. In [4,12,15,19,21,22,23,28,31,34] the explored graph is undirected and the robot can traverse edges in both directions. Graph exploration scenarios considered in the literature differ in an important way: it is either assumed that nodes of the graph have unique labels which the robot can recognize (as in, e.g., [18,22,31]), or it is assumed that nodes are anonymous (as in, e.g., [7,8,15,34]). We are concerned with the latter context. The efficiency measure adopted in papers dealing with graph exploration is either the completion time of this task, measured by the number of edge traversals, (cf., e.g., [31]), or the memory size of the robot, measured either in bits or by the number of states of the finite automaton modeling the robot (cf., e.g.[19, 24]). Time is not an issue in our approach, and we address the latter efficiency measure, i.e., memory space. Three versions of the exploration problem have been addressed in the literature: exploration with return (in which the robot has to perform exploration and return to its starting position), exploration with stop (in which the robot has to complete exploration and eventually stop), and perpetual exploration (the type of exploration considered in this paper). For

instance, it is shown in [19] that exploration with stop in n-node trees requires a robot with memory size $\Omega(\log\log\log n)$, and that exploration with return in n-node trees can be achieved by a robot with $O(\log^2 n)$ memory bits. Minimizing the memory of the robot for the exploration of anonymous undirected graphs has been addressed in, e.g., [10,15,19,27,34].

Also, a large part of the literature is concerned with labyrinth exploration. The exploration problem in such labyrinths is known to be strictly simpler than graph exploration [10]. In [20], Döpp proved that a robot is able to explore all finite one-component labyrinths (i.e., where the set L of forbidden cells is finite and connected), and asked whether there exists a universal finite automaton, i.e., one able to explore every finite labyrinth. Budach proved (see the sketch in [13] and the complete version in [15]) that no finite automaton can explore all finite labyrinths. The same result holds for co-finite labyrinths. In [11], Blum and Sakoda have shown that there exists a finite automaton able to explore all finite labyrinths using 4 pebbles (the automaton is universal for all labyrinths if 7 pebbles are allowed). Blum and Kozen [10] proved that a finite automaton with only 2 pebbles can explore all co-finite labyrinths. The problem was finally closed by Hoffmann [26] who showed that a finite automaton with a unique pebble cannot explore all finite labyrinths. Again, the same result holds for co-finite labyrinths. Furthermore, a trap for a finite automaton (using no pebble) can be constructed such that L has only three connected components (cf. [30]). Finally, for any finite set of non-cooperative automata, there exists a finite labyrinth that these automata cannot explore [2],

It is worth mentioning that our work has connections with derandomized random walks. There, the objective is to produce an explicit universal traversal sequence (UTS), i.e., a sequence of port labels, such that the path guided by this sequence visits all edges of any graph. However, even if bounds on the length of these sequences have been derived, they provide little knowledge on the minimum number of states for graph exploration by a robot. For instance, sequences of length $\Omega(n\log n)$ are required to traverse all degree 2 graphs with n nodes [6], although a 2-state robot can explore all degree 2 graphs.

2 Terminology and Model

An anonymous undirected graph with locally labeled ports is a graph whose nodes are unlabeled and where the edges incident to a node v have distinct labels $1,\ldots,d_v$, where d_v is the degree of v. Thus every undirected edge $\{u,v\}$ has two labels which are called its *port numbers* at u and at v. Port numbering is *local,* i.e., there is no relation between port numbers at u and at v. Unless specified otherwise, all considered graphs are supposed to be connected.

We are given a mobile entity traveling in an anonymous graph with locally labeled ports. The graph and its size are a priori unknown to the entity. The mobile entity is referred to as a *robot.* More precisely, a K-state robot is a finite Moore automaton $\mathcal{R}=(X,Y,\mathcal{S},\delta,\lambda,S_0)$ where $X\subseteq\mathbb{N}^2$, $Y\subseteq\mathbb{N}$, $\mathcal{S}$ is a set of K states among which is a specified state S_0 called the *initial* state, $\delta:\mathcal{S}\times X\to\mathcal{S}$,

and $\lambda : \mathcal{S} \to Y$. Initially the robot is at some node u_0 in the initial state $S_0 \in \mathcal{S}$. S_0 determines a local port number $p = \lambda(S_0) \in Y$, by which the robot leaves u_0. When incoming to a node v, the behavior of the robot is as follows. It reads the number i of the port through which it entered v and the degree d_v of v. The pair $(i, d_v) \in X$ is an input symbol that causes the transition from state S to state $S' = \delta(S, (i, d_v))$. S' determines a local port number $p = \lambda(S')$, by which the robot leaves v. The robot continues moving in this way, possibly infinitely.

As mentioned before, we consider the task of "perpetual" exploration in which the robot has to traverse all edges of the graph but is not required to stop. That is, it is not required that a final state be in $\mathcal{S}$. A robot is said to perform an *exploration* of a graph G, if starting at *any* node of G in the initial state S_0, it completes traversing all edges of G in finitely many steps.

3 Traps and Lower Bounds

In order to prove lower bounds for the exploration problem, we first study the maximum size of graphs that a given robot can explore. Let $\mathcal{R}$ be a robot. A *trap* for $\mathcal{R}$ is a pair (G, u_0) where $G = (V, E)$ is a graph, $u_0 \in V$, and starting at vertex u_0 the robot $\mathcal{R}$ fails to explore G, i.e., there exists an edge $e \in E$ such that, for any $t \geq 0$, the robot has not traversed e during the first t steps of the exploration. Given a K-state robot $\mathcal{R}$ (hence with $\lceil \log K \rceil$ memory bits), we construct a trap for this robot. Our objective is to construct small traps, or traps with small diameter.

For the purpose of constructing traps, let us introduce some tools. A graph G of maximum degree d is *edge-colored* if every edge of G is given a color, every two incident edges have different colors, and there are d colors used in total. There is a clear correspondence between regular edge-colored graphs and regular edge-labeled graphs in which the labels at the two extremities of each edge are identical.

Definition 1. *A sequence L of labels is a* pseudo-palindrome *if any of the following two conditions is satisfied:* (1) $L = \emptyset$, *or* (2) $L = L' \circ (\ell, \ell) \circ L''$, *where $L' \circ L''$ is a pseudo-palindrome, ℓ is a label, and $\circ$ denotes concatenation. In particular, a palindrome is a pseudo-palindrome precisely if its length is even.*

A sequence L' is a *reduction* of L if $L' = A \circ B$ and $L = A \circ L'' \circ B$ where L'' is a nonempty pseudo-palindrome, and A and B are two arbitrary sequences (possibly empty). A sequence is said *pp-free* if it has no reduction. A sequence L' is the *pp-reduction* of a sequence L if L' is pp-free and obtained from L by successive reductions. One can easily check that the pp-reduction of a sequence is unique (cf., e.g., Section 1.7 in [15]). For instance the pp-reduction of 11221211213223311313323112 21 is 1231. Obviously, given any d-regular edge-colored graph $G = (V, E)$ and any node $u \in V$, each sequence L of edge labels defines a path P from u in G. If L is a pseudo-palindrome then P starts and ends at u. (If G is a tree, L is a pseudo-palindrome if and only if P starts and ends at u.)

Theorem 1. *For every K-state robot and every $d \geq 3$, there exists a planar graph of maximum degree d with at most $K + 1$ nodes that the robot cannot explore.*

Proof. Let T_d be the infinite edge-colored regular tree of degree d. Let u_0 be any node. Assume that the robot starts from u_0 in state S_0. After at most K steps (hence after visiting at most $K+1$ nodes), the robot has been twice in the same state. Let S be such a state, and let u and u' be the first occurrences of two nodes where the robot is in state S. The robot $\mathcal{R}$ is at u at step t and at u' at step t'. Since the robot is in an edge-colored regular graph, the sequence of states becomes periodic after t. Let $p = t' - t$ be the period, and let u'' be the node reached by $\mathcal{R}$ at step $t + 2p = t' + p$. At this step, the robot is again in state S. Finally, let L be the ordered sequence of labels of the edges traversed by the robot from step $t + 1$ to step t', and let L' be its pp-reduction.

Intuitively, to construct a trap based on T_d, we modify T_d by merging two nodes so that the robot is trapped in a periodic movement in the modified graph. More precisely, we proceed according to the pp-reduction L' of the sequence of labels L visited while going from u to u' in T_d. First we define an intermediate graph G'', whose definition differs according to the structure of L'.

Case 1: L' is not a palindrome (in particular, L' is not empty). Then L' describes the simple path from u to u' in T_d. Since L' is not a palindrome, it can be written as $L' = (l_1, \ldots, l_q, l, \ldots, l', l_q, \ldots, l_1)$ with $l \neq l'$. Moreover, since L' is pp-free, we have also $l' \neq l_q$. Let v (resp., v') be the node reached from u (resp., u') after following the sequence of edge labels $l_1, \ldots, l_q$ (see Figure 1). Note that we may have $v = u$ and $v' = u'$, but not $v = v'$. Let w be the neighbor of v' such that $\{w, v'\}$ is labeled l'. Since L' is not a palindrome, we have $w \neq v$. We construct G'' as follows. We delete edges labeled l' incident to v and to w, and we replace these two edges by an edge between v and w. This edge is labeled l'. Note that G'' has exactly three connected components.

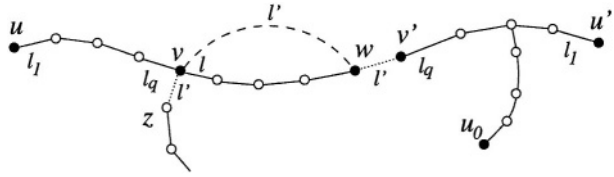


Fig. 1. Construction of G'' in the case where L' is not a palindrome. The dotted edges (v, z) and (w, v') are removed and the dashed edge (v, w) is added instead.

We prove that the behavior of the robot becomes periodic in G''. For that purpose, let us first recompute a starting node of the robot $\mathcal{R}$ such that $\mathcal{R}$ is at u in state S at step t in G''. (Note that the original starting node u_0 may be in a connected component different from the one of u and v. However, u and v are in the same component because $l_q \neq l'$.) To do that, let L'' be the sequence of

edge labels corresponding to the walk of the robot from u_0 to u in T_d. Starting from u, let v_0 be the node reached when the robot traverses the edges labeled by L'' in the reverse order. The robot starts in v_0 and, by construction, reaches u in state S at step t. Let us consider the next p steps of the exploration. Since the connected component of G'' containing u is a regular graph of degree d, the sequence of robot's states is the same in G'' as in T_d. Thus, at step t' the robot is in state S in G''. Any pseudo-palindrome defines a closed walk in G''. Recall that the pp-reduction L' of L can be written $L' = (l_1, \ldots, l_q, l, \ldots, l'', l', l_q, \ldots, l_1)$. The sequence $(l_1, \ldots, l_q)$ leads from u to v, and the sequence $(l, \ldots, l'')$ leads from v to w. Indeed, the modification of T_d does not modify the path from u to w because $l' \neq l_q$ and $l' \neq l$. From w, the robot takes the edge labeled l', which is the edge that was added between v and w during the construction of G''. Hence, the robot is back at v in G''. Finally $(l_q, \ldots, l_1)$ leads back from v to u, and $\mathcal{R}$ is in state S at u at step t'. The robot's behavior is thus periodic in G'', as claimed.

Let G' be the graph consisting of all edges traversed by the robot in G'' when starting from v_0. More precisely, G' is the graph composed of all nodes and edges that the robot traverses at least once during its journey from v_0 in G''. Since the robot's behavior is periodic in G'', G' is a finite graph. Actually, G' has at most K nodes. Indeed, after t' steps, the robot is trapped in a cycle. Thus, it does not visit new nodes after step t'. During the first t' steps, the robot visits at most $t'+1$ nodes. However, it is at the same node u at step t and t'. Hence, the robot visits at most $t' \leq K$ nodes.

To complete the construction of the trap, we add edges to make the degrees of every node in G' exactly d, so that the sequence of robot's states is the same in G' as in T_d. Since G'' is infinite and d-regular, and G' is a finite subgraph of G'', there are necessarily some nodes in G' with degree less than d. Thus, we now complete G' by pairing nodes (possibly including self-loops) until every node of G' (i.e., visited by the robot) is of degree exactly d.

More precisely, let x be a node that needs r additional incident edges. If r is even, we create $r/2$ self-loops around x. If r is odd, we create $(r-1)/2$ self-loops around x. Then every node needs at most one additional edge. G' is a tree, so one can match these nodes, adding one edge for each pair, so that the resulting graph remains planar. After that, there remains at most one unmatched node. We connect this node to an additional (new) node y of degree 1. (As y is never visited by the robot, its degree is immaterial). Therefore, we obtain a planar graph G with at most $K+1$ nodes (recall that G' has at most K nodes). The added edges are labeled locally. This labeling can be chosen arbitrarily because these edges are not traversed by the robot anyway (the robot only traverses edges of G'). Since exploration means traversing all edges, the robot fails to explore G, and thus (G, v_0) is a trap for $\mathcal{R}$.

Case 2: L' is a palindrome. There are two subcases.

Subcase 2.1: L' is empty (i.e., L is a pseudo-palindrome). Then $u = u'$. The behavior of the robot $\mathcal{R}$ becomes periodic in T_d because $u = u'$ and the robot is in the same state at u in steps t and t'. Hence, G' is defined as in the previous

case, i.e., as the graph consisting of all edges traversed by the robot in T_d when starting from u_0. G' is then transformed into G as before. Since, for the same reasons as for the general case, G' has at most K nodes, we get that G has at most $K + 1$ nodes. Not all edges of G are visited. Thus (G, u_0) is a trap for $\mathcal{R}$.

Subcase 2.2: L' is a nonempty (odd length) palindrome. Then L' concatenated with itself is a pseudo-palindrome, and thus $u = u''$. As in the previous cases, the behavior of the robot becomes periodic in T_d. However, unlike what happened in the previous cases, the period is $2p$, and the end of the first period occurs at step $t' + p$, instead of step t'. Hence a graph G' defined as in the previous cases may have more than K nodes. To keep G' small, we slightly change the definition of G'' compared to the previous cases. During the $2p$ steps following step t, the robot starts and ends at u in T_d. Since T_d is a tree, the robot visits every edge at least twice, and thus it visits at most p edges and $p + 1$ nodes. Thus in total, the robot visits at most $t + p + 1 = t' + 1$ nodes. Since $t' \leq K$, at most $K + 1$ nodes are visited. If the robot actually visits at most K nodes of T_d, then we set $G'' = T_d$. If $\mathcal{R}$ visits exactly $K + 1$ nodes of T_d, then we modify T_d as follows. Let $\hat{L}$ be the sequence of edge labels seen by the robot during the first $t + 2p$ steps of its journey. Note that $\hat{L}$ contains at least two different labels (i.e., it is not a sequence $(l, l, \ldots, l)$). Indeed, $K \geq d \geq 3$ and thus a sequence $\hat{L} = (l, l, \ldots, l)$ would imply that some edge is visited at least three times. Therefore the robot would have visited at most K nodes, a contradiction with our assumption that $\mathcal{R}$ visits exactly $K + 1$ nodes. Let $\hat{L} = (\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_{t+2p})$. Choose the first $i \geq 1$ such that $\hat{l}_i \neq \hat{l}_{i+1}$. For every j, let $\hat{u}_j$ be the node reached by the robot at the end of step j. We merge $\hat{u}_{i-1}$ and $\hat{u}_{i+1}$ by constructing two parallel edges between $\hat{u}_{i-1}$ and $\hat{u}_i$, one of which is labeled $\hat{l}_i$, while the other is labeled $\hat{l}_{i+1}$. The resulting graph is denoted G''. Clearly, the robot visits at most K nodes in G''. We now define G' and G as in the previous cases. G' has at most K nodes, and thus G has at most $K+1$ nodes. In spite of the double edge, the behavior of the robot is periodic in G'' because the sequence LL is a pseudo-palindrome and thus it defines a closed walk in any edge-colored graph. For the same reasons as in the previous cases, not all edges of G are traversed by $\mathcal{R}$, and thus (G, u_0) is a trap for $\mathcal{R}$, which completes the proof of Theorem 1. □

We can rephrase Theorem 1 as follows:

Corollary 1. *A robot that explores all n-node planar graphs requires at least $\lceil \log n \rceil$ memory bits.*

The next result links the number of states of a robot with the maximum diameter of the graphs that it can explore.

Theorem 2. *For every K-state robot and every $d \geq 3$, there exists a planar graph of maximum degree d and diameter at most $4\lceil \log_{d-1} K \rceil + 2$ that the robot cannot explore.*

Proof. We start from the intermediate graph G' defined in the proof of Theorem 1. We complete G' so that all nodes of G' are of degree d as follows. G'

has at most K nodes and for each node there are at most $d-1$ missing edges. Hence, we consider the d-regular tree B of depth $h = \lceil \log_{d-1} K \rceil$. B has at least $d(d-1)^{h-1} > K$ leaves. We add edges from every node of G' to different leaves of B, so that all visited nodes (the nodes of G') are of degree exactly d. The added edges are labeled locally arbitrarily. The resulting graph is denoted by G. Clearly, the pairing between the nodes of G' and the nodes of B can be done so that G is planar. To compute an upper bound on the diameter of G, let us consider an arbitrary node x of G'. During the construction of G', we used T_d, and we constructed G'' in which at most two nodes were modified. Therefore, since $d \geq 3$, at least one edge leads in G'' from x to a node r which is the root of an unmodified infinite subtree T of T_d. At distance at most h from node r, there are at least $(d-1)^h$ nodes in T. Since $(d-1)^h \geq K$, we get that there is a node of T, at distance at most h from r, that is in G'' but not in G'. Therefore, there exists a node in G', at distance at most $h-1$ from r, that has degree smaller than d in G'. This node is connected to the tree B in G. Thus, any node of G' is at distance at most $h+1$ from a node of B. The diameter of B is $2h$. Thus, the diameter of G is at most $(h+1)+2h+(h+1) = 4h+2$, which completes the proof. (As in Theorem 1, the graph G is a trap for $\mathcal{R}$ because the vertices of B are not visited by the robot.) □

As a direct consequence of Theorem 2, we have:

Corollary 2. *A robot that explores all graphs of diameter D and maximum degree d requires at least $\Omega(D \log d)$ memory bits.*

By Corollary 2, the best that a k-bit memory robot can do is to explore all graphs of diameter D and maximum degree d such that $k = \Omega(D \log d)$. In the next section, we show that this goal can be achieved.

4 An Exploration Algorithm

In this section, we present an algorithm called `Increasing-DFS`, that enables a robot to explore all graphs of sufficiently small diameter and maximum degree. Roughly speaking, exploration is achieved by using a sequence of *depth-first search (DFS)* operations at increasing depths from the initial position u_0 of the robot. The robot keeps in memory the current sequence of port numbers leading back to u_0 in the DFS tree. At Phase i, $i \geq 1$, the robot performs a DFS of depth bounded by i. In the case where one is given a robot $\mathcal{R}$ with k memory bits, we use the variant **k-Increasing-DFS**, that is `Increasing-DFS` in which the robot perpetually checks the size of the currently allocated memory. If this size exceeds k bits, then the robot stops.

Theorem 3. *Algorithm* `Increasing-DFS` *allows a robot to explore every graph. Moreover, Algorithm* **k-Increasing-DFS** *explores all graphs of diameter D and maximum degree d, whenever $k \geq \alpha D \log d$, for some positive constant α.*

Proof. Let the robot $\mathcal{R}$ start from node u_0 in graph G. After $\mathcal{R}$ has performed a DFS of depth i, it has visited all nodes at distance at most i from u_0. Let $i = D+1$ where D is the diameter of G. Thus, after the ith phase of Algorithm `Increasing-DFS`, all edges have been traversed, and thus exploration has been completed. If $k \geq \alpha D \log d$, then a stack of $D+1$ elements on $\log d$ bits, and a constant number of scalar variables, can be stored in the robot's memory, for $\alpha = O(1)$ large enough. Thus, when $i = D+1$, the exploration is completed using no more than k bits. Hence any graph of diameter D and maximum degree d can be explored. □

A direct consequence of Theorem 3 is the following:

Corollary 3. *All graphs of diameter D and maximum degree d can be explored by a robot using $O(D \log d)$ memory bits.*

The bound of Corollary 3 is tight (cf. Corollary 2).

As a final remark, observe that algorithm `Increasing-DFS` uses an infinite memory to explore some graphs of bounded size. Nevertheless, this phenomenon cannot be overcome by any exploration algorithm. Indeed, surprisingly, any infinite automaton that explores all graphs is required to use an infinite amount of memory to explore some finite graphs. In particular, for $d \geq 0$, let $\mathcal{G}_d$ be the set of all edge-colored d-regular graphs ($\mathcal{G}_d \neq \emptyset$ as witnessed by, e.g., the hypercube Q_d, or two nodes linked by d parallel edges). We have the following:

Theorem 4. *For any (infinite deterministic) automaton $\mathcal{R}$ that explores all graphs, and for any $G \in \mathcal{G}_d$, $\mathcal{R}$ uses infinitely many memory states when exploring G.*

Proof. Let $\mathcal{R}$ be an automaton that explores all graphs, and let $G \in \mathcal{G}_d$. As a consequence of Theorem 1, $\mathcal{R}$ is an *infinite* automaton $(X, Y, \mathcal{S}, \delta, \lambda, S_0)$, i.e., $|\mathcal{S}|$ is unbounded. Assume, for the purpose of contradiction, that $\mathcal{R}$ uses K states of $\mathcal{S}$ when executed in G, starting from some node, say u_0. Let $\mathcal{R}'$ be the automaton obtained by restricting $\mathcal{R}$ to the diagram induced by these K states of $\mathcal{S}$. More precisely, $\mathcal{R}' = (X, Y, \mathcal{S}', \delta', \lambda', S_0)$ where $\mathcal{S}'$ is the set of the K states visited by $\mathcal{R}$ when exploring G starting from u_0, λ' is λ restricted to $\mathcal{S}'$, and δ' is δ restricted to $\mathcal{S}' \times X$. Let $\mathcal{G}_d(\mathcal{R}')$ be the set of pairs (H, v_0) where $H = (V, E)$ is an edge-labeled graph and $v_0 \in V$, such that, starting at v_0 in H, $\mathcal{R}'$ visits only nodes of degree d and traverses only edges that have identical labels at their two extremities. Let (H, v_0) be the trap for $\mathcal{R}'$ constructed in the proof of Theorem 1. By our construction, we have $(H, v_0) \in \mathcal{G}_d(\mathcal{R}')$. Moreover, since $G \in \mathcal{G}_d$, we also have $(G, u_0) \in \mathcal{G}_d(\mathcal{R}')$. Let $(S_i)_{i \geq 0}$ be the sequence of states of $\mathcal{R}'$ when exploring G starting from u_0. By construction of $\mathcal{R}'$, $(S_i)_{i \geq 0}$ is also the sequence of states of $\mathcal{R}$ when exploring G starting from u_0. In fact, we have $\{S_i, i \geq 0\} = \mathcal{S}'$, and $S_{i+1} = \delta'(S_i, \lambda'(S_i, d)) = \delta(S_i, \lambda(S_i, d))$. Therefore, the sequence $(S_i)_{i \geq 0}$ is independent of any instance (graph, starting node) $\in \mathcal{G}_d(\mathcal{R}')$, and is independent of which automaton $\mathcal{R}$ or $\mathcal{R}'$ is exploring that instance. In particular, the sequence $(S_i)_{i \geq 0}$ is the same for $\mathcal{R}$ and $\mathcal{R}'$ in (H, v_0). Therefore,

the sequences of nodes visited by $\mathcal{R}$ and $\mathcal{R}'$ when exploring H starting from v_0 are identical. Since (H, v_0) is a trap for $\mathcal{R}'$, this latter fact is in contradiction with the fact that $\mathcal{R}$ is universal, and thus explores all graphs, including H. Hence $\mathcal{R}$ uses an infinite number of states when exploring G. □

5 Conclusion and Future Work

As mentioned in the introduction, Rollik [34] proved that no finite set of finite automata can separately (i.e., non-cooperatively) explore all undirected graphs. For his proof, Rollik constructed a trap for q robots of K states each, that is, a graph that none of the q robots explores completely. This trap is of size $O(K^q)$. Thus, an interesting direction of research is to look for smaller traps. In particular, we raise the question of whether there exists a trap of polynomial size for any set of q robots of K states each.

References

1. S. Albers and M. R. Henzinger. Exploring unknown environments. SIAM J. Computing 29:1164-1188, 2000.
2. H. Antelmann, L. Budach and H.A. Rollik. On universal traps. Elektronische Informationsverarbeitung und Kybernetic, EIK 15(3):123-131, 1979.
3. G. Asser. Bemerkungen zum Labyrinth-Problem. Elektronische Informationsverarbeitung und Kybernetic, EIK 13(4-5):203-216, 1977.
4. B. Awerbuch, M. Betke, R. Rivest and M. Singh. Piecemeal graph learning by a mobile robot. In 8th Conf. on Comput. Learning Theory, pages 321-328, 1995.
5. E. Bar-Eli, P. Berman, A. Fiat and R. Yan. On-line navigation in a room. J. Algorithms 17:319-341, 1994.
6. A. Bar-Noy, A. Borodin, M. Karchmer, N. Linial, and M.Werman. Bounds on universal sequences. SIAM J. Computing 18(2):268-277, 1989.
7. M. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In 30th Ann. Symp. on Theory of Computing (STOC), pages 269-278, 1998.
8. M. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In 35th Ann. Symp. on Foundations of Computer Science (FOCS), pages 75-85, 1994.
9. A. Blum, P. Raghavan and B. Schieber. Navigating in unfamiliar geometric terrain. SIAM J. Computing 26:110-137, 1997.
10. M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In 19th Symposium on Foundations of Computer Science (FOCS), pages 132-142, 1978.
11. M. Blum and W. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In 18th Ann. Symp. on Foundations of Computer Science (FOCS), pages 147-161, 1977.
12. M. Betke, R. Rivest and M. Singh. Piecemeal learning of an unknown environment. Machine Learning 18:231-254, 1995.
13. L. Budach. On the solution of the labyrinth problem for finite automata. Elektronische Informationsverarbeitung und Kybernetic, EIK 11(10-12):661-672, 1975.

14. L. Budach. Environments, labyrinths and automata. In Fund. Computat. Theory (FCT), LNCS 56, 54-64, 1977.
15. L. Budach. Automata and labyrinths. Math. Nachrichten, pages 195-282, 1978.
16. W. Coy. Automata in labyrinths. In Fund. Computat. Theory (FCT), LNCS 56, 65-71, 1977.
17. X. Deng, T. Kameda and C. H. Papadimitriou How to learn an unknown environment I: the rectilinear case. J. ACM 45:215-245, 1998.
18. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. J. Graph Theory 32:265-297, 1999.
19. K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree Exploration with Little Memory. In 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 588-597, 2002.
20. K. Döpp. Automaten in Labyrinthen. Elektronische Informationsverarbeitung und Kybernetic, EIK 7(2):79-94 & 7(3):167-190, 1971.
21. G. Dudek, M. Jenkins, E. Milios, and D. Wilkes. Robotic Exploration as Graph Construction. IEEE Transaction on Robotics and Automation 7(6):859-865, 1991.
22. C. Duncan, S. Kobourov and V. Kumar. Optimal constrained graph exploration. In 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 807-814, 2001.
23. P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective Tree Exploration To appear in 6th Latin American Theoretical Informatics Symposium (LATIN), 2004.
24. P. Fraigniaud, and D. Ilcinkas. Directed Graphs Exploration with Little Memory. In 21st Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 1996, pages 246-257, 2004.
25. A. Hemmerling. Labyrinth Problems: Labyrinth-Searching Abilities of Automata. Volume 114 of Teubner-Texte zur Mathematik. B. G. Teubner Verlagsgesellschaft, Leipzig, 1989.
26. F. Hoffmann. One pebble does not suffice to search plane labyrinths. In Fund. Computat. Theory (FCT), LNCS 117, 433-444, 1981.
27. D. Kozen. Automata and planar graphs. In Fund. Computat. Theory (FCT), 243-254, 1979.
28. A. López-Ortiz and S. Schuierer. On-line parallel heuristics and robot searching under the competitive framework. In 8th Scandinavian Workshop on Algorithm Theory (SWAT), 2002.
29. H. Müller. Endliche Automaten und Labyrinthe. Elektronische Informationsverarbeitung und Kybernetic, EIK 7/4, 261-264, 1971.
30. H. Müller. Automata catching labyrinths with at most three components. Elektronische Informationsverarbeitung und Kybernetic, EIK 15(l-2):3-9, 1979.
31. P. Panaite and A. Pelc, Exploring unknown undirected graphs, J. Algorithms 33:281-295, 1999.
32. M.O. Rabin, Maze threading automata. Seminar talk presented at the University of California at Berkeley, October 1967.
33. N. Rao, S. Kareti, W. Shi, and S. lyengar. Robot navigation in unknown terrains: Introductory survey of length,non-heuristic algorithms. Tech. Report ORNL/TM-12410, Oak Ridge National Lab., 1993.
34. H.A. Rollik. Automaten in planaren Graphen. Acta Informatica 13:287-298, 1980 (also in LNCS 67, pages 266-275, 1979).
35. CL. E. Shannon. Presentation of a maze-solving machine. In 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics), pages 173-180, 1951.

On Polynomially Time Bounded Symmetry of Information

Troy Lee[1] and Andrei Romashchenko[2*]

[1] CWI and University of Amsterdam
tlee@cwi.nl
[2] Institute for Information Transmission Problems
anromash@mccme.ru

Abstract. The information contained in a string x about a string y is defined as the difference between the Kolmogorov complexity of y and the conditional Kolmogorov complexity of y given x, i.e., $I(x:y) = \mathrm{C}(y) - \mathrm{C}(y|x)$. From the well-known Kolmogorov–Levin Theorem it follows that $I(x:y)$ is symmetric up to a small additive term $O(\log \mathrm{C}(x,y))$. We investigate if this property can hold for several versions of polynomial time bounded Kolmogorov complexity.

In particular, we study symmetry of information for some variants of distinguishing complexity CD where $\mathrm{CD}(x)$ is the length of a shortest program which accepts x and only x. We show relativized worlds where symmetry of information does not hold for deterministic and nondeterministic polynomial time distinguishing complexities $\mathrm{CD}^{\mathrm{poly}}$ and $\mathrm{CND}^{\mathrm{poly}}$ For nondeterministic polynomial time distinguishing with randomness, $\mathrm{CAMD}^{\mathrm{poly}}$, we prove that symmetry of information holds for most pairs of strings in any set in NP. In proving this last statement we extend a recent result of Buhrman et al. [6], which may be of independent utility.

1 Introduction

One of the most beautiful theorems in Kolmogorov Complexity is the principle of "Symmetry of Information", independently proven by Kolmogorov and Levin [22]. Roughly speaking, symmetry of information states that for any two strings x and y, the information contained in x about y is equal to the information contained in y about x, up to logarithmic factors. More formally, letting $\mathrm{C}(x)$ be the length of a shortest program which prints x, and $\mathrm{C}(y|x)$ be the length of a shortest program which prints y when given input x, symmetry of information can be stated as $\mathrm{C}(y) - \mathrm{C}(y|x) \approx \mathrm{C}(x) - \mathrm{C}(x|y)$. Besides its inherent attractiveness, this principle has also seen applications in diverse areas of theoretical computer science, for example in [1,10,21].

In this paper, we investigate the principal of symmetry of information when resource bounds are placed on the program to describe y given x. While the

* Supported in part by Russian Science Support Foundation and Russian Foundation for Basic Research, grants #03-01-0047 and #02-01-22001.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 463–475, 2004.
© Springer-Verlag Berlin Heidelberg 2004

argument of [22] can be used without modification to show that symmetry of information holds for programs using exponential time or polynomial space, things become trickier with polynomial time bounds. Though this question has been around for some time, few definite answers are known, see Sect. 7.1 of [14] for a survey. The series of works [15,16] give evidence that symmetry of information does not hold for polynomial time printing complexity: in particular, they show that the existence of certain kinds of one-way functions implies that polynomial time symmetry of information does not hold for printing complexity. Intuitively, if f is a polynomial time computable one-way function, and $f(x) = y$, then y is simple given x. On the other hand, if x is simple in polynomial time given y then this would provide a way to invert the function, by cycling through all small programs.

En route to showing that BPP is in the polynomial hierarchy, Sipser [19] introduced a relaxation of printing complexity called distinguishing complexity, denoted CD. For a string x, $\mathrm{CD}(x)$ is the length of a shortest program which accepts x and only x. Note that while printing and distinguishing complexity are equivalent up to a constant additive factor without resource bounds, this is not believed to be the case in the polynomial time setting [4,8].

The arguments of [15,16] leave open the question if symmetry of information can hold for distinguishing complexity. Now if f is a polynomial time computable one-way function and $f(x) = y$, then $\mathrm{CD}^{\mathrm{poly}}(x|y)$ is constant, as with a description of f, on input z we accept if and only if $f(z) = y$.

We investigate this possibility in considering symmetry of information for the following distinguishing complexity measures:

- Polynomial time distinguishing complexity, denoted $\mathrm{CD}^{\mathrm{poly}}(x)$.
- Nondeterministic polynomial time distinguishing complexity, which is denoted $\mathrm{CND}^{\mathrm{poly}}$.
- Nondeterministic polynomial time distinguishing with randomness, denoted $\mathrm{CAMD}^{\mathrm{poly}}$.

We show relativized worlds where symmetry of information fails in a strong way for $\mathrm{CD}^{\mathrm{poly}}$ and $\mathrm{CND}^{\mathrm{poly}}$. On the other hand, we show that for any set $A \in \mathrm{NP}$ symmetry of information holds for most pairs of strings $\langle x, y\rangle \in A$ with respect to the measure $\mathrm{CAMD}^{\mathrm{poly}}$. To prove this last statement we extend a recent result of [6], which may be of independent interest.

Another interesting approach to the definition of time-bounded Kolmogorov complexity is L. Levin's Kt complexity introduced in [13]. Recently D. Ronneburger proved that symmetry of information does not hold for Kt complexity in a very strong sense [18].

1.1 Background

Denote by C^{poly} a version of polynomial time-bounded Kolmogorov complexity, which can be $\mathrm{C}^{\mathrm{poly}}$, $\mathrm{CD}^{\mathrm{poly}}$, $\mathrm{CND}^{\mathrm{poly}}$, or $\mathrm{CAMD}^{\mathrm{poly}}$. To formulate the problem

of symmetry of information more precisely, we isolate three associated properties. The first is the *Easy Direction of Symmetry of Information:*

$$\begin{array}{l}\text{For any polynomial } p \text{ there exists a}\\ \text{polynomial } q \text{ such that for all } x, y:\\ C^q(x,y) \leq C^p(x) + C^p(y|x) + O(\log(n)).\end{array} \quad \text{(EDSI)}$$

Next is the *Hard Direction of Symmetry of Information:*

$$\begin{array}{l}\text{For any polynomial } p \text{ there exists a}\\ \text{polynomial } q \text{ such that for all } x, y:\\ C^q(x) + C^q(y|x) \lesssim C^p(x,y),\end{array} \quad \text{(HDSI)}$$

where the inequality should hold up to an additive logarithmic term, or at least up to $o(|x| + |y|)$.

Finally we also consider the property of *Symmetry of Mutual Information:*

$$\begin{array}{l}\text{For any polynomial } p \text{ there exists a}\\ \text{polynomial } q \text{ such that for all } x, y:\\ C^q(x) + C^q(y|x) \lesssim C^p(y) + C^p(x|y)\end{array} \quad \text{(SMI)}$$

Notice that if both (EDSI) and (HDSI) hold for a complexity measure C, then also (SMI) holds for C. The property (EDSI) is quite easy to prove for $\mathrm{C}^{\mathrm{poly}}$, $\mathrm{CD}^{\mathrm{poly}}$, $\mathrm{CND}^{\mathrm{poly}}$, or $\mathrm{CAMD}^{\mathrm{poly}}$. For the sake of completeness, in Sect. 3 we present a proof of this statement for distinguishing complexities.

It was shown in [15,16] that (HDSI) is not true for $\mathrm{C}^{\mathrm{poly}}$ if certain kinds of one-way functions exist (using a relativization from [9] we get a relativized world where (HDSI) is not true for $\mathrm{C}^{\mathrm{poly}}$). Are (HDSI) and (SMI) true for $\mathrm{CD}^{\mathrm{poly}}$ and $\mathrm{CND}^{\mathrm{poly}}$? It was claimed (without a complete proof) in [3] that these properties do not hold in some relativized world. In this paper and [12] we give the first published proof of this fact. Let us note that non-relativized variants of the properties above are probably very hard to prove, as the answer depends on such computational complexity problems as "P = NP?" and "P = PSPACE?".

1.2 Our Results

We show a relativized world where property (HDSI) fails in a very strong way for $\mathrm{CD}^{\mathrm{poly}}$ and $\mathrm{CND}^{\mathrm{poly}}$. In fact, we show that there is a set A and a polynomial p such that for any constant k, any polynomial q, for infinitely many $\langle x, y\rangle \in A$

$$k \cdot \mathrm{CD}^{p(n),A}(x,y) < \mathrm{CD}^{q(n),A}(x) + \mathrm{CD}^{A,q(n)}(y|x),$$

where $n = |x| + |y|$. Analogously, for any $\epsilon > 0$ there is a set A and a polynomial p such that for any polynomial q, for infinitely many $\langle x, y\rangle \in A$

$$(2-\epsilon)\mathrm{CND}^{p(n),A}(x,y) < \mathrm{CND}^{q(n),A}(x) + \mathrm{CND}^{q(n),A}(y|x),$$

$n = |x| + |y|$. The bound $(2-\epsilon)$ in the last inequality is tight as

$$2\mathrm{CND}^{A,\mathrm{poly}}(x,y) > \mathrm{CND}^{A,\mathrm{poly}}(x) + \mathrm{CND}^{A,\mathrm{poly}}(y|x) - O(1).$$

We also find relativizations where property (SMI) fails in a strong way for $\mathrm{CD}^{\mathrm{poly}}$ and $\mathrm{CND}^{\mathrm{poly}}$.

We do not know if property (HDSI) holds absolutely for $\mathrm{CAMD}^{\mathrm{poly}}$ complexity. We are able to prove, however, that it holds for most pairs $\langle x, y\rangle$ in any set $A \in \mathrm{NP}$. More generally, the following property holds:

$$\begin{aligned}&\text{For any set } A \text{ and any polynomial } p \text{ there is a polynomial } q \text{ such}\\ &\text{that for most pairs of strings } \langle x, y\rangle \in A \cap \{0,1\}^n \text{ it holds that}\\ &\mathrm{CAMD}^{p,A}(x, y) \geq \mathrm{CAMD}^{q,A}(x) + \mathrm{CAMD}^{q,A}(y|x) - O(\log^3 n).\end{aligned} \tag{1}$$

We also unconditionally prove the following weaker form of property (HDSI):

$$\begin{aligned}&\text{For any polynomial } p(n) \text{ there exists a}\\ &\text{polynomial } q(n) \text{ such that for any } x, y\\ &\mathrm{C}^p(x, y) \geq \mathrm{CAMD}^q(x) + \mathrm{CAMD}^q(y|x) - O(\log^3(|x| + |y|)).\end{aligned} \tag{2}$$

1.3 Organization

The rest of the paper is organized as follows. In Sect. 3 we shortly discuss a few simple properties of $\mathrm{CD}^{\mathrm{poly}}$ complexity and note that (HDSI) and (SMI) do not for $\mathrm{CD}^{\mathrm{poly}}$ complexity with some oracles. In Sect. 4 we discuss a relativization such that (HDSI) and (SMI) are false for $\mathrm{CND}^{\mathrm{poly}}$ complexity in a strong way. Most of technical proofs in Sects. 3 and 4 are omitted. The full proofs can be found in [12]. In Sect. 5 we prove (1) and (2); technically it is the most difficult part of the paper.

2 Preliminaries

We use the following notation:

- denote by $\mathbb{B}$ the set $\{0,1\}$; similarly, $\mathbb{B}^n$ is the set of all binary strings of length n;
- denote by $|x|$ the length of a binary string x;
- denote by $\|A\|$ the cardinality of a finite set A;
- for a set $A \subset \mathbb{B}^*$ denote by $A^{=n}$ the set $\{x : x \in A \text{ and } |x| = n\}$.
- for a set of pairs of strings $A \subset \mathbb{B}^* \times \mathbb{B}^*$ denote by $A^{=n}$ the set $\{\langle x, y\rangle \in A : |x| + |y| = n\}$.

2.1 Kolmogorov Complexity Measures

We use notation for Kolmogorov complexity from [14]:

Definition 1. *The Kolmogorov complexity* $C(y|x)$ *is defined as*

$$\min_p\{|p| : U(p, x) = y\},$$

where U *is a universal recursive function.*

The choice of U affects the Kolmogorov complexity by at most an additive constant. In this paper we are interested in time bounded versions of Kolmogorov complexity.

Definition 2. *Printing complexity* $\mathrm{C}^t(y|x)$ *is defined as*

$$\mathrm{C}^t(y|x) = \min_p\{|p| : U(p,x) = y \text{ and } U(p,x) \text{ runs in at most } t(|x|+|y|) \text{ steps}\}$$

for a universal machine U. Also $\mathrm{C}^t(z) = \mathrm{C}^t(z|\lambda)$.

The choice of universal machine U affects $\mathrm{C}^t(x|y)$ by at most an additive constant and the time bound t by at most a $\log(t)$ multiplicative factor.

Definition 3. *Distinguishing complexity* $\mathrm{CD}^t(y|x)$ *is defined as the minimal length of a program* p *such that*

1. $U(p,x,y)$ *accepts,*
2. $U(p,x,z)$ *rejects* $\forall z \neq y$,
3. $U(p,x,z)$ *runs in at most* $t(|x|+|z|)$ *steps*

Once again, $\mathrm{CD}^t(z) = \mathrm{CD}^t(z|\lambda)$.

There are a few other variants of distinguishing complexity. In [4] a nondeterministic variant of distinguishing complexity was defined. This definition is very similar t Definition 3; the difference is that the universal machine is nondeterministic. This version of complexity is denoted CND^t, where t is a time bound. Further, in [6] a complexity based on the class AM was defined. In this case the universal machine is nondeterministic and probabilistic. This complexity is denoted CAMD^t. We also use *relativized* version of Kolmogorov complexities, allowing the universal machine to query an oracle set.

2.2 Language Compression Theorems

A fundamental theorem of Kolmogorov complexity, and one that is very useful in applications, is the following:

Theorem 1 (Language Compression Theorem). *For any recursively enumerable set A, and all* $x \in A^{=n}$ *we have* $\mathrm{C}(x) \leq \log \|A^{=n}\| + O(\log n)$.

This is as x can be described by its index in the enumeration of $A^{=n}$.

In particular, this theorem is essentially used in the proof of (HDSI) in the resource unbounded case given in [22]. Similarly, our results about resource bounded symmetry of information (both positive and negative) crucially rely on recent resource bounded language compression theorems. In [6] the following analogue of the Language Compression Theorem is shown for CND complexity.

Theorem 2 ([6]). *There is a polynomial* $p(n)$ *such that for any set* $A \subset \mathbb{B}^*$ *and for all* $x \in A^{=n}$ *we have* $\mathrm{CND}^{p,A^{=n}}(x) \leq \log \|A^{=n}\| + O(\delta(n))$ *where* $\delta(n) = (\sqrt{\log \|A^{=n}\| + \log(n)})\log(n))$.

Further [6] show that with the power of Arthur-Merlin protocols a Language Compression Theorem holds which is optimal up to an additive $\log^3 n$ term:

Theorem 3 ([6]). *There is a polynomial $p(n)$ such that for any set $A \subset \mathbb{B}^*$ and for all $x \in A^{=n}$ we have* $\mathrm{CAMD}^{p,A^{=n}}(x) \le \log \|A^{=n}\| + O(\log^3(n))$.

For comparison we remark that for CD complexity the situation is somewhat different. In [4] it is shown that there is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$ it holds that $\mathrm{CD}^{p(n),A^{=n}}(x) \le 2\log \|A^{=n}\| + O(\log n)$. Furthermore, [5] show that there is a set A where this bound is tight up to $O(\log n)$ terms. That is, the factor of 2 in general cannot be improved.

3 On CD Complexity

We start with a positive fact: (EDSI) holds for all versions of polynomial time complexity under consideration.

Proposition 1. *Let C denote* CD, CND *or* CAMD *complexity. Then for any polynomial $p(n)$ there exists a polynomial $q(n)$ such that for any oracle A and any x, y*

$$C^{q(n),A}(x,y) \le C^{p(n),A}(x) + C^{q(n),A}(y|x) + O(\log n),$$

where $n = |x| + |y|$.

In contrast to Proposition 1, inequality (HDSI) and (SMI) does not hold for $\mathrm{CD}^{\mathrm{poly}}$ complexity in some relativized worlds. The proof of the next proposition follows the idea outlined in [3]:

Proposition 2. *There exists an oracle A and a polynomial $p(n)$ satisfying the following condition. For any $\epsilon > 0$ and large enough n there exists a pair $\langle x, y\rangle \in \mathbb{B}^n \times \mathbb{B}^n$ such that*

- $\mathrm{CD}^{2^{\epsilon n},A^{=2n}}(y) > (1-\epsilon)n - O(1)$,
- $\mathrm{CD}^{2^{\epsilon n},A^{=2n}}(x|y) > (1-\epsilon)n - O(\log n)$,
- $\mathrm{CD}^{p(n),A^{=2n}}(x) = O(1)$,
- $\mathrm{CD}^{p(n),A^{=2n}}(y|x) = O(1)$ *and even* $\mathrm{C}^{p(n),A^{=2n}}(y|x) = O(1)$,

i.e., $\mathrm{CD}^{p(n),A^{=2n}}(x) + \mathrm{CD}^{p(n),A^{=2n}}(y|x) \ll \mathrm{CD}^{2^{\epsilon n},A^{=2n}}(y) + \mathrm{CD}^{2^{\epsilon n},A^{=2n}}(x|y)$. *Thus, (SMI) and (HDSI) does not hold with the oracle A.*

The full proofs see in [12].

4 On CND Complexity

We show that properties (HDSI) and (SMI) are not true for a relativized version of polynomial time bounded CND complexity.

Theorem 4. *Let $m = m(n), s = s(n), t = t(n)$ be functions such that $2^{s(n)} + 2^{m(n)} \leq 2^{3n}$ and $t(n)2^{m(n)} \leq 2^{3n-3}$. Then there is a polynomial $p(n)$, and sets A, X such that*

- $X^{=3n} \subset \mathbb{B}^{3n}$, $\|X^{=3n}\| = 2^{s(n)}$,
- $A^{=6n} \subset \mathbb{B}^{3n} \times \mathbb{B}^{3n}$,
- $\|\{y : (x,y) \in A^{=6n}\}\| \geq 7/8 \cdot 2^{3n}$ *for any* $x \in X^{=3n}$,
- $\| \bigcup_{x \notin X} \{y : (x,y) \in A^{=6n}\}\| \leq 1/8 \cdot 2^{3n}$,

and for large enough n, for all $x \in X^{=3n}$, for at least $3/4 \cdot 2^{3n}$ strings $y \in \mathbb{B}^{3n}$ the following conditions hold:

$$\begin{aligned} &\langle x,y\rangle \in A^{=6n}, \\ &\mathrm{CND}^{p,A^{=6n}}(x|y) \quad \leq s(n) + O(\delta(n)), \\ &\mathrm{CND}^{p,A^{=6n}}(x,y) \quad \leq (3n + s(n)) + O(\delta(n)), \\ &\mathrm{CND}^{t(n),A^{=6n}}(y|x) \geq 3n - O(1), \\ &\mathrm{CND}^{t(n),A^{=6n}}(x) \quad \geq m(n) - O(1), \end{aligned}$$

where $\delta(n) = \sqrt{n}\log(n)$.

Our proof of Theorem 4 is based on the Language Compression Theorem. Note that the term $\delta(n) = \sqrt{n}\log(n)$ above comes from Theorem 2. We omit the details here. The full proof see in [12].

Corollary 1. *There exists an oracle A such that a $\mathrm{CND}^{\mathrm{poly}}$ version of (HDSI) does not hold. Moreover, for any $\varepsilon > 0$ there exists a polynomial p such that for any polynomial q for large enough n*

$$(2-\varepsilon)\mathrm{CND}^{p,A^{=6n}}(x,y) \ll \mathrm{CND}^{q,A^{=6n}}(x) + \mathrm{CND}^{q,A^{=6n}}(y|x)$$

for most $\langle x,y\rangle \in A^{=6n}$.

Proof. It follows from Theorem 4 for $s(n) = \varepsilon n$, $m(n) = (3-\varepsilon)n$, $t(n) = 2^{\varepsilon n/2}$. The bound $(2-\varepsilon)$ in Corollary 1 is tight. This can be easily seen as,

$$\mathrm{CND}^{\mathrm{poly},A^{=6n}}(x,y) > \mathrm{CND}^{\mathrm{poly},A^{=6n}}(x) - O(1)$$

and $\mathrm{CND}^{\mathrm{poly},A^{=6n}}(x,y) \geq \mathrm{CND}^{\mathrm{poly},A^{=6n}}(y|x) - O(1)$. Hence for any oracle A

$$2\mathrm{CND}^{p,A^{=6n}}(x,y) \geq \mathrm{CND}^{q,A^{=6n}}(x) + \mathrm{CND}^{q,A^{=6n}}(y|x) - O(1).$$

Corollary 2. *There exists an oracle A such that a $\mathrm{CND}^{\mathrm{poly}}$ version of (SMI) does not hold, i.e., for some polynomial p, any polynomial q, and large enough n*

$$\mathrm{CND}^{p,A^{=6n}}(y) + \mathrm{CND}^{p,A^{=6n}}(x|y) \ll \mathrm{CND}^{q,A^{=6n}}(x) + \mathrm{CND}^{q,A^{=6n}}(y|x)$$

for most $\langle x,y\rangle \in A^{=6n}$.

Proof. Again, we can apply Theorem 4. Set $s(n) = n$, $m(n) = 2n$, $t(n) = 2^{\varepsilon n}$ (for small enough $\varepsilon > 0$). Than for most $\langle x,y\rangle \in A^{=6n}$ we have $\mathrm{CND}^{p,A^{=6n}}(y) + \mathrm{CND}^{p,A^{=6n}}(x|y) \leq 4n + O(1)$ and $\mathrm{CND}^{q,A^{=6n}}(x) + \mathrm{CND}^{q,A^{=6n}}(y|x) \geq 5n - O(1)$.

5 On CAMD Complexity

In this section we study symmetry of information under the CAMD complexity measure. In contrast to the case of CND complexity, with the power of nondeterminism and randomness we can prove some positive results, showing that some weaker versions of (HDSI) hold for CAMD.

Our proof will follow the proof in the resource unbounded case as given in [22]. We first review this proof to see how it can be used in our case. Let α, β be two strings such that $|\alpha| + |\beta| = n$, and suppose that $\mathrm{C}(\alpha, \beta) = m$. We define the set $A_{x,m} = \{y : \mathrm{C}(x, y) \leq m\}$. Notice that $\|A_{x,m}\| \leq 2^{m+1}$ and that given x and m the set $A_{x,m}$ is recursively enumerable. Thus as $\beta \in A_{\alpha,m}$ by the Language Compression Theorem (Theorem 1), $\mathrm{C}(\beta|\alpha) \leq \log \|A_{\alpha,m}\| + O(\log n)$. Let k^* be such that $2^{k^*} \leq \|A_{\alpha,m}\| < 2^{k^*+1}$. Then the above says that $\mathrm{C}(\beta|\alpha) \leq k^* + O(\log n)$.

Now consider the set $B_{m,k} = \{x : \|A_{x,m}\| \geq 2^k\}$. Notice that the size of $B_{m,k}$ is less than 2^{m-k}, and that $\alpha \in B_{m,k^*}$. The set $B_{m,k}$ is recursively enumerable given m, k, thus by the Language Compression Theorem, $\mathrm{C}(\alpha) \leq m - k^* + O(\log n)$. And so

$$\begin{aligned} \mathrm{C}(\alpha) + \mathrm{C}(\beta|\alpha) &\leq m - k^* + k^* + O(\log n) \\ &\leq \mathrm{C}(\alpha, \beta) + O(\log n) \end{aligned}$$

If we substitute polynomial time printing complexity in the above argument, then the set $A_{x,m}$ is in NP. Further, by the approximate lower bound counting property of AM [2] there is an AM algorithm which accepts with high probability for $x \in B_{m,k}$ and rejects with high probability for $x \notin B_{m,k-1}$. We have, however, no guarantee on the algorithm's behavior for $x \in B_{m,k-1}$. In the next theorem, we extend the language compression results of [6] to also work for AM gap sets of this type, allowing the above argument to go through.

Theorem 5. *Let $A \subseteq \mathbb{B}^*$. Suppose there is a polynomial time bound $q(n)$, and predicate Q such that*

- *for all $u \in A^{=n}$, $\Pr_{r \in \mathbb{B}^{q(n)}}[\exists v\, Q(u, v, r) = 1] \geq 2/3$*
- *$\|L = \{u \in \mathbb{B}^n : \Pr_{r \in \mathbb{B}^{q(n)}}[\exists v\, Q(u, v, r) = 1] \geq 1/3\}\| \leq 2^k$,*

and for all u, v, r the predicate $Q(u, v, r)$ can be computed in polynomial time. Then there is a polynomial time bound $p(n)$ such that for all $u \in A^{=n}$, we have $\mathrm{CAMD}^p(u) \leq k + O(\log^3 n)$.

Before going into the proof, we briefly recall the technique of [6]. Let $\mathrm{TR} : \mathbb{B}^n \times \mathbb{B}^d \to \mathbb{B}^m$ be the function underlying Trevisan's extractor [20], that is the composition of a good error correcting code with the Nisan-Wigderson generator [17]. The output of $\mathrm{TR}(u, e)$ is the evaluation of the Nisan-Wigderson generator on seed e when using $\hat{u}$ as the 'hard' function supplied to the generator, where $\hat{u}$ is the image of u under an error correcting code. The key property of this function, what makes it a good extractor and compressor, is that if $\mathrm{TR}(u, e)$ is not close to uniform over choice of $e \in \mathbb{B}^d$ on some set $B \subseteq \mathbb{B}^m$, then u has

a short description given oracle access to B. In [6] it is shown that u can be printed in polynomial time from this description and oracle access to B.

To give the elements of a set $A \in \mathbb{B}^n$ short descriptions, we let the set B be the image of $A \times \mathbb{B}^d$ under the the function TR. That is, $B = \cup_{x \in A} \cup_{e \in \mathbb{B}^d} \mathrm{TR}(x, e)$. Notice that for any $x \in A$, $\Pr_e[\mathrm{TR}(x, e) \in B] = 1$. On the other hand if we take m to be $\log \|A\| + d + 1$ then the probability that a uniformly chosen $y \in \mathbb{B}^m$ is in B is less than 1/2. Thus all the elements of A have a short description relative to B. Now notice that with nondeterminism and an oracle for A, we can decide membership in B, thus all the elements of A have a short CND^A description. The elements of A can be given an even more succinct CAMD^A description by using the randomness in the AM protocol to simulate part of the probabilistic argument in [17,20].

Proof. (Theorem 5) By amplification and the results of [7], we can transform the predicate Q into a predicate Q' taking random strings of length a polynomial $q'(n)$ and with the property

- if $u \in A^{=n}$ then $\Pr_r[\exists v\, Q'(u, v, r) = 1] = 1$
- $\|L' = \{u : \Pr_r[\exists v\, Q'(u, v, r) = 1] \geq 2^{-n-2}\}\| \leq 2^k$

for r chosen uniformly over $\mathbb{B}^{q'(n)}$.

For each $r \in \mathbb{B}^{q'(n)}$ we define a set

$$B_r = \{w \ : \ \exists u \in \mathbb{B}^n, \exists v, e\ \mathrm{TR}(u, e) = w \wedge Q'(u, v, r) = 1\}$$

Clearly if $u \in A^{=n}$, then $\Pr_e[B_r(\mathrm{TR}(u, e)) = 1] = 1$, for any $r \in \mathbb{B}^{q'(n)}$. We now calculate the probability that for a randomly chosen $w \in \mathbb{B}^m$ and randomly chosen $r \in \mathbb{B}^{q'(n)}$, that $w \in B_r$. As for a 0/1 variable the probability of being 1 is equal to the expectation of the variable, we have

$$\Pr_{r,w}[w \in B_r] = E_{r,w}[B_r(w)].$$

By linearity of expectation, we can divide the latter into two contributions, that from elements w for which $\exists u \in L'$ and seed e such that $\mathrm{TR}(u, e) = w$, and those w for which this is not the case.

$$E_{r,w}[B_r(w)] = \sum_{\substack{w = \mathrm{TR}(u,e) \\ u \in L'}} E[B_r(w)] + \sum_{\substack{w \neq \mathrm{TR}(u,e) \\ u \in L'}} E[B_r(w)]$$

By taking $m = k + d + 2$ the first term can be bounded by 1/4. The second term is bounded by $2^m 2^{-n-2} \leq 1/4$. Going back to probability notation, we have for any $u \in A^{=n}$

$$\Pr_{r,e}[B_r(\mathrm{TR}(u, e)) = 1] - \Pr_{r,w}[B_r(w) = 1] \geq 1/2.$$

It follows by the hybrid argument that there is an $i \in [m]$ such that

$$\Pr_{x,r,r'}[B_r(\hat{u}_1(x) \ldots \hat{u}_{i-1}(x)\hat{u}(x)r]' - \Pr_{x,r,r',b}[B_r(\hat{u}_1(x) \ldots \hat{u}_{i-1}(x)br')] \geq 1/2m \quad (3)$$

Let $F(x,b,r') = \hat{u}_1(x)\dots\hat{u}_{i-1}(x)br'$. Our algorithm to approximate $\hat{u}$ will do the following: on input x, choose uniformly at random b,r,r' and evaluate $B_r(F(x,b,r'))$; if this evaluates to 1, then output b, otherwise output $1-b$. Call the output of this algorithm $g_b(x,r,r')$. It follows from equation (3) that

$$\Pr_{x,b,r,r'}[\hat{u}(x) = g_b(x,r,r')] \geq 1/2 + 1/2m$$

The rest of the argument now proceeds as in the proof for relativized language compression in AM (Theorem 3 in [6]), to show that the computation of $g_b(x,r,r')$ can be approximated by an AM algorithm.

As the proof of Theorem 5 relativizes, this argument can be used with respect to any set A of pairs of strings, not just the set of pairs with complexity at most m.

Theorem 6. *There is a polynomial $p(n)$ such that for any set $A \subset \mathbb{B}^* \times \mathbb{B}^*$ and all $\langle x,y\rangle \in A^{=n}$*

$$\log \|A^{=n}\| \geq \mathrm{CAMD}^{p,A^{=n}}(x) + \mathrm{CAMD}^{p,A^{=n}}(y|x) - O(\log^3 n).$$

Furthermore, if $A \in \mathrm{NP}$ then then there is a polynomial $q(n)$ such that

$$\log \|A^{=n}\| \geq \mathrm{CAMD}^{q}(x) + \mathrm{CAMD}^{q}(y|x) - O(\log^3 n).$$

Proof. Fix n and $\langle \alpha,\beta\rangle \in A^{=n}$. Denote $m = \log\|A^{=n}\|$ and $A_x = \{y : (x,y) \in A^{=n}\}$. Membership in the set A_x can be decided in polynomial time given x and the oracle $A^{=n}$. As $\beta \in A_\alpha$ it follows from Theorem 3 that $\mathrm{CAMD}^{q,A^{=n}}(\beta|\alpha) \leq \log\|A_\alpha\| + O(\log^3 n)$.

Now consider the set $B_k = \{x : \|A_x\| \geq 2^k\}$. Let k^* be such that $2^{k^*} \leq \|A_\alpha\| < 2^{k^*+1}$. Then $\alpha \in B_{k^*}$. Again by the approximate lower bound counting property of AM, as shown in [2], there is a predicate Q (computable in polynomial time given the oracle $A^{=n}$) such that

- If $x \in B_k$ then $\Pr_r[\exists y Q(x,y,r) = 1] \geq 2/3$
- If $x \notin B_{k-1}$ then $\Pr_r[\exists y Q(x,y,r) = 1] \leq 1/3$

Thus if $\Pr_r[\exists y\ Q(x,y,r) = 1] > 1/3$ then $x \in B_{k-1}$. However $\|A^{=n}\| = 2^m$ implies that $\|B_{k-1}\| \leq 2^{m-k+1}$. Now by Theorem 3 we obtain $\mathrm{CAMD}^{q,A^{=n}}(\alpha) \leq m - k^* + O(\log^3 n)$.

Putting the above together we have

$$\mathrm{CAMD}^{q,A^{=n}}(\alpha) + \mathrm{CAMD}^{q,A^{=n}}(\beta|\alpha) \leq m - k^* + k^* + O(\log^3 n) \leq m + O(\log^3 n)$$

which gives the first statement of the theorem.

To prove the "furthermore", note that approximate lower bound counting of NP sets can be done in AM [2], and apply Theorem 5 to give the bound on (unrelativized) CAMD complexity of NP sets.

Corollary 3. *For any set* $A \subset \mathbb{B}^* \times \mathbb{B}^*$ *and any polynomial* $p(n)$ *there is a polynomial* q *such that for all but at most a* $1/n$ *fraction of* $\langle x, y\rangle \in A^{=n}$,

$$\mathrm{CAMD}^{p(n),A^{=n}}(x,y) \geq \mathrm{CAMD}^{q(x),A^{=n}}(x) + \mathrm{CAMD}^{q,A^{=n}}(y|x) - O(\log^3 n).$$

Furthermore, if $A \in \mathrm{NP}$ *then*

$$\mathrm{CAMD}^{p(n)}(x,y) \geq \mathrm{CAMD}^{q}(x) + \mathrm{CAMD}^{q}(y|x) - O(\log^3 n).$$

Proof. For all but at most a $1/n$ fraction of $\langle x, y\rangle \in A^{=n}$ we have

$$\mathrm{CAMD}^{p(n),A^{=n}}(x,y) \geq \log \|A^{=n}\| - O(\log n).$$

Applying Theorem 6 we get the first statement of the corollary. Applying the "furthermore" of Theorem 6 gives the furthermore here.

Theorem 7. *For any strings* $x, y \in \mathbb{B}^n$, *and polynomial* $p(n)$ *there is a polynomial* $q(n)$ *such that* $\mathrm{C}^p(x,y) \geq \mathrm{CAMD}^q(x) + \mathrm{CAMD}^q(y|x) - O(\log^3 n)$.

Proof. Fix a pair of strings $\langle \alpha, \beta\rangle$. Let $n = |\alpha| + |\beta|$, and suppose that $\mathrm{C}^p(\alpha,\beta) = m$. Consider the set $A = \{\langle x, y\rangle : C^p(x,y) \leq m\}$. As membership in A can be decided in nondeterministic polynomial time, we may invoke the "furthermore" of Theorem 6 to give $\log \|A\| \geq \mathrm{CAMD}^q(\alpha) + \mathrm{CAMD}^q(\beta|\alpha) - O(\log^3 n)$ for some polynomial q. On the other hand, $\|A\| \leq 2^{m+1}$, and the theorem is proven.

From Theorem 7 we obtain as a corollary a result of [16], up to an additive $O(\log^3(n))$ factor.

Corollary 4. *If* P = NP *then for any polynomial* $p = p(n)$ *there is a polynomial* $q = q(n)$ *such that for all* $x, y \in \mathbb{B}^n$

$$\mathrm{C}^p(x,y) \geq \mathrm{C}^q(x) + \mathrm{C}^q(y|x) - O(\log^3 n)$$

It remains an interesting open question if polynomial time symmetry of information for printing complexity holds under a weaker assumption than P = NP.

Acknowledgments. We would like to thank Harry Buhrman and Lance Fortnow for helpful comments on [3] and [4] and on the history of the problems under consideration. We also thank Harry Buhrman and Dieter van Melkebeek for helpful comments and conversations about Sect. 5, and Detlef Ronneburger for sharing his result on Kt complexity.

References

1. E. Allender, H. Buhrman, M. Koucky, D. van Melkebeek, and D. Ronneburger. Power from random strings. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science,* pages 669–678. IEEE, 2002.
2. L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing,* pages 421–429. ACM, 1985.
3. H. Buhrman and L. Fortnow. Distinguishing complexity and symmetry of information. Technical Report TR-95-11, Department of Computer Science, The University of Chicago, 1995.
4. H. Buhrman, L. Fortnow, and S. Laplante. Resource bounded Kolmogorov complexity revisited. *SIAM Journal on Computing,* 31(3):887–905, 2002.
5. H. Buhrman, S. Laplante, and P.B. Miltersen. New bounds for the language compression problem. In *Proceedings of the 15th IEEE Conference on Computational Complexity,* pages 126–130. IEEE, 2000.
6. H. Buhrman, T. Lee, and D. van Melkebeek. Language compression and pseudorandom generators. To appear in 19th IEEE Conference on Computational Complexity, 2004.
7. M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Randomness and Computation,* volume 5 of *Advances in Computing Research,* pages 429–442. JAI Press, Greenwich, 1989.
8. L. Fortnow and M. Kummer. On resource-bounded instance complexity. *Theoretical Computer Science A,* 161:123–140, 1996.
9. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way functions. In *Proceedings of the 21st ACM Symposium on the Theory of Computing,* pages 41–61. ACM, 1989.
10. T. Jiang, J. Seiferas, and P. Vitányi. Two heads are better than two tapes. *Journal of the ACM,* 44(2):237–256, 1997.
11. A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems Information Transmission,* 1(1):1–7, 1965.
12. T. Lee, A. Romashchenko. On Polynomially Time Bounded Symmetry of Information. Electronic Colloquium on Computational Complexity, Report TR04-031, April 2004.
13. L. A. Levin. Universal Search Problems. *Problems Information Transmission,* 9(3):265–266, 1973.
14. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, New York, second edition, 1997.
15. L. Longpré and S. Mocas. Symmetry of information and one-way functions. *Information Processing Letters,* 46(2):95–100, 1993.
16. L. Longpré and O. Watanabe. On symmetry of information and polynomial time invertibility. *Information and Computation,* 121(l):14–22, 1995.
17. N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences,* 49:149–167, 1994.
18. D. Ronneburger. Personal Communication, 2004.
19. M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing,* pages 330–335. ACM, 1983.
20. L. Trevisan. Construction of extractors using pseudo-random generators. In *Proceedings of the 31st ACM Symposium on the Theory of Computing,* pages 141–148. ACM, 1999.

21. N. Vereshchagin and P. Vitányi. Kolmogorov's structure function with an application to the foundations of model selection. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science,* pages 751–760. IEEE, 2002.
22. A. Zvonkin and L. Levin. The complexity of finite objects and the algorithmic concepts of information and randomness. *Russian Mathematical Surveys,* 25:83–124, 1970.

Scaled Dimension and the Kolmogorov Complexity of Turing-Hard Sets*

John M. Hitchcock[1**], María López-Valdés[2], and Elvira Mayordomo[2]

[1] Department of Computer Science, University of Wyoming, USA.
jhitchco@cs.uwyo.edu.

[2] Departamento de Informática e Ingeniería de Sistemas, María de Luna 1, Universidad de Zaragoza, 50018 Zaragoza, SPAIN.
{marlopez,elvira}@unizar.es

Abstract. Scaled dimension has been introduced by Hitchcock et al. (2003) in order to quantitatively distinguish among classes such as $\mathbf{SIZE}(2^{\alpha n})$ and $\mathbf{SIZE}(2^{n^\alpha})$ that have trivial dimension and measure in ESPACE.

This paper gives an exact characterization of effective scaled dimension in terms of resource-bounded Kolmogorov complexity. We can now view each result on the scaled dimension of a class of languages as upper and lower bounds on the Kolmogorov complexity of the languages in the class.

We prove a Small Span Theorem for Turing reductions that implies the class of $\leq_T^{P/poly}$-hard sets for ESPACE has $(-3)^{rd}$-pspace dimension 0. As a consequence we have a nontrivial upper bound on the Kolmogorov complexity of all hard sets for ESPACE for this very general nonuniform reduction, $\leq_T^{P/poly}$. This is, to our knowledge, the first such bound. We also show that this upper bound does not hold for most decidable languages, so $\leq_T^{P/poly}$-hard languages are unusually simple.

1 Introduction

The relationship between uniform and nonuniform complexity measures is one of the main sources of open problems in computational complexity. In this context it is very informative to quantify the difference in size of nonuniform and uniform complexity classes and this has been possible so far for space-bounded complexity classes. Lutz started in [20] by showing that the Boolean circuit-size complexity class $\mathbf{SIZE}(\frac{2^n}{n})$ has measure 0 in ESPACE (linear exponential space). In 2000, Lutz [21] defined resource-bounded dimension as an effectivization of Hausdorff dimension. He refined his previous result by proving that $\mathbf{SIZE}(\alpha\frac{2^n}{n})$ has dimension α in ESPACE for each $\alpha \in [0,1]$.

* This research was supported in part by Spanish Government MEC project TIC 2002-04019-C03-03.

** Part of this research was done while this author was visiting the Universidad de Zaragoza.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 476–487, 2004.
© Springer-Verlag Berlin Heidelberg 2004

But classes such as $\mathrm{SIZE}(2^{\alpha n})$ or $\mathrm{SIZE}\,(2^{n^{\alpha}})$ resisted the dimension approach because they have trivial dimensions. In 2003, the definition of scaled dimension [9] made it possible to precisely quantify the difference between those circuit-size classes. Several results on scaled dimension of other classes have also been proven [9,8].

In this paper we explore the interpretation of space-bounded scaled dimension as an information content measure. Space-bounded Kolmogorov complexity has been investigated extensively [15,6,22,16,19,4,12]. Roughly speaking, for $A \subseteq \{0,1\}^*$, $m \in \mathbb{N}$ and space bound t, the space-bounded Kolmogorov complexity $KS^t(A[0\ldots m-1])$ is the length of the shortest program that prints the m-bit prefix of characteristic sequence of A using at most t units of workspace. In this setting $KS^t(A[0\ldots m-1])$ is the amount of information of $A[0\ldots m-1]$ that is accessible by a t-bounded space computation.

Here we obtain an exact characterization of space-bounded scaled dimension in terms of space-bounded Kolmogorov complexity for the cases where such a characterization is possible. For example, the $(-1)^{\mathrm{st}}$-scaled pspace-dimension of a class X is the minimum s for which there is a c such that for every $A \in X$,

$$KS^{m^c}(A[0\ldots m-1]) \leq m - m^{1-s} \text{ i.o. } m$$

Equivalently, $(-1)^{\mathrm{st}}$-scaled pspace-dimension of X is the the minimum s for which there is a c that for every $A \in X$,

$$KS^{2^{cn}}(A_{\leq n}) \leq 2^{n+1} - 2^{n(1-s)} \text{ i.o. } n$$

This means that the $(-1)^{\mathrm{st}}$-scaled pspace-dimension of X is directly related to the best i.o. upper bound (equivalently, to the worse a.e. lower bound) of the form $2^{n+1} - 2^{n\alpha}$ on the space-bounded Kolmogorov complexity of all languages in X.

We can now interpret each scaled dimension result as a Kolmogorov complexity (tight) upper bound.

Juedes and Lutz in [14] prove a measure Small Span Theorem for $\leq_{\mathrm{T}}^{\mathrm{P/poly}}$-reductions in ESPACE. This result says that for each $A \in \mathrm{ESPACE}$, either the class of languages reducible to A (lower span) or the class of problems to which A can be reduced (upper span) has pspace-measure 0. Here, we improve this result by showing that, for $\leq_{\mathrm{T}}^{\mathrm{P/poly}}$-reductions, for each $A \in \mathrm{ESPACE}$, either the lower span or the upper span of A has $(-3)^{\mathrm{rd}}$-scaled pspace-dimension 0. This result also improves the scaled dimension Small Span Theorem for polynomial-time many-one reductions given by Hitchcock in [8].

In particular, the class of $\leq_{\mathrm{T}}^{\mathrm{P/poly}}$-hard sets for ESPACE has $(-3)^{\mathrm{rd}}$-scaled pspace-dimension 0 and therefore unusually low space-bounded Kolmogorov complexity.

Juedes and Lutz extensively study in [14] upper and lower bounds on $KS(A_{=n})$. Their best results for hard languages are for the $\leq_{\mathrm{m}}^{\mathrm{P/poly}}$-reductions, that is, non-uniform but many-one reductions. Our result is the first nontrivial upper bound on the Kolmogorov complexity of $\leq_{\mathrm{T}}^{\mathrm{P/poly}}$-hard sets.

We also show that the upper bound we have obtained on the Kolmogorov complexity of these (very general) hard languages is very unusual, because it does not hold for most decidable languages (in the sense of both measure and scaled dimension).

The paper is organized as follows. Section 2 reviews the main concepts in scaled dimension. Section 3 contains our characterization and section 4 our results for the Kolmogorov complexity of hard sets.

Notation Summary

For each $n \in \mathbb{N}$, we will use both

$$A_{=n} = A \cap \{0,1\}^n \text{ and } A_{=n} = A[2^n - 1 \ldots 2^{n+1} - 2]$$
$$A_{\leq n} = A \cap \{0,1\}^{\leq n} \text{ and } A_{\leq n} = A[0 \ldots 2^{n+1} - 2]$$

We use $\log^{(i)}$ for the i-times iterated application of log, $\log^{(i)}(x) = \overbrace{\log(\ldots \log}^{i} x)$.

$\mathrm{E}_1\mathrm{SPACE} = \mathrm{ESPACE} = \mathrm{DSPACE}(2^{O(n)})$ and $\mathrm{E}_2\mathrm{SPACE}{=}\mathrm{DSPACE}(2^{n^{O(1)}})$. DEC is the class of decidable languages. comp is the class of computable functions, $\mathrm{p}_1\mathrm{space}{=}$ pspace is the class of polynomial-space computable functions. The definition of $\mathrm{p}_i\mathrm{space}$ and $\mathrm{E}_i\mathrm{SPACE}$ can be found for instance in [21].

$\mathrm{R}(\mathrm{comp}) = \mathrm{DEC}$ and $\mathrm{R}(\mathrm{p}_i\mathrm{space}) = \mathrm{E}_i\mathrm{SPACE}$, for $i \geq 1$.

Throughout this paper, Δ denotes one of the classes comp, $\mathrm{p}_i\mathrm{space}$ (for some $i \geq 1$).

2 Scaled Dimension

Hitchcock, Lutz and Mayordomo [9] introduced resource-bounded scaled dimension. In this section we review the essentials of this theory and state some useful properties.

A scale is a function $g : H \times [0, +\infty) \to \mathbb{R}$ where $H = (a, \infty)$ for some $a \in \mathbb{R} \cup \{-\infty\}$. A scale must satisfy certain properties stated in [9]. Here we concentrate in the following family of scales.

Definition. We define $g_i : H_i \times [0, \infty) \to \mathbb{R}$ and $g_{-i} : H_i \times [0, \infty) \to \mathbb{R}$ by recursion on $i \in \mathbb{N}$ as follows:

$$g_0(m, s) = ms.$$
$$g_{i+1}(m, s) = 2^{g_i(\log m, s)}$$
$$g_{-i}(m, s) = \begin{cases} m + g_i(m, 0) - g_i(m, 1 - s) & \text{for } 0 \leq s \leq 1 \\ g_i(m, s) & \text{for } s > 1 \end{cases}$$

The domain of g_i coincides with that of g_{-i} and is of the form $H_i = (a_i, \infty)$, where $a_0 = -\infty$ and $a_{i+1} = 2^{a_i}$. We write $m_i = m_{-i} = \max\{0, a_i\}$.

Example 1. For $0 \leq s \leq 1$,

$$g_3(m,s) = 2^{2^{(\log\log m)^s}}$$
$$g_2(m,s) = 2^{(\log m)^s}$$
$$g_1(m,s) = m^s$$
$$g_0(m,s) = ms$$
$$g_{-1}(m,s) = m + 1 - m^{1-s}$$
$$g_{-2}(m,s) = m + 2 - 2^{(log m)^{1-s}}$$
$$g_{-3}(m,s) = m + 2^2 - 2^{2^{(\log\log m)^{1-s}}}$$

Scaled dimension is defined using functions called scaled gales [9]. Particular cases of this concept are gales and martingales.

Definition. Let $k \in \mathbb{Z}$ and $s \in [0,\infty)$.

1. An $s^{(k)}$-gale is a function $d : \{0,1\}^* \to [0,\infty)$ satisfying
$$d(w) = 2^{-g_k(|w|+1,s)+g_k(|w|,s)}[d(w0) + d(w1)]$$
for all $w \in \{0,1\}$ with $|w| \geq m_k$.
2. An s-gale is a $s^{(0)}$-gale, that is, a function $d : \{0,1\}^* \to [0,\infty)$ satisfying
$$d(w) = 2^{-s}[d(w0) + d(w1)]$$
for all $w \in \{0,1\}^*$.
3. A martingale is a 0-gale, that is, a function $d : \{0,1\}^* \to [0,\infty)$ satisfying
$$d(w) = \frac{d(w0) + d(w1)}{2}$$
for all $w \in \{0,1\}^*$.

Definition. Let $k \in \mathbb{Z}$, $s \in [0,+\infty)$ and d be an $s^{(k)}$-gale.

1. We say that d succeeds on a sequence $S \in \mathbf{C}$ if
$$\limsup_{n\to\infty} d(S[0\ldots n]) = \infty$$
2. The success set of d is
$$S^\infty[d] = \{S \in \mathbf{C} \mid d \text{ succeeds on } S\}$$

Definition. Let $X \subseteq \mathbf{C}$ and $k \in \mathbb{Z}$.

1. $\mathcal{G}_\Delta^{(k)}(X)$ is the set of all $s \in [0,\infty)$ such that there is a Δ-computable $s^{(k)}$-gale d such that $X \subseteq S^\infty[d]$.
2. The k^{th}-order scaled Δ-dimension of X is
$$\dim_\Delta^{(k)}(X) = \inf \mathcal{G}_\Delta^{(k)}(X)$$

3. The k^{th}-order scaled dimension of X in $R(\Delta)$ is

$$\dim^{(k)}(X|R(\Delta)) = \dim_{\Delta}^{(k)}(X \cap R(\Delta))$$

For the scale $g_0(m,s) = ms$ we have the resource-bounded dimensions defined in [21].

$$\dim_{\Delta}(X) = \dim_{\Delta}^{(0)}(X)$$
$$\dim(X|R(\Delta)) = \dim^{(0)}(X|R(\Delta))$$

For every X, k and Δ, $\dim_{\Delta}^{(k)}(X)$ is in [0,1], and $\dim_{\Delta}^{(k)}(R(\Delta)) = 1$ [9].

For this reason Δ is the right resource bound for dimension in $R(\Delta)$. If $\Delta \in \{\text{comp}, \text{p}_i\text{space}\}$,

$$\dim_{\Delta}^{(k)}(X) \leq \dim_{\Delta}^{(k+1)}(X)$$

We are particularly interested in the case $\Delta = \text{pspace}$, that is, $\dim_{\text{pspace}}^{(k)}(X)$ and $\dim^{(k)}(X|\text{ESPACE})$.

We will use pspace-measure and measure in ESPACE when referring to results in [14].

Definition. A class $X \subseteq \mathbf{C}$ has pspace-measure 0 (denoted by $\mu_{\text{pspace}}(X) = 0$) iff there exists a martingale $d \in \text{pspace}$ such that, $X \subseteq S^{\infty}[d]$.

A class $X \subseteq \mathbf{C}$ has pspace-measure 1 (denoted by $\mu_{\text{pspace}}(X) = 1$) iff X^c has pspace-measure 0.

A class $X \subseteq \mathbf{C}$ has measure 0 in ESPACE iff $X \cap \text{ESPACE}$ has pspace-measure 0. This is denoted by $\mu(X|\text{ESPACE}) = 0$.

A class $X \subseteq \mathbf{C}$ has measure 1 in ESPACE iff X^c has measure 0 in ESPACE. This is denoted by $\mu(X|\text{ESPACE}) = 1$.

A basic result relating measure and dimension is the following

Proposition 1. *[21,9] Let $X \subseteq \mathbf{C}$, $k \in \mathbb{Z}$*

1. *If $\dim_{\text{pspace}}^{(k)}(X) < 1$, then $\mu_{\text{pspace}}(X) = 0$.*
2. *If $\dim^{(k)}(X|\text{ESPACE}) < 1$, then $\mu(X|\text{ESPACE}) = 0$.*

Finally, we will use the following inverses of g_k.

Definition. Let $i \in \mathbb{N}$. We define $f_i : \mathbb{N} \times [0,\infty) \to \mathbb{R}$ and $f_{-i} : \mathbb{N} \times [0,\infty) \to \mathbb{R}$ by

$$f_i(n,x) = \frac{\log^{(i)} x}{\log^{(i)} n}$$
$$(f_{-i})(n,x) = 1 - \frac{\log^{(i)}(n-x)}{\log^{(i)} n}$$

Notice that for each $k \in \mathbb{Z}$, $x \in [0,\infty)$, f_k tends to the inverse of g_k, that is,

$$g_k(n, f_k(n,x)) = x + \epsilon(n)$$

where $\lim_n \epsilon(n) = 0$.

3 Characterization

In this section we examine the relationship between scaled dimension and space-bounded Kolmogorov complexity. We start with a brief summary of definitions and notation for resource-bounded Kolmogorov complexity.

Definition. Given a Turing machine M, $t : \mathbb{N} \to \mathbb{N}$, $S \in \mathbf{C}$ and $n \in \mathbb{N}$

1. The t-space-bounded Kolmogorov complexity of $S[0 \ldots n-1]$ is,

$$KS_M^{t(n)}(S[0 \ldots n-1]) = \min\left\{|\pi| \;\middle|\; M(\pi) = S[0 \ldots n-1] \text{ in } \leq t(n) \text{ space}\right\}$$

2. The t-time-bounded Kolmogorov complexity of $S[0 \ldots n-1]$ is,

$$K_M^{t(n)}(S[0 \ldots n-1]) = \min\left\{|\pi| \;\middle|\; M(\pi) = S[0 \ldots n-1] \text{ in } \leq t(n) \text{ time}\right\}$$

Well-know techniques [17] show that there is a machine U that is optimal in the sense that for each machine M there is a constant c such that for all total computable t, $S \in \mathbf{C}$ and $n \in \mathbb{N}$, we have

$$KS_U^{ct(n)+c}(S[0 \ldots n-1]) \leq KS_M^{t(n)}(S[0 \ldots n-1]) + c$$

$$K_U^{ct(n)\log(t(n))}(S[0 \ldots n-1]) \leq K_M^{t(n)}(S[0 \ldots n-1]) + c$$

As usual, we fix an optimal machine U and omit it from the notation. Our characterization will use the following notation.

Definition. Let $S \in \mathbf{C}$, $k \in \mathbb{Z}$, t a resource bound.

$$\mathcal{KS}_{(k)}^t(S) = \liminf_n f_k(n, KS^{t(n)}(S[0 \ldots n-1]))$$

$$\mathcal{K}_{(k)}^t(S) = \liminf_n f_k(n, K^{t(n)}(S[0 \ldots n-1]))$$

For instance, for k between -2 and 2 we have

$$\mathcal{KS}_{(2)}^t(S) = \liminf_n \frac{\log\log(KS^{t(n)}(S[0 \ldots n-1]))}{\log\log n}$$

$$\mathcal{KS}_{(1)}^t(S) = \liminf_n \frac{\log(KS^{t(n)}(S[0 \ldots n-1]))}{\log n}$$

$$\mathcal{KS}_{(0)}^t(S) = \liminf_n \frac{KS^{t(n)}(S[0 \ldots n-1])}{n}$$

$$\mathcal{KS}_{(-1)}^t(S) = \liminf_n \left(1 - \frac{\log(n - KS^{t(n)}(S[0 \ldots n-1]))}{\log n}\right)$$

$$\mathcal{KS}^t_{(-2)}(S) = \liminf_n \left(1 - \frac{\log\log(n - KS^{t(n)}(S[0\ldots n-1]))}{\log\log n}\right)$$

The following observation states the precise meaning of $\mathcal{KS}$ in terms of i.o. upper bounds.

Observation 1. *Let $k \in \mathbb{Z}$ and $S \in \mathbf{C}$. Let t be a resource bound. Then,*

$$\mathcal{KS}^t_{(k)}(S) = \inf\{s \in [0,\infty) | \exists^\infty n\, KS^{t(n)}(S[0\ldots n-1]) < g_k(n,s)\}$$

For example, $\mathcal{KS}^t_{(-1)}(S)$ is the smallest s for which

$$KS^{t(n)}(S[0\ldots n-1]) < n - n^{1-s} \text{ i.o. } n$$

For classes of languages we take the worse case upper bound.

Definition. Let $X \subseteq \mathbf{C}$, $k \in \mathbb{Z}$ and $j \in \mathbb{N}$

$$\mathcal{KS}^{\mathrm{p}_j\mathrm{space}}_{(k)}(X) = \inf_{t \in \mathrm{p}_j\mathrm{space}} \sup_{S \in X} \mathcal{KS}^t_{(k)}(S)$$

$$\mathcal{K}^{\mathrm{comp}}_{(k)}(X) = \inf_{t \in \mathrm{comp}} \sup_{S \in X} \mathcal{K}^t_{(k)}(S)$$

The main result in this section is the following characterization of scaled-dimension.

Theorem 2. *Let $X \subseteq \mathbf{C}$*

1. *For all $i, j \in \mathbb{N}$ with $i \leq j$*

$$\dim^{(i)}_{\mathrm{p}_j\mathrm{space}}(X) = \mathcal{KS}^{\mathrm{p}_j\mathrm{space}}_{(i)}(X)$$

$$\dim^{(-i)}_{\mathrm{p}_j\mathrm{space}}(X) = \mathcal{KS}^{\mathrm{p}_j\mathrm{space}}_{(-i)}(X)$$

2. *For all $k \in \mathbb{Z}$*

$$\dim^{(k)}_{\mathrm{comp}}(X) = \mathcal{K}^{\mathrm{comp}}_{(k)}(X)$$

A similar characterization for the cases $\dim^{(-i)}_{\mathrm{p}_j\mathrm{space}}$ $(i > j)$ is not possible because it is known (Theorem 3.3 in [14]) that for each $A \in \mathrm{ESPACE}$ there is an $\epsilon > 0$ such that $KS^{2^{2n}}(A_{\leq n}) < 2^{n+1} - 2^{\epsilon n}$ a.e. n, therefore

$$\mathcal{KS}^{\mathrm{pspace}}_{(-2)}(\mathrm{ESPACE}) = 0$$

whereas we know that $\dim^{(-2)}_{\mathrm{pspace}}(\mathrm{ESPACE}) = 1$ [21].

The case $i = 0$ corresponds to resource-bounded dimension and is proven in [7].

A dual version of Theorem 2 can be proven for the packing or strong dimension as characterized in [3].

Theorem 2 is proven from the next two lemmas. The first one states that dimension is smaller that $\mathcal{KS}$, and it only holds for $i \leq j$ in the space-bounded case.

Lemma 1. *Let* $X \subseteq \mathbf{C}$

1. *For all* $i, j \in \mathbb{N}$ *with* $i \leq j$

$$\dim^{(i)}_{\mathrm{p}_j\mathrm{space}}(X) \leq \mathcal{KS}^{\mathrm{p}_j\mathrm{space}}_{(i)}(X)$$

$$\dim^{(-i)}_{\mathrm{p}_j\mathrm{space}}(X) \leq \mathcal{KS}^{\mathrm{p}_j\mathrm{space}}_{(-i)}(X)$$

2. *For all* $k \in \mathbb{Z}$

$$\dim^{(k)}_{\mathrm{comp}}(X) \leq \mathcal{K}^{\mathrm{comp}}_{(k)}(X)$$

The second inequality, $\mathcal{KS}$ smaller than dimension, holds without restriction on the scale used. This will be useful in the next section.

Lemma 2. *Let* $X \subseteq \mathbf{C}$. *For all* $j \in \mathbb{N}$, $k \in \mathbb{Z}$

$$\mathcal{KS}^{\mathrm{p}_j\mathrm{space}}_{(k)}(X) \leq \dim^{(k)}_{\mathrm{p}_j\mathrm{space}}(X)$$

$$\mathcal{K}^{\mathrm{comp}}_{(k)}(X) \leq \dim^{(k)}_{\mathrm{comp}}(X)$$

Lemma 2 holds for the polynomial time scaled dimension and the corresponding polynomial-time-bounded Kolmogorov complexity. This can be proven by using the techniques in [11] and [10].

Our characterization also holds when restricting to Kolmogorov complexity of prefixes of the form $A_{\leq n}$, except for the 0^{th}-scale case.

Theorem 3. *Let* $X \subseteq \mathbf{C}$,

1. *For all* $i, j \in \mathbb{N}$ *with* $0 < i \leq j$,

$$\dim^{(i)}_{\mathrm{p}_j\mathrm{space}}(X) < s$$

iff there is a $t \in \mathrm{p}_j\mathrm{space}$ *such that for any* $A \in X$

$$KS^{t(2^{n+1})}(A_{\leq n}) < g_i(2^{n+1}, s) \text{ i.o. } n.$$

2. *For all* $i, j \in \mathbb{N}$ *with* $0 < i \leq j$,

$$\dim^{(-i)}_{\mathrm{p}_j\mathrm{space}}(X) < s$$

iff there is a $t \in \mathrm{p}_j\mathrm{space}$ *such that for any* $A \in X$

$$KS^{t(2^{n+1})}(A_{\leq n}) < g_{-i}(2^{n+1}, s) \text{ i.o. } n.$$

3. *For all* $k \in \mathbb{Z}$, $k \neq 0$,

$$\dim^{(k)}_{\mathrm{comp}}(X) < s$$

iff there is a $t \in \mathrm{comp}$ *such that for any* $A \in X$

$$KT^{t(2^{n+1})}(A_{\leq n}) < g_k(2^{n+1}, s) \text{ i.o. } n.$$

For example for the pspace case, $\dim^{(-1)}_{\text{pspace}}(X) < s$ iff there is a c such that for any $A \in X$

$$KS^{2^{c(n+1)}}(A_{\leq n}) < 2^{n+1} - 2^{(n+1)(1-s)} \text{ i.o. } n.$$

As a final remark, notice that it is not equivalent in general to consider $KS(A_{=n})$ and $KS(A_{\leq n})$. Whereas $KS^{2^{cn}}(A_{=n}) < 2^n - 2^{\epsilon n}$ implies that $KS^{2^{c'n}}(A_{\leq n}) < 2^{n+1} - 2^{\epsilon n}$, the quantity $KS^{2^{cn}}(A_{\leq n})$ can be much lower than $KS(A_{=n})$, relative to the corresponding length. Juedes and Lutz extensively study $KS(A_{=n})$ in [14], mainly for languages in ESPACE and languages that are $\leq^{\text{P/poly}}_{\text{m}}$-hard, that is, hard for many-one non-uniform reductions.

4 The Kolmogorov Complexity of Hard and Weakly Hard Sets

In this section we are interested in adaptive nonuniform reductions in the class ESPACE, namely P/poly-Turing reductions ($\leq^{\text{P/poly}}_{\text{T}}$) which are nonuniform Turing reductions that can be computed by polynomial-size circuits.

The lower and upper spans are defined as follows.

Definition. Let $A \subseteq \{0,1\}^*$

1. The $\leq^{\text{P/poly}}_{\text{T}}$- lower span of A is
$$(\text{P/poly})_{\text{T}}(A) = \{B \subseteq \{0,1\}^* | B \leq^{\text{P/poly}}_{\text{T}} A\}$$
2. The $\leq^{\text{P/poly}}_{\text{T}}$- upper span of A is
$$(\text{P/poly})^{-1}_{\text{T}}(A) = \{B \subseteq \{0,1\}^* | A \leq^{\text{P/poly}}_{\text{T}} B\}$$

Juedes and Lutz [14] prove the following Small Span Theorem for these reductions.

Theorem 4. *[14] For every* $A \in \text{ESPACE}$

$$\mu((\text{P/poly})_{\text{T}}(A) | \text{ESPACE}) = 0$$

or

$$\mu_{\text{pspace}}((\text{P/poly})^{-1}_{\text{T}}(A)) = 0.$$

This theorem states that for each $A \in \text{ESPACE}$, at least one of the lower and upper spans of A is small in the sense of resource-bounded measure.

Small Span Theorems for the class of exponential time languages and polynomial time reductions have been studied for both measure and dimension [13, 1,18,2,5,8].

Here we prove the following generalization of Theorem 4.

Theorem 5. *For every* $A \in \mathrm{ESPACE}$,

$$\dim^{(1)}(\,(\mathrm{P/poly})_{\mathrm{T}}(A) \mid \mathrm{ESPACE}) = 0$$

or

$$\dim^{(-3)}_{\mathrm{pspace}}(\,(\mathrm{P/poly})^{-1}_{\mathrm{T}}(A)\,) = 0.$$

The proof is based in the following lemma.

Lemma 3.

$$\dim^{(1)}\left(\left\{A \subseteq \{0,1\}^* \;\middle|\; \dim^{(-3)}_{\mathrm{pspace}}(\,(\mathrm{P/poly})^{-1}_{\mathrm{T}}(A)\,) > 0\right\} \;\middle|\; \mathrm{ESPACE}\right) = 0.$$

We use the lemma to establish our Small Span Theorem.

Proof of Theorem 5.

Let $Z = \left\{A \subseteq \{0,1\}^* \;\middle|\; \dim^{(-3)}_{\mathrm{pspace}}(\,(\mathrm{P/poly})^{-1}_{\mathrm{T}}(A)\,) > 0\right\}$. We consider two cases.

1. Suppose that $(\mathrm{P/poly})_{\mathrm{T}}(A) \cap \mathrm{ESPACE} \subseteq Z$. Then it follows from Lemma 3 that $\dim^{(1)}((\mathrm{P/poly})_{\mathrm{T}}(A) \mid \mathrm{ESPACE}) = 0$.
2. Otherwise there is a language $B \in (\mathrm{P/poly})_{\mathrm{T}}(A) \cap \mathrm{ESPACE} \cap Z^c$. Then we have $(\mathrm{P/poly})^{-1}_{\mathrm{T}}(A) \subseteq (\mathrm{P/poly})^{-1}_{\mathrm{T}}(B)$ and

$$\dim^{(-3)}_{\mathrm{pspace}}(\,(\mathrm{P/poly})^{-1}_{\mathrm{T}}(A)\,) \leq \dim^{(-3)}_{\mathrm{pspace}}(\,(\mathrm{P/poly})^{-1}_{\mathrm{T}}(B)\,) = 0.$$

□

Theorem 5 generalizes Theorem 4 because $\dim^{(-3)}(X) < 1$ implies $\mu_{\mathrm{pspace}}(X) = 0$. Hitchcock shows in in [8] that $(-2)^{\mathrm{nd}}$-scaled Small Span Theorems are not possible, since for A a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete language for ESPACE, $\dim^{(-2)}_{\mathrm{pspace}}(\mathrm{P}^{-1}_{\mathrm{m}}(A)) = 1$. Therefore we can't substitute –3 by a bigger scale in the statement of Theorem 5.

Because of the connections we have obtained between scaled dimension and Kolmogorov complexity we can conclude the following.

Theorem 6. *For every* $A \in \mathrm{ESPACE}$, *if*

$$\dim^{(1)}((\mathrm{P/poly})_{\mathrm{T}}(A) | \mathrm{ESPACE}) > 0$$

then

$$\mathcal{KS}^{(-3)}_{\mathrm{pspace}}((\mathrm{P/poly})^{-1}_{\mathrm{T}}(A)) = 0$$

Proof. The theorem follows from Theorem 5 and Lemma 2.

□

In particular for hard languages we have the following corollary.

Corollary 1. *Let* $\mathcal{H}$ *be the class of languages that are* $\leq^{\mathrm{P/poly}}_{\mathrm{T}}$*-hard for* ESPACE. *Then*

$$\mathcal{KS}^{(-3)}_{\mathrm{pspace}}(\mathcal{H}) = 0,$$

that is, for each $\epsilon > 0$ *there is a* c *such that for every* $\leq^{\mathrm{P/poly}}_{\mathrm{T}}$*-hard* H

$$\mathcal{KS}^{2^{cn}}(H_{\leq n}) < 2^{n+1} - 2^{2^{(\log n)^{1-\epsilon}}} \quad i.o.\ n$$

This corollary tells us that $\leq_T^{P/poly}$-hard languages are unusually simple, since for most languages the opposite holds, even when allowing any resource bound on the Kolmogorov complexity.

Theorem 7. *For very resource bound t, the class of all sets A such that for any ϵ*

$$KS^{t(2^n)}(A_{\leq n}) < 2^{n+1} - 2^{2^{(\log n)^{1-\epsilon}}} \quad i.o.\ n$$

has $(-3)^{rd}$ comp-*dimension 0.*

Proof. The result follows from our characterization in Theorem 2. □

Theorem 7 implies that most decidable languages (in a very strong sense) don't have the upper bound on Kolmogorov complexity that $\leq_T^{P/poly}$-hard languages have.

Notice that the best known lower bound on the Kolmogorov complexity of $\leq_T^{P/poly}$-hard sets is Theorem 4.1. in [14], stating that for each $\leq_T^{P/poly}$-hard H there is an $\epsilon > 0$ such that

$$KS^{2^{n^\epsilon}}(H_{\leq n}) > 2^{n^\epsilon} \text{ a.e. } n$$

5 Conclusion

We have obtained a Kolmogorov complexity characterization of scaled dimension for the cases where such a characterization is possible. We expect that the combination of fractal and information theory techniques will produce interesting results on the problem of uniform vs nonuniform complexity.

Acknowledgement. We thank Jack Lutz for very helpful discussions on reference [14].

References

1. K. Ambos-Spies, W. Merkle, J. Reimann, and F. Stephan. Hausdorff dimension in exponential time. In *Proceedings of the 16th IEEE Conference on Computational Complexity,* pages 210–217, 2001.
2. K. Ambos-Spies, H.-C. Neis, and S. A. Terwijn. Genericity and measure for exponential time. *Theoretical Computer Science,* 168(1):3–19, 1996.
3. K. B. Athreya, J. M. Hitchcock, J. H. Lutz, and E. Mayordomo. Effective strong dimension in algorithmic information and computational complexity. In *Proceedings of the Twenty–First Symposium on Theoretical Aspects of Computer Science,* volume 2996 of *Lecture Notes in Computer Science,* pages 632–643. Springer-Verlag, 2004.
4. J. L. Balcázar and R. V. Book. Sets with small generalized Kolmogorov complexity. *Acta Informatica,* 23:679–688, 1986.
5. H. Buhrman and D. van Melkebeek. Hard sets are hard to find. *Journal of Computer and System Sciences,* 59(2):327–345, 1999.

6. J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the 24th IEEE Symposium on the Foundations of Computer Science,* pages 439–445, 1983.
7. J. M. Hitchcock. *Effective Fractal Dimension: Foundations and Applications.* PhD thesis, Iowa State University, 2003.
8. J. M. Hitchcock. Small spans in scaled dimension. In *Proceedings of the 19th IEEE Conference on Computational Complexity,* 2004. To appear.
9. J. M. Hitchcock, J. H. Lutz, and E. Mayordomo. Scaled dimension and nonuniform complexity. *Journal of Computer and System Sciences.* To appear. Preliminary version appeared in *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming,* pages 278–290, 2003.
10. J. M. Hitchcock and N. V. Vinodchandran. Dimension, entropy rates, and compression. In *Proceedings of the 19th IEEE Conference on Computational Complexity,* 2004. To appear.
11. J.M. Hitchcock and E. Mayordomo. Base invariance of feasible dimension. Manuscript, 2003.
12. D. T. Huynh. Resource-bounded Kolmogorov complexity of hard languages. In *Structure in Complexity Theory,* pages 184–195, Berlin, 1986. Springer-Verlag.
13. D. W. Juedes and J. H. Lutz. The complexity and distribution of hard problems. *SIAM Journal on Computing,* 24(2):279–295, 1995.
14. D. W. Juedes and J. H. Lutz. Completeness and weak completeness under polynomial-size circuits. *Information and Computation,* 125(1):13–31, 1996.
15. A. N. Kolmogorov. Three approaches to the quantitative definition of 'information'. *Problems of Information Transmission,* 1:1–7, 1965.
16. L. A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control,* 61:15–37, 1984.
17. M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, Berlin, 1997. Second Edition.
18. W. Lindner. On the polynomial time bounded measure of one-truth-table degrees and p-selectivity, 1993. Diplomarbeit, Technische Universität Berlin.
19. L. Longpré. *Resource Bounded Kolmogorov Complexity, a Link Between Computational Complexity and Information Theory.* PhD thesis, Cornell University, 1986. Technical Report TR-86-776.
20. J. H. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences,* 44(2):220–258, 1992.
21. J. H. Lutz. Dimension in complexity classes. *SIAM Journal on Computing,* 32:1236–1259, 2003.
22. M. Sipser. A complexity-theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing,* pages 330–335, 1983.

A Geometric Approach to Parameterized Algorithms for Domination Problems on Planar Graphs

Henning Fernau[1] and David Juedes[2]

[1] Universität Tübingen, WSI für Informatik, Sand 13, 72076 Tübingen, Germany and The University of Newcastle, Department of Electrical Engineering and Computer Science, University Drive, Callaghan, NSW 2308, Australia,
fernau@{informatik.uni-tuebingen.de,cs.newcastle.edu.au}
[2] School of E.E & C.S., Ohio University, Athens, OH, 45701 U.S.A.
juedes@ohio.edu

Abstract. This paper revisits design and analysis techniques for fixed parameter algorithms for PLANAR DOMINATING SET and other problems on planar structures. As our main result, we use new geometric arguments concerning treewidth-based algorithms to show that determining whether a planar graph G has a dominating set of size k can be solved in $O(2^{16.4715\sqrt{k}} + n^3)$ steps. This result improves on the best known treewidth-based algorithm by Kanj and Perkovič that runs in time $O(2^{27\sqrt{k}}n)$. Our main result nearly matches the new branchwidth-based algorithm for PLANAR DOMINATING SET by Fomin and Thilikos that runs in time $O(2^{15.13\sqrt{k}}k+n^3)$. Algorithms for other problems on planar structures are explored. In particular, we show that PLANAR RED/BLUE DOMINATING SET can be solved in time $O(2^{24.551\sqrt{k}}n)$. This leads to the main results, namely, that faster parameterized algorithms can be obtained for a variety of problems that can be described by planar boolean formulae. This gives the best-known parameterized algorithms for PLANAR VERTEX COVER, PLANAR EDGE DOMINATING SET, and FACE COVER.

1 Introduction

A problem is said to be *fixed parameter tractable* [7] if it can be solved in time $f(k)n^c = O(n^c)$ for some fixed constant c and every fixed value of the "parameter" k. Such fixed parameter algorithms may be of great practical interest, even when the original problem is NP-complete, if the value of the parameter is small for most (or all) practical instances of the problem. This is the case for the many NP-problems concerning programming languages because the treewidth of structured (i.e., goto-free) programs is usually small [14]. Nevertheless, the function $f(k)$ plays a crucial role in the range of practical applicably of fixed parameter algorithms. In this vein, Alber *et al.* [2,1] made progress on fixed parameter algorithms for PLANAR DOMINATING SET by giving a new treewidth-based algorithm and proving an upper bound on the running time of the algorithm of

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 488–499, 2004.
© Springer-Verlag Berlin Heidelberg 2004

$O(2^{70\sqrt{k}}n)$. While this algorithm is asymptotically faster than the $O(8^k n)$ algorithm by Alber *et. al* [3], the large constant in the exponent of the running time would seem to indicate that the other algorithms are better choices for most reasonable values of k. As mentioned by Alber *et al.* [1], it may be the case, however, that analysis of treewidth-based algorithms for PLANAR DOMINATING SET can be substantially improved. Indeed, subsequent improvements were made Kanj and Perkovič [11], who gave a treewidth-based algorithm for PLANAR DOMINATING SET running in time $O(2^{27\sqrt{k}}n)$. Recently, Fomin and Thilikos [9] gave a new branchwidth-based algorithm for PLANAR DOMINATING SET that runs in time $O(k^4 + 2^{15.13\sqrt{k}}k + n^3)$.

This extended abstract revisits design and analysis techniques for fixed parameter algorithms for PLANAR DOMINATING SET and other related problems on planar structures. We use new purely geometric arguments concerning treewidth-based algorithms to show that determining whether a planar graph G has a dominating set of size k can be solved in $O(2^{16.4715\sqrt{k}} + n^3)$ steps. Algorithms for other problems on planar structures are also explored. In particular, we show that PLANAR RED/BLUE DOMINATING SET can be solved in time $O(2^{24.551\sqrt{k}}n)$. As observed by Cai, Fellows, Juedes, and Rosamond [5], fast parameterized algorithms for this problem lead to fast parameterized algorithms for each problem in the subclass Planar $\mathbf{TMIN_1}$ [5] of the syntactic class Planar TMIN defined by Khanna and Motwani [12]. This generic result allows us to achieve fast parameterized algorithms for PLANAR VERTEX COVER and FACE COVER. The corresponding algorithm for FACE COVER is by far the best known to date. We note that all proofs are omitted from this extended abstract. The omitted proofs are provided in the appendix for the program committee.

2 Problems of Interest

We primarily restrict attention to the following problems on planar graphs.

PLANAR DOMINATING SET : Given a planar graph G and an integer k, determine whether G has a *dominating set* of size k. A set D is a *dominating set* of an undirected graph $G = (V, E)$ if every vertex $v \in V - D$ is connected via an edge to a vertex in D.

PLANAR RED/BLUE DOMINATING SET : Given a planar bipartite graph $G = ([V_{red}, V_{blue}], E)$ and an integer k, determine whether there exists a set $V' \subseteq V_{red}$ such that every vertex $v \in V_{blue}$ is connected to some vertex in V'.

FACE COVER : Given an undirected plane graph $G = (V,E)$, i.e., the planar graph G along with an embedding of G in the plane, and an integer k, determine whether G has a *face cover* of size k. A *face cover* C of a plane graph $G = (V, E)$ is a set of faces that cover all vertices of G, i.e., for every vertex $v \in V$, there exists a face $f \in C$ so that v lies on the boundary of f.

PLANAR EDGE DOMINATING SET : Given an undirected planar graph $G = (V,E)$ and an integer k, determine whether there is an *edge domination set* $D \subseteq E$ of size k. An edge domination set of an undirected graph $G = (V, E)$ is a set $D \subseteq E$ which dominates all *edges* of G, i.e., for all $e \in E \setminus D$ there exists an edge in D which is adjacent to e.

We show that FACE COVER and PLANAR EDGE DOMINATING SET can be solved in time $O(2^{24.551\sqrt{k}}n)$, thus drastically improving on the constants given in [1,6].

3 The Structure of r-Outerplanar Graphs

We try to adhere to standard notions regarding graph theory, e.g., the open neighbourhood of a vertex v in a graph G is denoted by $N(v)$.

To simplify formulations, let us assume that planar graphs are given together with an embedding in the plane (a plane graph), since for the following algorithmic discussions an arbitrary embedding will do. This allows for some "geometric arguments." If C is an embedded cycle and $x \in C$ some vertex, then we can speak about the left and right neighbour of x on C, if we imagine ourselves to be placed in the center of the cycle. Similarly, if y is some point outside of C which is connected by edges to vertices $x_0, \ldots, x_{m-1}$ (and no others) on C (in that order from left to right, when "standing" in the center of C, viewing in direction of y), then there uniquely exists some j such that the region described by the edges $\{y, x_j\}$, $\{y, x_{(j+1) \bmod m}\}$ and the left-to-right path from $x_{(j+1) \bmod m}$ to x_j contains all edges from y to any x_i on C. We call $\{y, x_j, x_{(j+1) \bmod m}\}$ the *upper triple* associated to y and C (also in the boundary case when $m = 0$ and $m = 1$, when the triple will be $\{y\}$ and $\{y, x_0\}$, respectively). Likewise, if y is a point in the interior of C and z is another reference "point" within C, then there are points x, x' on C adjacent to y such that all other vertices x'' on C which are adjacent to y are on only one of the two sub-paths into which C is cut by x and x', while z is contained in the other "half." We will call $\{y, x, x'\}$ the *lower triple* associated to y and z.

Let us review the basic concepts related to r-outerplanar graphs. A planar graph is *outerplanar* if all vertices are on the same face. A planar graph is *r-outerplanar* if either $r = 1$ and it is outerplanar or $r > 1$ and, after deleting all vertices on the exterior face (which form the *exterior layer* L_1), an $(r-1)$-outerplanar graph is obtained. In this way, we may speak of the layers $L_1, \ldots, L_r$ of an r-outerplanar graph.

Let $x \in L_i$, where $i \geq 2$. Then the unique cycle $B(x)$ in layer L_{i-1} satisfying that x is contained in the open region enclosed by $B(x)$ and no other vertex of layer L_{i-1} is contained in this region, is called the *boundary cycle* or *ring* of x. The existence and uniqueness of such a boundary cycle $B(x)$ is easy to see.

A *ring decomposition* of an r-outerplanar graph G is a directed forest $\mathcal{F}_G$ of height $r-1$ which is described as follows: the nodes of the trees are rings of vertices of G and the different trees (may) correspond to different components of G. If B' and B'' are boundary cycles, then B'' is a child of B' iff $B(x) = B'$

for any $x \in B''$. The ith layer of $\mathcal{F}_G$ consists of c_i boundary cycles $B(i,1), \ldots, B(i,c_i)$ from the ith layer L_i of G, and some *interconnection pattern* $J_i = L_i \setminus B_i$ with $B_i = \bigcup_{j=1}^{c_i} B(i,j)$.

Since different components of a graph can be processed separately in the algorithms presented below, we can assume that $\mathcal{F}_G$ is in fact a (directed) tree. Notationally, we stress this by writing $\mathcal{T}_G$ instead of $\mathcal{F}_G$. If $\mathcal{T}_G$ happens to be a path, we will call G *simple r-outerplanar*. The tree $\mathcal{F}'_G$ is obtained from $\mathcal{F}_G$ by attaching a child to any leaf $B(i,j)$ of $\mathcal{F}_G$.

Let us finally stress the fact that a ring decomposition may be different from the so-called layer decomposition introduced in [1]; e.g., two rings may share an edge, see Fig. 1.

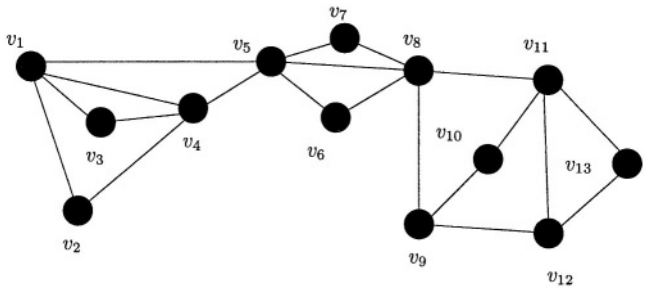


Fig. 1. $\mathcal{F}_G$ has only two non-connected nodes, corresponding to $B(v_3) = (v_1, v_2, v_4)$ and $B(v_{10}) = (v_8, v_9, v_{12}, v_{11})$. Note that $J_1 = L_1 \setminus B_1 = \{v_5, v_6, v_7, v_{13}\}$ and $J_2 = L_2 \setminus B_2 = \{v_3, v_{10}\}$. If there was another edge between v_2 and v_9, then $B(v_3) = B(v_4) = B(v_6) = (v_1, v_2, v_9, v_8, v_5, v_1)$, and hence $B(v_3)$ and $B(v_{10})$ would share the edge $\{v_8, v_9\}$.

4 Algorithms for PLANAR DOMINATING SET with Sub-linear Exponents

The following proposition is well-known and crucial to the combinatorial counting argument basic to all ways for obtaining parameterized algorithms with sub-linear exponents for PLANAR DOMINATING SET.

Proposition 1. *A plane graph with a dominating set of size of at most k is r-outerplanar with $r \leq 3k$.*

From an algorithmic point of view, the following observation might be even more useful, although it delivers the same worst-case bound in the case of simple r-outerplanar graphs.

Lemma 1. *A plane graph G with a dominating set of size of at most k has a ring decomposition $\mathcal{F}_G$ with a dominating set of size of at most k.*

Therefore, each of the algorithms discussed below will start with the prelude given in Algorithm 1. The algorithm prelude can be performed in linear time, as seen in [1,4].

Algorithm 1 Algorithmic Prelude

Input: A plane graph G and parameter $k \in \mathbb{N}$.
Compute the ring decompositon $\mathcal{F}_G$.
Compute a minumum dominating set D of $\mathcal{F}_G$.
if $|D| > k$ **then**
  answer NO
else
  continue
end if

After executing the prelude, algorithms for PLANAR DOMINATING SET proceed as follows. If G has a dominating set of size k, then Proposition 1 tells us that G is $\leq 3k$ outerplanar. So, assume that G is $\leq 3k$ outerplanar. In the following section, we will show how to obtain a sequence S—called a *layerwise separator*—of "small" separators $S_3, \ldots, S_{r-2}$, where S_i contains vertices from L_{i-1}, L_i and L_{i+1} that separate layer L_{i-1} from L_{i+2}, such that $\| S \| = |S_3| + \cdots + |S_{r-2}| \leq h(k)$ for some linear-bounded function h. Once we have such a "small" layerwise separator we can essentially split G into disconnected components, each of which is $O(\sqrt{k})$-outerplanar.

It must be the case that one of the families $F_i = \bigcup_{j=i \bmod 3, 3 \leq j \leq r-2} S_j$, $i = 1, 2, 3$ satisfies $|F_i| \leq h(k)/3$. Without loss of generality, assume that $|F_1| \leq h(k)/3$. Now, F_1 can be partitioned into $g(k)$ mutually disjoint subfamilies $F_1^m = \bigcup_{j_i = 1+m+ig(k)} S_{j_i}$, $m = 0, \ldots, g(k) - 1$, $i = 0, \ldots, \lfloor r/g(k) \rfloor$. At least one F_1^m satisfies $|F_1^m| \leq h(k)/(3g(k))$. Notice that the separator $S^m = F_1^m$ splits the graph into components that are at most $3g(k) + 3$ outerplanar.

As detailed in [1], to find the minimum dominating set in G, it suffices to search through all possible three possible domination "choices" for each vertex v in the partial layerwise separator $S^m = (S_{j_0}, \ldots, S_{j_q})$. Schematically, we can think of the "rest" of the vertices of G being "inbetween" some S_{j_i} and $S_{j_{i+1}}$. The possible domination choices are "v is in the dominating set," "v is dominated by component to the left," or "v is dominated by a vertex $u \in S_m$ or a component to the right." Since $|S^m| \leq h(k)/3g(k)$, there are $3^{h(k)/3g(k)}$ possible domination choices for S^m. Using a setting of domination choices for S^m, it is possible to split the graph into $\leq 3g(k)+5$-outerplanar components that are precolored with the colors {free, already dominated, not dominated}. (These components may contain contain vertices from S^m.) Making use of the well-known fact (shown by Bodlaender in 1989) that an r-outerplanar graph has treewidth at most $3r - 1$, we can solve the "precolored" DOMINATING SET problem on the remaining graph components in time $O(4^{9g(k)})$ time. Kanj and Perkovič [11]

sketched a dynamic programming algorithm for precolored DOMINATING SET on r-outerplanar graphs running in time $O(27^r)$. This allows us to compute the size of the minimum dominating set in time $O(3^{9g(k)})$ for each precolored component. Using the notation introduced so far, we arrive at the following result.

Lemma 2. PLANAR k-DOMINATING SET *can be solved in time*

$$O(h(k)/g(k)3^{h(k)/(3g(k))+9g(k)} + n^3).$$

More details are described in Algorithm 2.

Algorithm 2 A Generic Subexponential Algorithm for Planar DS

Input: A plane graph G and parameter $k \in \mathbb{N}$
Perform Algorithm 1 and kernelization (see [4])
Try to find a "small" partial layerwise separator S^m with $|S^m| \leq h(k)/(3g(k))$.
if such a small separator cannot be found **then**
  answer NO
end if
for all $3^{h(k)/3g(k)}$ settings of domination choices for the vertices in S^m **do**
  for each precolored graph chunk of G **do**
    Compute a minimal dominating set for the precolored graph chunk
  end for
  if minimum dominating set of G using this choice is $\leq k$ **then**
    answer YES
  end if
end for
answer NO

Since $h(k)$, as determined in the next section, will be bounded by dk for some d, $g(k)$ would then be best chosen as $c\sqrt{k}$ such that $dk/(3c\sqrt{k}) = 9c\sqrt{k}$, i.e., $c = \sqrt{d}/(3\sqrt{3})$, because $3^{dk/(3c\sqrt{k})}$ is the asymptotics of the time spent to test all possible separator settings, and $3^{3*3c\sqrt{k}}$ is the time to solve the problem on the remaining graph chunks. Hence, $9c\sqrt{k} = 9\sqrt{dk}/(3\sqrt{3}) = \sqrt{3dk}$. This gives us an $O(\sqrt{k}3^{2\sqrt{3dk}} + n^3) = O(\sqrt{k}2^{2\log_2(3)\sqrt{3dk}} + n^3)$ algorithm.

For convenience, we list a table for different values of $f(d) = 2\log_2(3)\sqrt{3dk}$ rounded up, which we use in the following sections.

$d =$	9	15	16	20	39	45	51
$f(d) =$	16.48	21.27	21.97	24.56	34.29	36.84	39.21

These numbers correspond to the constant factor in the exponent of the running times for different estimates of $h(k)$.

5 The Strategy of How to Obtain Separators

Here we assume that knowledge of the minimum dominating set D of a given plane graph G.

5.1 Simple r-Outerplanar Graphs

Let us first consider the case when G is a simple r-outerplanar graph.

Our aim is to obtain a set of vertices S_i from layers L_{i-1}, L_i and L_{i+1} that separate the vertices of L_{i-1} from the vertices of L_{i+2}. In a fashion similar to [1], we achieve this by deleting "upper", "lower", and "middle" triples of vertices in $D \cap B_i$ and $D \cap J_i$. Let $b_i = |D \cap B_i|$ and $j_i = |D \cap J_i|$.

Lemma 3. *There exists a set S_i that separates L_{i-1} from L_{i+2} of size $|S_i| \leq 3b_{i-1} + 3j_i + 3b_i + 3j_{i+1} + 3b_{i+1}$.*

Since $\sum_{i=3}^{r-2} |S_i| \leq 9k = h(k)$, for simple r-outerplanar graphs, we get an $2^{16.4715\sqrt{k}}$-algorithm. This is appears to be a lower bound on the running-time that we can obtain by the current approach.

5.2 The General Case

Let us try to transfer our reasoning to the general case.

As **a first preparatory step** for layer L_i, let us think about how much it would "cost" to actually separate each $B(i,j)$ from the other $B(i,j')$ such that we can be sure that any (remaining) dominating set vertex on $B(i,j)$ or in the interior of $B(i,j)$ only dominates vertices on $B(i,j)$ or in the interior of $B(i,j)$ or from J_i. Furthermore, "indirect" influences of different rings by interconnection patterns will be ruled out. Moreover, this reasoning will imply that later on we can safely speak about dominating vertices "private" to $B(i,j)$. The vertices "taken out" in this preparatory step will be put into the separator we are going to construct.

For the general case, we will need to sufficiently separate the boundary cycles in each layer so that we can employ the previous arguments for simple r-outerplanar graphs. In particular, we say that a set S^i *properly separates* L_i according to D if each $x \in (D \cap L_i) - S^i$ dominates only vertices in one $B(i,j) - S^i$ for some $1 \leq j \leq c_i$. The following results are crucial to our construction of S^i.

Lemma 4. *If $x \in L_i \setminus B(i,j)$ and P_1 and P_2 are two paths leading from x to some points z_1 and z_2 on $B(i,j)$, using only vertices from J_i on the way, then either $z_1 = z_2$ or z_1 and z_2 are neighbours.*

Lemma 5. *It suffices to take out at most four vertices to separate two linked rings of the same layer.*

As we show next, we can construct such an S^i by applying Lemma 3 and 4 using only $4c_i$ vertices. The proof and construction are contained in the appendix.

Lemma 6. *There exists an S^i with $|S^i| \leq 4c_i$ that properly separates L_i according to any dominating D.*

We now describe a **second preparatory step** for layer L_i, sketched in the following lemmma, which will simplify the "triple" arguments in what follows.

Lemma 7. *Having performed the first preparatory step for layer L_i, there are at most $2c_i$ vertices on $B_{i-1} \cup B_i$ whose removal leaves a graph such that any path from any $x \in B_{i-1}$ downto B_i must get to some ring $B(i,j)$ specific to x, i.e., there is a partial mapping $x \mapsto j$.*

The proof of Lemma 7 is omitted; however, we give some intuition concerning the approach. By removing S^i from G, we partition L_i into regions that can only reach a single $B(i,j)$. The planarity of G forces an ordering of these regions w.r.t. B_{i-1}. Hence, it is possible to delete at most 2 shared fringe vertices from B_{i-1} per $B(i,j)$ to force all paths starting at $x \in B_{i-1}$ to go to $B(i,j)$.

We now assume that we have performed both preparatory steps *for layers L_i and L_{i+1}*. This way, we take out at most $6c_i + 6c_{i+1}$ vertices.

After these preparatory steps, the graph will look "locally" like a simple r-outerplanar graph. Therefore, the analysis (as detailed in the appendix) of that previously considered case basically transfers. Altogether, $|S_i| \leq 3b_{i-1} + 3j_i + 3b_i + 3d_{i+1} + 6c_i + 6c_{i+1}$.

Theorem 1. *If G is a plane graph with a dominating set of size at most k, then the overall sum of the sizes of the layerwise separators can be upperbounded by $h(k) = 9k + 12c(k)$, where $c(k)$ upperbounds $\sum_{i=1}^{r} c_i$.*

In [1] $c(k) \leq 3k$; however, this bound may not hold for ring decompostions. Nevertheless, by adapting some trade-off computations from [11], we can significantly improve on the previously known bound.

6 A Trade-Off Computation

Kanj and Perkovič made the following observation: There is no need to cut off branches in $\mathcal{T}'_G$ starting in a node n whose distance $\delta(n)$ to its farest descendant is only $\delta(n) \leq D(k) = O(\sqrt{k})$, since, in the worst case, only $D(k)$ layers would be added to each graph chunk.

Using some simplified arguments, we show that their computations carry over to our new analysis, as well. Actually, the above analysis was "tuned" in order to exploit their technique best possible.

For a given bounding function D, call a node n in $\mathcal{T}'_G$ *deep, medium,* and *shallow,* respectively, if $\delta(n) > D(k)$, $\delta(n) = D(k)$, or $\delta(n) < D(k)$, respectively.

Let μ denote the number of medium nodes in $\mathcal{T}'_G$. Any medium node n contains $D(k)$ layers in $\mathrm{int}(n)$ and therefore at least $D(k)/3$ vertices from the dominating set D. Given medium nodes n and n' in $\mathcal{T}_G$ (and hence rings in G), then $\mathrm{int}(n) \cap \mathrm{int}(n') = \varnothing$. If $\mu > 3k/D(k)$, then the number of dominating set vertices in $\bigcup\{\mathrm{int}(n) : n \text{ is medium}\}$ would exceed k. Hence, $\mu \leq 3k/D(k)$.

A node is called *m-complex* if it has at least m non-shallow children. Considering the number of inner nodes in m-ary trees with μ leaves, it is readily seen that there are no more than $\lfloor(\mu-1)/(m-1)\rfloor$ many m-complex nodes. Denoting the number of non-shallow children of of n by $c(n)$, it follows that the sum of all $c(n)$, taken over complex nodes, is no larger than $\lfloor m(\mu-1)/(m-1)\rfloor$.

In the following, let us specialize towards $m = 2$.[1] Terming a non-2-complex node *simple* generalizes of aforementioned notion of simple r-outerplanar graphs. Our reasoning entails:

Lemma 8. $\sum\{c(n) \mid n \text{ is complex}\} \leq 2\mu \leq 6k/D(k)$.

Let us now reconsider the considerations of the previous section under the observation that we only try to separate deep nodes (rings). This leads immediately to our main theorem.

Theorem 2. *The* PLANAR DOMINATING SET *problem can be solved in time*

$$O(\sqrt{k}3^{6\sqrt{3k}+o(\sqrt{k})} + n^3) \approx O(\sqrt{k}2^{6\log_2(3)\sqrt{3k}} + n^3) \approx O(2^{16.4715\sqrt{k}} + n^3).$$

We observe that similar improvements can be obtained for many domination-type problems mentioned in [1]. Our results can also be used to improve the constants derived in [6] for the DOMINATING SET PROBLEM on $K_{3,3}$-minor-free or K_5-minor-free graphs by tightening the bound on the treewidth of planar graphs with dominating sets of size k. .

Theorem 3. *If G is a planar graph which has a dominating set of size k, then G has treewidth of at most $6\sqrt{3k} + o(\sqrt{k}) = O(10.40\sqrt{k})$.*

This largely improves the upperbound $6\sqrt{34k}$ proved in [1] and also nearly matches the treewidth bound $9.55 \approx (4.5)^{1.5}$ derived in [9] via completely different arguments. Combining this treewidth bound with Theorem 3 in [6], we obtain the following.

Corollary 1. *If an n-vertex graph G is $K_{3,3}$-minor-free or K_5-minor-free, then in time $O(4^{9.55\sqrt{k}}n + n^2)$ it can be determined whether G has a dominating set of size at most k.*

The strategy outlined in the last two sections is not only applicable to dominating set, but rather to any planar graph problem where the sum of the sizes of the layerwise separators can be estimated as a linear function in k and in the number of rings. For many problems, we can really restrict our attention to the case of simple r-outerplanar graphs, since the estimates involving the number of rings and the optimization computations do not rely on the structure of the concrete problem in any way. We exemplify this approach in the next section.

7 PLANAR RED/BLUE DOMINATING SET and Related Problems

The techniques outlined in the previous sections apply to other problems on planar graphs. For PLANAR RED/BLUE DOMINATING SET, it suffices to build an appropriate separator S_i to separate layers L_{i-1} from layer L_{i+3}.

[1] In [11], the case $m = 3$ was considered instead.

Lemma 9. *There exists a separator S_i that separates layer L_{i-1} from layer L_{i+3} in G of size*

$$|S_i| \leq 3b_{i-1} + 3j_i + 7b_i + 4j_{i+1} + 5b_{i+1} + 5j_{i+2} + 5b_{i+2} + 6c_i + 6c_{i+1} + 6c_{i+2}.$$

The construction and proof can be found in the appendix.

Adapting the notion of "partial layerwise separator" to allow for considering only every fourth layer, there exist a partial layerwise separator S with $||S|| \leq h(k) = 20k+18c(k)$. Since the trade-off computations only involve combinatorics on ring decomposition trees, they readily carry over to this case, basically making the term $c(k)$ disappear when it comes to the running time of algorithms.

Taking either the $27^l n$-algorithm (l denoting the number of layers of the graph, i.e., its outerplanarity) as sketched in [11] or the 3^{tw}-algorithm (*tw* denotes the treewidth of the graph; observe that $tw < 3out$) exhibited in [1] and finding a c such that the overall size of the partial layerwise separators $20k/(4c\sqrt{k})$ matches the treewidth of the remaining graph chunks $12c\sqrt{k}$, i.e., $c = \frac{\sqrt{5}}{\sqrt{12}} = \frac{\sqrt{5}}{2\sqrt{3}}$. This gives the following result.

Theorem 4. *There exists a $O(3^{2\cdot 10\cdot \frac{\sqrt{3}}{\sqrt{5}}+o(\sqrt{k})}n) = O(2^{24.551\sqrt{k}}n)$ algorithm for* PLANAR RED/BLUE DOMINATING SET. *Moreover, a planar graph which has a k-red/blue dominating set has treewidth of at most $20\frac{\sqrt{3}}{\sqrt{5}} + o(\sqrt{k}) \approx 15.5\sqrt{k}$.*

Theorem 4 has several important applications, as we show next.

As observed by Cai, Fellows, Juedes, and Rosamond [5], results concerning PLANAR RED/BLUE DOMINATING SET can be used to show that each problem in a subset of the class Planar TMIN has a $c^{\sqrt{k}}$ parameterized algorithm. Planar TMIN is a syntactic class that was defined by Khanna and Motwani [12], along with two other classes, to characterize those problem that admit PTAS. The precise definition of Planar TMIN and it subclasses, defined in [5], require a brief explanation.

Given a collection of variables X, a *minterm* is simply a conjunction (AND) of literals over X. A literal l is *negative* if it is the negation of some variable $x_i \in X$, i.e., $l = \neg x_i$. Otherwise, a literal is *positive*. A minterm m is positive if all of the literals in m are positive. Likewise, a minterm m is negative if all of the literals in m are negative. A first order formula (FOF) is a disjunction (OR) of minterms. A FOF is positive if all of its minterms are positive. Similarly, A FOF is negative if all of its minterms are negative. The *width* of a FOF is the number of minterms in the formula. The *size* of a minterm is the number of literals in the minterm.

The class TMIN [12] is the class of all NP-optimization problems that can be written (rewritten) as follows. Given a collection C of positive FOFs over n variables, find a minimum weighted truth assignment T that satisfies all FOFs in C. Given a collection C of FOFs over n variables X, the *incidence graph* of C is the bipartite graph G_C with edges between FOFs in C and the set of variables X such that there is an edge between a formula and a variable if and only the

variable appears in a minterm in the formula. The class Planar TMIN is the class TMIN restricted to problems with planar incidence graphs.

As shown in [5], there exist problems in Planar TMIN that are W[1]-hard, and hence not all problems in Planar TMIN have parameterized tractable algorithms unless $FPT = W[1]$. In contrast, we show here that all the problems in subclass of Planar TMIN, known as Planar TMIN_1, have fast parameterized algorithms.

Planar TMIN_1 is the subclass of Planar TMIN where all minterms are restricted to be of size 1. For example, PLANAR VERTEX COVER is in Planar TMIN_1 since the minimum vertex cover in a planar graph G can be described as the minimum weighted truth assignment satisfying $\bigwedge_{(u,v)\in E} (x_u \vee x_v)$. Each term $x_u \vee x_v$ is FOF, and the the incidence graph of this collection of FOFs is planar.

Theorem 4 leads immediately to the following result.

Theorem 5. *([5]) Each problem Π in* PLANAR TMIN_1 *can be solved in time $O(2^{O(\sqrt{k})}g(n))$ for some polynomial $g(n)$.*

The main application of this general result is that faster algorithms for Red/Blue Dominating Set can be used to achieve faster algorithms for a variety of problems in Planar TMIN_1. In particular, this leads to the fastest-known parameterized algorithms for FACE COVER.

Corollary 2. *The* FACE COVER *problem can be solved in time $O(2^{24.551\sqrt{k}}n)$.*

This largely improves the best-known algorithm running in time $O(3^{36\sqrt{34k}}n)$ [1].

Corollary 3. *The* EDGE DOMINATING SET ON PLANAR GRAPHS *problem can be solved in time $O(2^{24.551\sqrt{k}}n)$.*

8 Perspectives

Here we show another approach to further lower the huge constants limiting the worst-case behaviour of parameterized algorithms with sublinear exponents for PLANAR DOMINATING SET and related problems. We feel that there may be room for further improvements with this approach. Firstly, a more involved geometric argument may further reduce the vertices needed in the case of simple r-outerplanar graphs (which then probably generalizes). Secondly, trade-off computations others than the ones envisaged by Kanj and Perkovič might apply. Finally, the tree-decomposition based algorithms might be improved by lowering down the upperbound of $3r$ for r-outerplanar graphs by simplifying the graph beforehand using appropriate reduction rules to remove most of the interconnection patterns. Interestingly, the trade-off trick from [11] does not apply to problems like PLANAR VERTEX COVER, since the number of components are irrelevant there (so far). Nonetheless, improvements for the constants in this problem should be possible, especially by using the last technique. Alternatively, further research on geometric approaches using branchwidth instead of using treewidth and outerplanarity may be promising. Finally, one might ask whether both approaches could be combined.

References

1. J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for DOMINATING SET and related problems on planar graphs. *Algorithmica,* 33:461–493, 2002.
2. J. Alber, H. L. Bodlaender, H. Fernau, and R. Niedermeier. Fixed-parameterized algorithms for planar dominating set and related problems. In *Proceedings of the 7th Scandinavian Workshop on Algorithm Theory SWAT,* volume 1851 of *LNCS,* pages 97–110. Springer, 2000.
3. J. Alber, H. Fan, M. Fellows, H. Fernau, and R. Niedermeier. Refined search tree technique for dominating set on planar graphs. In *Proceedings of the 25th International Symposium on the Mathematical Foundations of Computer Science MFCS,* volume 2136 of *LNCS,* pages 111–122. Springer, 2000.
4. J. Alber, M. R. Fellows, and R. Niedermeier. Efficient data reduction for dominating set: A linear problem kernel for the planar case. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory SWAT,* volume 2368 of *LNCS,* pages 150–159. Springer, 2002.
5. L. Cai, M. R. Fellows, D. Juedes, and F. Rosamond. On efficient polynomial-time approximation schemes for problems on planar structures, 2001. Manuscript. Some of these results appear in [8].
6. E. D. Demaine, M. T. Hajiaghayi, and D. Thilikos. Exponential speedup of fixed parameter algorithms on $K_{3,3}$-minor-free or K_5-minor-free graphs. In *Proceedings of the 13th Annual International Symposium on Algorithms and Computation ISAAC,* volume 2518 of *LNCS,* pages 262–273. Springer, 2002.
7. R. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.
8. M. R. Fellows. Parameterized complexity: the main ideas, connections to heuristics and research frontiers. In *Proceedings of the 12th International Symposium on Algorithms and Computation ISAAC,* volume 2223 of *LNCS,* pages 291–307. Springer, 2001.
9. F. V. Fomin and D. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. In *Proceedings of the 14th Annual Symposium on Discrete Algorithms SODA,* pages 168–177. ACM Press, 2003.
10. M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness.* W.H. Freeman, 1979.
11. I. A. Kanj and L. Perkovič. Improved parameterized algorithms for planar dominating set. In *Proceedings of the 27th International Symposium on the Mathematical Foundations of Computer Science MFCS,* volume 2420 of *LNCS,* pages 399–410. Springer, 2002.
12. S. Khanna and R. Motwani. Toward a syntactic characterization of PTAS. In *Proceedings of the 28th Annual Symposium on Theory of Computing STOC,* pages 329–337. ACM Press, 1996.
13. T. Kloks, C. M. Lee, and J. Liu. New algorithms for k-face cover, k-feedback vertex set, and k-disjoint cycles on plane and planar graphs. In *Proceedings of the 28th Workshop on Graph-Theoretic Concepts in Computer Science WG,* volume 2573 of *LNCS,* pages 282–295. Springer, 2002.
14. M. Thorup. Structural programs have small tree-width and good register allocation. In *Proceedings of the 23rd Workshop on Graph-Theoretic Concepts in Computer Science WG,* volume 1335 of *LNCS,* pages 318–332. Springer, 1997.

Polynomial Time Approximation Schemes and Parameterized Complexity

Jianer Chen[1], Xiuzhen Huang[1], Iyad A. Kanj[2], and Ge Xia[1]

[1] Department of Computer Science, Texas A&M University, College Station, TX 77843-3112.
{chen,xzhuang,gexia}@cs.tamu.edu***

[2] School of Computer Science, Telecommunications and Information Systems, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604-2301.
ikanj@cs.depaul.edu†

Abstract. In this paper, we study the relationship between the approximability and the parameterized complexity of NP optimization problems. We introduce the notion of *efficient fixed-parameter tractability* and prove that, under a very general constraint, an NP optimization problem has a fully polynomial time approximation scheme *if and only if* the problem is efficiently fixed-parameter tractable. By enforcing a constraint of planarity on the *W*-hierarchy studied in parameterized complexity theory, we obtain a class of NP optimization problems, the *planar W-hierarchy,* and prove that all problems in this class have efficient polynomial time approximation schemes (EPTAS).

1 Introduction

According to the NP-completeness theory [15], many optimization problems of theoretical interest and practical importance are NP-hard, thus cannot be solved optimally in polynomial time unless $P = NP$. Many approaches have been proposed to cope with the NP-hardness of such problems. The most celebrated among these approaches is polynomial time approximation, which involves compromising an optimal solution for a "good" solution that is computable in polynomial time.

A notable class of NP-hard optimization problems has *fully polynomial time approximation schemes* (FPTAS). An FPTAS algorithm is an efficient approximation algorithm whose approximation ratio is bounded by $1 + \epsilon$ and whose running time is bounded by a polynomial in both the input size and $1/\epsilon$, where the relative error bound ϵ can be any positive real number. Examples of FPTAS problems include the well-known KNAPSACK problem and the MAKESPAN problem on a fixed number of processors [16]. A more general class of NP-hard optimization problems admits *polynomial time approximation schemes* (PTAS),

*** This research was supported in part by the NSF under grants CCR-0000206 and CCR-0311590.

† This work was supported in part by DePaul University Competitive Research Grant.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 500–512, 2004.
© Springer-Verlag Berlin Heidelberg 2004

which have polynomial time approximation algorithms of approximation ratio $1 + \epsilon$ for each fixed relative error bound $\epsilon > 0$. A large number of NP-hard optimization problems belong to the class PTAS [16], including, based on the recent research, the well-known EUCLIDEAN TRAVELING SALESMAN problem [4] and the GENERAL MULTIPROCESSOR JOB SCHEDULING problem [10].

Contrary to the efficiency of FPTAS algorithms, the running time of a general PTAS algorithm of approximation ratio $1+\epsilon$ can be of the form $O(n^{t(\epsilon)})$, where n is the input size and $t(\epsilon)$ is a function of ϵ that can be very large even for moderate values of ϵ. Downey [11] (see also Fellows [13]) examined many recently developed PTAS algorithms for NP-hard optimization problems, and discovered that for the relative error bound value of $\epsilon = 20\%$, most of these PTAS algorithms have $t(\epsilon) > 10^6$, i.e., the running time of these PTAS algorithms exceeds the order of n^{100000}! Obviously, these PTAS algorithms are not practically feasible.

Observing this fact, recent research has proposed to further refine the class PTAS. We say that an optimization problem Q has an *efficient polynomial time approximation scheme* (EPTAS) if for any $\epsilon > 0$, there is an approximation algorithm of ratio $1 + \epsilon$ for Q whose running time is bounded by a polynomial of the input size whose degree is independent of ϵ. In particular, all FPTAS problems belong to the class EPTAS.

Clearly, EPTAS algorithms are superior to PTAS algorithms whose running time is of the form $O(n^{t(\epsilon)})$ in terms of their efficiency. In fact, many PTAS algorithms developed for NP-hard optimization problems are actually EPTAS algorithms. Moreover, there are a number of well-known NP-hard optimization problems, such as the EUCLIDEAN TRAVELING SALESMAN problem [4], the GENERAL MULTIPROCESSOR JOB SCHEDULING problem [10], and the MAKESPAN problem on unbounded number of processors [16], for which early developed PTAS algorithms had running time of the form $O(n^{t(\epsilon)})$, but later were improved to EPTAS algorithms. On the other hand, very recently Cai et al.[7] studied the syntactic characterizations of PTAS problems proposed by Khanna and Motwani [18], and showed strong evidence that there are PTAS problems that have no EPTAS.

Therefore, it is interesting to study the characterization of EPTAS problems. The current paper attempts to characterize EPTAS (as well as FPTAS) problems using the parameterized complexity framework. Parameterized complexity theory [12] is a recently proposed approach dealing with NP optimization problems, which studies the computational complexity of optimization problems in terms of both the instance size and a properly selected parameter. The class of *fixed parameter tractable problems* has been introduced to characterize such parameterized problems that become feasible for small parameter values. On the other hand, a hierarchy, the *W-hierarchy* $\bigcup_{i\geq 1} W[i]$, has been introduced to capture the fixed parameter intractability of optimization problems. Parameterized complexity theory has drawn considerable attention recently because of its applications in developing practical algorithms and in deriving computational lower bounds for NP optimization problems (see the surveys [11,13] for recent progress in the area).

We start by identifying a subclass, *efficient-FPT,* of fixed parameter tractable problems, and prove that under a very general condition (the *scalability* condition, see section 3 for a formal definition), a problem is in FPTAS if and only if it is in efficient-FPT. This provides an almost[1] very precise characterization of the approximation class FPTAS in terms of parameterized complexity.

We then study the characterization of the class EPTAS. We enforce a constraint of planarity on the *W*-hierarchy in parameterized complexity theory, and introduce the syntactic classes PLANAR MIN-$W[h]$, PLANAR MAX-$W[h]$, and PLANAR $W[h]$-SAT (this approach is similar to that of Khanna and Motwani [18] in their efforts to characterize the class PTAS). These syntactic classes capture many NP optimization problems in the class EPTAS, such as PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, and PLANAR MAX-SAT. By extending Baker's techniques [5] and techniques more recently developed in the study of parameterized algorithms [1,14], we prove that all problems in these syntactic classes belong to the class EPTAS. These syntactic classes seem to form the core of a significant class of EPTAS problems. Finally, we point out that our syntactic classes are significantly different from the PTAS syntactic classes introduced by Khanna and Motwani [18]: our syntactic classes characterize only EPTAS problems while the syntactic classes in [18] include PTAS problems that are not in EPTAS [7], while on the other hand, our syntactic classes contain EPTAS problems that cannot be characterized by the syntactic classes in [18].

Our results combined with a result by Cesati and Trevisan [8] show that all problems expressible by our syntactic classes are fixed-parameter tractable. Moreover, a byproduct derived from an immediate result in our discussion shows that for any fixed integer $t \geq 0$, the PLANAR t-NORMALIZED WEIGHTED SATISFIABILITY problem is solvable in polynomial time, which answers an open problem posed by Downey and Fellows [12].

2 Preliminaries and Further Definitions

We assume certain familiarity with the background in parameterized computation and in computational optimization. For detailed discussions on these topics, the readers are referred to [2,12,15].

A *parameterized problem* Q is a subset of $\Omega^* \times N$, where Ω is a fixed alphabet and N is the set of all non-negative integers. Therefore, each instance of the parameterized problem Q is a pair (x, k), where the second component, i.e., the non-negative integer k, is called the *parameter.* We say that the parameterized problem Q is *fixed-parameter tractable* [12] if there is a (parameterized) algorithm that decides whether an input (x, k) is a member of Q in time $f(k)|x|^c$, where c is a fixed constant and $f(k)$ is a recursive function independent of the input length $|x|$. Let FPT denote the class of all fixed parameter tractable problems.

An NP optimization problem Q has a *polynomial time approximation scheme* (PTAS) if there is an algorithm A_Q that takes a pair (x, ϵ) as input, where x is

[1] The scalability property is needed.

an instance of Q and $\epsilon > 0$ is a real number, and returns a feasible solution y for x such that the approximation ratio of the solution y is bounded by $1 + \epsilon$, and for each fixed $\epsilon > 0$, the running time of the algorithm A_Q is bounded by a polynomial of $|x|$. An NP optimization problem Q has a *fully polynomial time approximation scheme* (FPTAS) if it has a PTAS A_Q such that the running time of A_Q is bounded by a polynomial of $|x|$ and $1/\epsilon$.

Observe that the time complexity of a PTAS algorithm may be of the form $O(2^{1/\epsilon}|x|^c)$ for a fixed constant c or of the form $O(|x|^{1/\epsilon})$. Obviously, the latter type of computations with small ϵ values will turn out to be practically infeasible. This leads to the following definition [8].

Definition 1. An NP optimization problem Q has an *efficient polynomial time approximation scheme* (EPTAS) if it admits a polynomial-time approximation scheme whose time complexity is bounded by $f(1/\epsilon)|x|^c$, where f is a recursive function and c is a constant.

An NP optimization problem Q can be parameterized in a natural way as follows.

Definition 2. Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem. The *parameterized version* of Q is defined as follows:

(1) $Q_{\geq} = \{(x, k) \mid x \in I_Q \wedge opt_Q(x) \geq k\}$, if Q is a maximization problem;
(2) $Q_{\leq} = \{(x, k) \mid x \in I_Q \wedge opt_Q(x) \leq k\}$, if Q is a minimization problem.

The above definition offers the possibility to study the relationship between the approximability and the parameterized complexity of NP optimization problems. However, there is an essential difference between the two categories: an approximation algorithm for an NP optimization problem constructs a solution for a given instance of the problem, while a parameterized algorithm only provides a "yes/no" decision on an input. To make the comparison meaningful, we need to extend the definition of parameterized algorithms in a natural way so that when a parameterized algorithm returns a "yes" decision, it also provides an "evidence" to support the conclusion (see [6] for a similar treatment).

Definition 3. Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem. We say that *a parameterized algorithm A_Q solves the parameterized version of Q* if

(1) in case Q is a maximization problem, then on an input pair (x, k) in $Q_{\geq}$, the algorithm A_Q returns "yes" with a solution y in $S_Q(x)$ such that $f_Q(x, y) \geq k$, and on any input not in $Q_{\geq}$, the algorithm A_Q simply returns "no";

(2) in case Q is a minimization problem, then on an input pair (x, k) in $Q_{\leq}$, the algorithm A_Q returns "yes" with a solution y in $S_Q(x)$ such that $f_Q(x, y) \leq k$, and on any input not in $Q_{\leq}$, the algorithm A_Q simply returns "no".

3 FPTAS and Efficient-FPT

In this section, we present a characterization for the approximation class FPTAS in terms of parameterized complexity. Recall that a fixed-parameter tractable

problem has an algorithm of running time of the form $f(k)n^c$, where f is an arbitrary recursive function. By enforcing a further constraint on the function $f(k)$, we introduce the following subclass of the class FPT:

Definition 4. An NP optimization problem Q is *efficiently fixed-parameter tractable* (efficient-FPT) if its parameterized version is solvable by a parameterized algorithm of running time bounded by a polynomial of $|x|$ and k.

Note that efficient-FPT does not necessarily imply polynomial time computability: NP optimization problems, in particular a large variety of scheduling problems, may have their optimal values much larger than the input size. In consequence, the parameterized versions of these problems may have their parameter values k much larger than the input size.

Definition 5. An optimization problem $Q = (I_Q, S_Q, f_Q, opt_Q)$ is said to be *scalable* if there are polynomial time computable functions g_1 and g_2 and a fixed polynomial q such that:

1. for any instance $x \in I_Q$, and any integer $d \geq 1$, $x_d = g_1(x, d)$ is an instance of Q such that: $|x_d| \leq q(|x|)$ and $|opt_Q(x_d) - opt_Q(x)/d| \leq q(|x|)$; and
2. for any solution y_d to the instance x_d, $y = g_2(x_d, y_d)$ is a solution to the instance x such that: $|f_Q(x_d, y_d) - f_Q(x, y)/d| \leq q(|x|)$.

Most NP optimization problems are scalable. In particular, if an NP optimization problem Q has its optimal value $opt(x)$ bounded by a polynomial of $|x|$ for all instances x, then the problem Q is automatically scalable — simply let $x_d = g_1(x, d) = x$ for any integer d, and for a solution y_d to $x_d = x$, let $g_2(x_d, y_d) = y_d$. This immediately implies that most set problems and graph problems are scalable, including the well-known NP-hard problems such as BIN PACKING, 3D-MATCHING, SET COVER, VERTEX COVER, and DOMINATING SET. Moreover, most NP optimization problems involving large numbers (i.e., the *number problems* defined by Garey and Johnson [15]), such as KNAPSACK and MAKESPAN, are also scalable. We pick Q = MAKESPAN as an example to illustrate how such a problem involving large numbers can be scaled. An instance x of MAKESPAN consists of n jobs of integral processing times $t_1, t_2, \ldots, t_n$, respectively (we will refer to the jth job by t_j), and an integer m, the number of identical processors, and asks to construct a scheduling of the jobs on the m processors so that the completion time (i.e., the *makespan*) is minimized. For a given instance $x = (t_1, t_2, \ldots, t_n; m)$ of MAKESPAN and a given integer $d \geq 0$, we define

$$x_d = g_1(x, d) = (t'_1, t'_2, \ldots, t'_n; m),$$

where $t'_i = \lceil t_i/d \rceil$ for $i = 1, 2, \ldots, n$, which is also an instance for MAKESPAN. A solution y_d to the instance x_d is a scheduling that partitions the n jobs in x_d into m subsets: $y_d = (T'_1, \ldots, T'_m)$, where T'_i is the set of jobs in x_d that are assigned to the ith processor. We define $y = g_2(x_d, y_d)$ to be the same index partitioning of the jobs in x: $y = (T_1, \ldots, T_m)$ (i.e., a job t_j is in T_i if and only if the job t'_j is in T'_i). Obviously, $y = g_2(x_d, y_d)$ is a solution for the instance x, and the functions

g_1 and g_2 are computable in polynomial time. To see the relation between the solution $y = (T_1, \ldots, T_m)$ for x and the solution $y_d = (T'_1, \ldots, T'_m)$ for x_d, note that the makespan of y is equal to $\max_i\{\sum_{t_j \in T_i} t_j\}$, and the makespan of y_d is equal to $\max_i\{\sum_{t'_j \in T'_i} t'_j\}$. We have

$$f_Q(x_d, y_d) = \max_i\{\sum_{t'_j \in T'_i} t'_j\} = \max_i\{\sum_{t_j \in T_i} \lceil t_j/d \rceil\}$$
$$\geq \max_i\{\sum_{t_j \in T_i} t_j/d\} = \max_i\{\sum_{t_j \in T_i} t_j\}/d = f_Q(x, y)/d. \quad (1)$$

On the other hand

$$f_Q(x_d, y_d) = \max_i\{\sum_{t'_j \in T'_i} t'_j\} = \max_i\{\sum_{t_j \in T_i} \lceil t_j/d \rceil\}$$
$$\leq \max_i\{\sum_{t_j \in T_i} (t_j/d + 1)\} \leq \max_i\{\sum_{t_j \in T_i} t_j\}/d + n = f_Q(x, y)/d + n. \quad (2)$$

Here we have used the fact that the total number of jobs in each subset T_i is bounded by n. Combining (1) and (2), we get $|f_Q(x_d, y_d) - f_Q(x, y)/d| \leq n$. Similarly, it can be verified that the instances x and x_d satisfy $|opt_Q(x_d) - opt_Q(x)/d| \leq n$. In conclusion, the MAKESPAN problem is scalable.

Theorem 1. *Let $Q = \langle I_Q, S_Q, f_Q, opt_Q \rangle$ be a scalable NP optimization problem. Then Q has an FPTAS if and only if Q is efficient-FPT.*

Proof. One direction of the theorem was implicitly proved in [6]. Suppose that Q has an FPTAS A_Q, which is an algorithm such that on any instance x of Q and any given $\epsilon > 0$, the algorithm A_Q constructs a solution of ratio bounded by $1+\epsilon$ for x, in time $p(|x|, 1/\epsilon)$, where $p(|x|, 1/\epsilon)$ is a polynomial of $|x|$ and $1/\epsilon$. Cai and Chen proved ([6], Theorem 3.2) that then the parameterized version of Q can be solved in time $O(p(|x|, 2k))$. In consequence, the problem Q is efficient-FPT.

To show the converse, we consider specifically the case when Q is a maximization problem (a proof for minimization problems can be similarly derived). Suppose that the problem Q is efficient-FPT, and the parameterized version $Q_{\geq}$ is solvable in time $p(k, |x|)$, which is a polynomial in k and $|x|$. Since Q is scalable, we let g_1 and g_2 be the polynomial time computable functions, and q be the polynomial in the definition of the scalability of Q. For a given instance x of Q and a real number $\epsilon > 0$, consider the following algorithm (assume $n = |x|$):

1. let $x_1 = g_1(x, 1)$; if $(x_1, 3q(n)/\epsilon)$ is not in $Q_{\geq}$, then try all instances $(x, 1), (x, 2), \ldots, (x, 3q(n)/\epsilon + q(n))$ to construct an optimal solution for x; STOP.
2. use binary search on d to find an integer $d \geq 1$ such that $(x_d, 3q(n)/\epsilon)$ is in $Q_{\geq}$, but $(x_{d+1}, 3q(n)/\epsilon)$ is not in $Q_{\geq}$;
3. construct an optimal solution y_d for the instance x_d;
4. let $y_0 = g_2(x_d, y_d)$; output y_0 as a solution for x.

We discuss the correctness and the complexity of the above algorithm. First note that by the definition, $|x_d| \leq q(n)$ for any integer d. If $(x_1, 3q(n)/\epsilon)$ is not in $Q_{\geq}$, then $opt_Q(x_1) < 3q(n)/\epsilon$. Moreover, since Q is scalable, we have $|opt_Q(x_1) - opt_Q(x)/1| \leq q(n)$. Combining these two relations, we get $opt_Q(x) \leq opt_Q(x_1) + q(n) < 3q(n)/\epsilon + q(n)$. Thus, step 1 of the algorithm will correctly construct an optimal solution for the instance x (note by our definition, on input $(x, opt_Q(x))$, the parameterized algorithm must return "yes" with an optimal solution to the instance x). Moreover, since checking each instance (x, k) takes time $p(k, n)$, where $k = 1, 2, \ldots, 3q(n)/\epsilon + q(n)$, step 1 of the algorithm takes time bounded by $O((3q(n)/\epsilon + q(n))p(3q(n)/\epsilon + q(n), n))$, which is a polynomial of n and $1/\epsilon$.

If $(x_1, 3q(n)/\epsilon)$ is in $Q_{\geq}$, then we execute step 2 of the algorithm. First we need to show that there must be an integer $d \geq 1$ such that $(x_d, 3q(n)/\epsilon)$ is in $Q_{\geq}$ but $(x_{d+1}, 3q(n)/\epsilon)$ is not in $Q_{\geq}$. We already know that $(x_d, 3q(n)/\epsilon)$ is in $Q_{\geq}$ for $d = 1$. Thus, we only need to show that there must be a d such that $(x_d, 3q(n)/\epsilon)$ is not in $Q_{\geq}$. Since Q is an NP optimization problem, we have $opt_Q(x) < 2^{r(n)}$, where $r(n)$ is a polynomial in n. Therefore if we let $d = 2^{r(n)}$, then from the scalability of the problem Q, we have $|opt_Q(x_d) - opt_Q(x)/d| \leq q(n)$, which gives immediately $opt_Q(x_d) < 1 + q(n) \leq 3q(n)/\epsilon$ (here we assume without loss of generality that $q(n) \geq 1$ and $0 < \epsilon < 1$). Thus, the integer d in step 2 of the algorithm must exist and $d \leq 2^{r(n)}$. Since we use binary search on d, the total number of instances $(x_d, 3q(n)/\epsilon)$ we check in step 2 is bounded by $r(n)$. By our assumption, each instance $(x_d, 3q(n)/\epsilon)$ of $Q_{\geq}$ can be tested in time $p(3q(n)/\epsilon, q(n))$ (note that $|x_d| \leq q(n)$). Therefore, the running time of step 2 of the algorithm is also bounded by a polynomial of n and $1/\epsilon$.

Now consider step 3. Since $(x_{d+1}, 3q(n)/\epsilon)$ is not in $Q_{\geq}$, we have $opt_Q(x_{d+1}) < 3q(n)/\epsilon$. By the scalability of Q, we have

$$|opt_Q(x_d) - opt_Q(x)/d| \leq q(n) \quad \text{and} \quad |opt_Q(x_{d+1}) - opt_Q(x)/(d+1)| \leq q(n).$$

From this we get (note since $d \geq 1$, we have $(d+1)/d \leq 2$)

$$\begin{aligned} opt_Q(x_d) &\leq \frac{opt_Q(x)}{d} + q(n) = \frac{d+1}{d} \cdot \frac{opt_Q(x)}{d+1} + q(n) \\ &\leq \frac{d+1}{d}(opt_Q(x_{d+1}) + q(n)) + q(n) \\ &\leq 2 \cdot opt_Q(x_{d+1}) + 3q(n) \leq \frac{6q(n)}{\epsilon} + 3q(n). \end{aligned}$$

Thus, by checking all instances (x_d, k), where $k = 1, 2, \ldots, 6q(n)/\epsilon + 3q(n)$, each taking time $p(k, q(n))$, we will be able to construct the optimal solution y_d for the instance x_d. In conclusion, step 3 of the algorithm also takes time polynomial in n and $1/\epsilon$.

Summarizing the above discussion, we conclude that the running time of the algorithm is bounded by a polynomial in n and $1/\epsilon$. What remains is to bound the approximation ratio for the solution y_0 of the instance x.

By our construction, $f_Q(x_d, y_d) = opt_Q(x_d)$ and $y_0 = g_2(x_d, y_d)$. By the scalability of Q,

$$|opt_Q(x_d) - f_Q(x, y_0)/d| = |f_Q(x_d, y_d) - f_Q(x, y_0)/d| \leq q(n). \tag{3}$$

Thus, $f_Q(x, y_0) \geq d \cdot opt_Q(x_d) - d \cdot q(n)$. Since $(x_d, 3q(n)/\epsilon)$ is in $Q_{\geq}$, we have $opt_Q(x_d) \geq 3q(n)/\epsilon$, which gives (note $0 < \epsilon < 1$ thus $d/\epsilon \geq d$):

$$f_Q(x, y_0) \geq 3d \cdot q(n)/\epsilon - d \cdot q(n) = q(n)(3d/\epsilon - d) \geq 2dq(n)/\epsilon. \tag{4}$$

Now from (3) and the inequality $|opt_Q(x_d) - opt_Q(x)/d| \leq q(n)$, we get

$$\begin{aligned} |opt_Q(x)/d - f_Q(x, y_0)/d| &\leq |opt_Q(x_d) - opt_Q(x)/d| + |opt_Q(x_d) - f_Q(x, y_0)/d| \\ &\leq 2q(n). \end{aligned}$$

Thus $opt_Q(x) - f_Q(x, y_0) \leq 2dq(n)$ (recall that Q is a maximization problem). This eventually gives us

$$opt_Q(x)/f_Q(x, y_0) \leq 1 + 2dq(n)/f_Q(x, y_0) \leq 1 + \epsilon.$$

The last inequality is from (4). In conclusion, the approximation ratio of the solution y_0 for the instance x is bounded by $1 + \epsilon$.

This proves that the algorithm above is an FPTAS for the problem Q, completing the proof of the theorem. □

As an application of Theorem 1, the scalability as shown earlier and the well-known dynamic programming algorithm of running time $O(nk^m)$ [15] for the MAKESPAN problem conclude immediately that the MAKESPAN problem has an FPTAS when the number m of processors is a fixed constant. This is a major result in [22].

We make a few remarks on Theorem 1. Since the first group of publications on FPTAS for NP optimization problems [17,22], there has been a line of research trying to characterize problems in FPTAS [3,20,23]. Most of the early work in this direction [3,20] characterizes the class FPTAS in terms of certain polynomial time computable functions. These characterizations do not provide any clue on how to detect the existence of such functions, or on how to develop FPTAS for the problems (the interested readers are referred to [20], Theorem 4.20, for a more detailed discussion). Very recently, Woeginger [23], in an effort to overcome this difficulty, considered a class of optimization problems that can be formulated via dynamic programming of certain structures. He showed that as long as the cost and transition functions of such problems satisfy certain arithmetical and structural conditions, the problems have FPTAS.

In comparison to these related works, Theorem 1 in this paper seems to have the following advantages. First, as we have shown for the MAKESPAN problem, the scalability property of an NP optimization problem is satisfied in most cases and, in general, can be checked in a straightforward manner. Thus, in most

cases, the existence of FPTAS for an NP optimization problem is reduced to the development of an efficient-FPT algorithm for the problem. Moreover, the proof of Theorem 1 describes in detail how an efficient-FPT algorithm is converted into an FPTAS algorithm. On the other hand, Theorem 1 seems to cover more FPTAS problems than Woeginger's characterization [23]: generally, a dynamic programming formulation for an NP optimization problem directly implies an efficient-FPT algorithm for the problem.

4 Planar W-Hierarchy and EPTAS

In the previous section, we have shown how a subclass of the parameterized class FPT, the class efficient-FPT, provides a nice characterization for the approximation class FPTAS. In this section, we study the approximation class EPTAS in terms of the parameterized class, the W-hierarchy.

We note that a significant amount of research has been done on studying the approximation properties in terms of their syntactic descriptions. For instance, Papadimitriou and Yannakakis [21] introduced the syntactic classes MAXNP and MAXSNP of optimization problems, which, via proper approximation ratio preserving reductions, turn out to be exactly the class of NP optimization problems that have polynomial time constant-ratio approximation algorithms [19]. Khanna and Motwani [18] proposed the syntactic classes MPSAT, TMAX, and TMIN by enforcing a planar structure on first order Boolean formulas of depth 3, and showed that most known PTAS problems are expressible by these classes.

In an approach parallel to that of Khanna and Motwani [18], we study the approximation class EPTAS by enforcing a planar structure on the W-hierarchy in parameterized complexity. A circuit is a *Π_h-circuit* if it has depth h and its output node is an AND gate. A Π_h-circuit is a *Π_h^+-circuit* if all of its inputs are labeled by positive literals, and is a *Π_h^--circuit* if all of its inputs are labeled by negative literals. A Π_h-circuit α is *planar* if α becomes a planar graph after removing the output gate in α. Without loss of generality, we can assume that circuits are of a special leveled form where all inputs are in level 0, and all AND and OR gates are organized into alternating levels with edges only going from a level to the next level [9].

Definition 6. We define the following syntactic optimization classes:

PLANAR MIN-$W[h]$: consists of every optimization problem Q such that each instance of Q can be expressed as a planar Π_h^+-circuit α, and the problem is to look for a satisfying assignment of minimum weight for α.

PLANAR MAX-$W[h]$: consists of every optimization problem Q such that each instance of Q can be expressed as a planar Π_h^--circuit α, and the problem is to look for a satisfying assignment of maximum weight for α.

PLANAR $W[h]$-SAT: consists of every optimization problem Q such that each instance of Q can be expressed as a planar Π_h-circuit α, and the problem is to look for an assignment that satisfies the largest number of depth-$(h-1)$ gates in the circuit α.

We make a few remarks on the above optimization classes. The classes PLANAR MIN-$W[h]$, PLANAR MAX-$W[h]$, and PLANAR $W[h]$-SAT are optimization versions, with a planarity constraint, of the problem WCS(h), which is the representative complete problem for the hth level $W[h]$ of the W-hierarchy in parameterized complexity theory [12]. The class PLANAR $W[h]$-SAT captures the optimization problems such as the MAXIMUM SATISFIABILITY problem, where the objective is to construct a solution that satisfies the maximum number of constraints. In particular, the problem PLANAR MAXSAT formulated by Khanna and Motwani [18] belongs to the class PLANAR $W[2]$-SAT. The classes PLANAR MIN-$W[h]$ and PLANAR MAX-$W[h]$ capture the optimization problems where the objective is to construct an optimal (minimum or maximum) solution that satisfies all the constraints. Most optimization problems on planar graphs belong to the classes PLANAR MIN-$W[h]$ or PLANAR MAX-$W[h]$. For example, for an instance G of the MINIMUM VERTEX COVER on planar graphs, we can convert G into a planar Π_2^+-circuit α_G by making each vertex v in G an input of α_G and replacing each edge $[v, w]$ in G by an OR gate with the two inputs v and w, which is connected to the unique output AND gate of the circuit α_G. It is easy to see that the minimum vertex covers of the graph G correspond to the minimum weight assignments that satisfy the circuit α_G, and vice versa. We will show that all optimization problems expressible by our syntactic classes have EPTAS.

Let G be a planar graph (not necessarily connected) and $\pi(G)$ be a planar embedding of G. A vertex v is in *layer*-1 in $\pi(G)$ if v is on the boundary of the unbounded region of $\pi(G)$. We define G_1 to be the subgraph of G induced by all layer-1 vertices. Inductively, a vertex v is in layer-i, $i > 1$, if v is on the unbounded region of the embedding of the graph $G - (G_1 \cup \ldots \cup G_{i-1})$ induced by the embedding $\pi(G)$. Define G_i to be the subgraph of G induced by all layer-i vertices. The embedding $\pi(G)$ is *q-outerplanar* if it has at most q layers.

Now consider a planar Π_h^+-circuit α_w with output gate w. Let $\alpha = \alpha_w - w$ be the subgraph of α_w with the output gate w removed. By the definition, the graph α has a planar embedding $\pi(\alpha)$. Let G be a subgraph of α that is induced by q consecutive layers in $\pi(\alpha)$, where $q \geq 2h$, and let $\pi(G)$ be the embedding of G induced from $\pi(\alpha)$. Obviously, the embedding $\pi(G)$ is q-outerplanar. We consider the following optimization problem:

MIN (h, q)-SAT
Given the graph G and the q-outerplanar embedding $\pi(G)$ of G, as defined above, construct an assignment of minimum weight for the input variables in G so that all depth-$(h-1)$ gates in α_w that are in the middle $q - 2h$ layers in $\pi(G)$ (i.e., the $(h+1)$st, ..., and the $(q-h)$th layers in $\pi(G)$) are satisfied.

We point out that assigning all input variables in G the value 1 will satisfy all depth-$(h-1)$ gates in the middle $q - 2h$ layers in $\pi(G)$. This is because all literals in α_w are positive, and α_w has depth h. So any input variable or any gate that is connected via a path in α_w to a depth-$(h-1)$ gate in the middle $q - 2h$ layers in $\pi(G)$ must necessarily be contained in G, and hence, when all

these input variables in G are assigned the value 1, all the depth-$(h-1)$ gates in the middle $q-2h$ layers in $\pi(G)$ will be satisfied.

Lemma 1. *The problem* MIN (h,q)-SAT *can be solved in time* $O(81^q n)$.

Using Lemma 1, and an approach similar to that in [5], we develop an EPTAS for the PLANAR MIN-$W[h]$ problem. For an instance α_w of PLANAR MIN-$W[h]$ and a given $\epsilon > 0$, we let $q = 2h(\lceil 1/\epsilon \rceil + 1)$. We start with a planar embedding $\pi(\alpha)$ of the circuit α_w with the output gate w removed ($\alpha = \alpha_w - w$). We divide the layers of $\pi(\alpha)$ into "chunks", each consisting of q consecutive layers, such that two consecutive chunks overlap with $2h$ layers. We then apply Lemma 1 to construct a minimum weight assignment to the input variables in each chunk that satisfies all depth-$(h-1)$ gates in the middle $q-2h$ layers in the chunk. Now we merge the assignments over all chunks to obtain an assignment for the input variables in α_w satisfying α_w. It can be shown that this gives an approximation algorithm of ratio $1+\epsilon$ running in time $O(81^{2h/\epsilon}n)$ (and hence an EPTAS) for the PLANAR MIN-$W[h]$ problem. Similar approaches give EPTAS for the PLANAR MAX-$W[h]$ and PLANAR $W[h]$-SAT problems. We have the following theorem.

Theorem 2. *For each integer* $h \geq 1$, *the classes* PLANAR MIN-$W[h]$, PLANAR MAX-$W[h]$, *and* PLANAR $W[h]$-SAT *are subclasses of the class EPTAS.*

Cesati and Trevisan [8] proved that if an optimization problem is in the class EPTAS then its parameterized version is fixed-parameter tractable. Combining this with Theorem 2, we get the following:

Corollary 1. *For every positive integer* h, *the classes* PLANAR MIN-$W[h]$, PLANAR MAX-$W[h]$, *and* PLANAR $W[h]$-SAT *are subclasses of FPT.*

As a byproduct of the techniques use to prove Lemma 1, we solve an open problem posed by Downey and Fellows ([12], page 482), who asked about the parameterized complexity of the following problem:

PLANAR t-NORMALIZED WEIGHTED SATISFIABILITY
Given a Π_t-circuit α that is a planar graph in the strict sense (i.e., even without removing the output gate) and a parameter k, does α have a satisfying assignment of weight k?

Theorem 3. *For each fixed integer* t, *the* PLANAR t-NORMALIZED WEIGHTED SATISFIABILITY *problem is solvable in polynomial time.*

Theorem 3 implies that the PLANAR t-NORMALIZED WEIGHTED SATISFIABILITY problem is fixed-parameter tractable, answering the open problem posed in [12].

References

1. J. ALBER, H. BODLAENDER, H. FERNAU, T. KLOKS, AND R. NIEDERMEIER, Fixed parameter algorithms for dominating set and related problems on planar graphs, *Algorithmica 33,* (2002), pp. 461-493.
2. G. AUSIELLO, P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI-SPACCAMELA, AND M. PROTASI, *Complexity and Approximation, Combinatorial optimization problems and their approximability properties,* Springer Verlag, (1999).
3. G. AUSIELLO, A. MARCHETTI-SPACCAMELA, AND M. PROTASI, Toward a unified approach for the classification of NP-complete optimization problems, *Theoretical Computer Science 12,* (1980), pp. 83-96.
4. S. ARORA, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, *J. ACM 45,* (1998), pp. 753-782.
5. B. BAKER, Approximation algorithms for NP-complete problems on planar graphs, *J. ACM 41,* (1994), pp. 153-180.
6. L. CAI AND J. CHEN, On fixed-parameter tractability and approximability of NP optimization problems, *Journal of Computer and System Sciences 54,* (1997), pp. 465-474.
7. L. CAI, M. FELLOWS, D. JUEDES AND F. ROSAMOND, On efficient polynomial-time approximation schemes for problems on planar structures, *Journal of Computer and System Sciences,* to appear.
8. M. CESATI AND L. TREVISAN, On the efficiency of polynomial time approximation schemes, *Information Processing Letters 64,* (1997), pp. 165-171.
9. J. CHEN, Characterizing parallel hierarchies by reducibilities, *Information Processing Letters 39,* (1991), pp. 303-307.
10. J. CHEN AND A. MIRANDA, A polynomial time approximation scheme for general multiprocessor job scheduling, *SIAM J. on Comp. 31,* (2001), pp. 1-17.
11. R. DOWNEY, Parameterized complexity for the skeptic, *Proc. 18th IEEE Annual Conference on Computational Complexity* (CCC'03), (2003), pp. 132-153.
12. R. DOWNEY AND M. FELLOWS, *Parameterized Complexity,* Springer-Verlag, (1999).
13. M. FELLOWS, Parameterized complexity: the main ideas and some research frontiers, *LNCS 2223* (ISAAC'01), pp. 291-307, Springer-Verlag, Berlin (2001).
14. M. FRICK AND M. GROHE, Deciding first-order properties of locally tree-decomposable structures, *J. ACM 48,* (2001), pp. 1184-1206.
15. M. GAREY AND D. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* H. Freeman, New York, (1979).
16. D. HOCHBAUM, *Approximation Algorithms for NP-hard Problems,* PWS Publishing Company, Boston, MA, 1997.
17. O. IBARRA AND C. KIM, Fast approximation algorithms for the knapsack and sum of subset problems, *J. ACM 22,* (1975), pp. 463–468.
18. S. KHANNA AND R. MOTWANI, Towards a syntactic characterization of PTAS, *Proc. 28th Annual ACM Symp. on Theory of Computing* (STOC'96), (1996), pp. 468-477.
19. S. KHANNA, R. MOTWANI, M. SUDAN, AND U. VAZIRANI, On syntactic versus computational views of approximability, *SIAM J. on Comp. 28,* (1998), pp. 164-191.
20. A. PAZ AND S. MORAN, Non deterministic polynomial optimization problems and their approximations, *Theoretical Computer Science 15,* (1981), pp. 251-277.

21. C. PAPADIMITRIOU AND M. YANNAKAKIS, Optimization, approximation, and complexity classes, *Journal of Computer and System Sciences 43,* (1991), pp. 425-440.
22. S. SAHNI, Algorithms for scheduling independent tasks, *J. ACM 23,* (1976), pp. 116-127.
23. G. WOEGINGER, When does a dynamic programming formulation guarantee the existence of an FPTAS? *Proc. 10th Annual ACM-SIAM Symp. on Discrete Algorithms* (SODA'99), (1999), pp. 820-829.

Epistemic Foundation of the Well-Founded Semantics over Bilattices

Yann Loyer[1] and Umberto Straccia[2]

[1] PRiSM (CNRS UMR 8144), Université de Versailles, FRANCE
Yann.Loyer@prism.uvsq.fr
[2] I.S.T.I. - C.N.R., Via G. Moruzzi,1 I-56124 Pisa (PI) ITALY
Umberto.Straccia@isti.cnr.it

Abstract. We define new, both model-theoretical and fixpoint-based, characterizations of the well-founded semantics for logic programs in the general setting of bilattices. This work lights the role of the CWA, used in the well-founded semantics as a carrier of falsehood, and shows that the definition of that semantics does not require any separation of positive and negative information nor any program transformation.

1 Introduction

One of the most important problems of logic programming consists in defining the intended meaning or semantics of any given logic program. Classical logic programming has the set $\{\mathtt{f}, \mathtt{t}\}$ (false, true) as its intended truth space, and the usual semantics of a negation-free logic program is given by its unique minimal Herbrand model [7], relying on the *Closed World Assumption* (CWA) to *complete* the available knowledge. The CWA assumes that all atoms not entailed by a program are false [21], and is motivated by the fact that explicit representation of negative information in logic programs is not feasible since the addition of explicit negative information could overwhelm a system. But, in order to increase the expressivity of the language, it is often necessary to allow some non-monotonic modes of negation, having as a consequence that the existence of such a model is no more guaranteed.

A widely used solution consists in allowing partial models by extending the classical set of truth-values $\{\mathtt{f}, \mathtt{t}\}$ to the set $\{\mathbf{f}, \mathbf{t}, \bot\}$, where $\bot$ stands for *unknown* and is closely related to null values in database systems. Over that set can be defined two orderings: the truth order $\preceq_t$ that extends the classical order $\mathbf{f} \preceq_t \bot \preceq_t \mathbf{t}$), and the knowledge order $\preceq_k$ ($\bot$ represents less knowledge than $\mathtt{f}$ and $\mathtt{t}$ that are both incomparable).

A first approach [10,13] has lead to the *Kripke-Kleene semantics* of logic programs, based on an extension $\Phi_{\mathcal{P}}$ to the Van Emden-Kowalski's immediate consequence operator $T_{\mathcal{P}}$ [7], and thus on the classical evaluation of negation (the evaluation of a negative literal $\neg A$ is given by the negation of the evaluation of A). That semantics can be defined as the least model of the program and computed as the least fixpoint of $\Phi_{\mathcal{P}}$ with respect to the knowledge order. Unfortunately

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 513–524, 2004.
© Springer-Verlag Berlin Heidelberg 2004

that semantics is often considered to be too weak in the sense that it does not provide enough knowledge.

A more informative approach has lead to the *stable model* approach [11], which defines a whole family of models, based on the so-called Gelfond-Lifschitz transform [8]. Informally, the main principle of that approach is the *separation of the role of positive and negative information.* That is, the negative literals of a program are first evaluated to obtain a negation-free program whose minimal Herbrand model can be computed. As a consequence, this separation avoids the natural management of classical negation which is a major feature of the Kripke-Kleene semantics of logic programs with negation. To overcome the fact that some programs have no stable model over the set of truth-values {f,t}, Przymusinski ([19,20]) extended the Gelfond-Lifschitz transform, and thus the notion of stable model semantics to allow partial stable models. Remarkably, one among these stable models, the *minimal* one according to the knowledge ordering, is often considered as the favorite one and, as shown in [20], is one-to-one related with the so-called *well-founded semantics* defined independently from the stable models in [22]. It is not unusual that, rather than to compute the whole set of stable models, one computes the well-founded semantics only.

The following step was to move from three-valued logics, allowing the representation of incomplete information, to the well-known four-valued set of truth-values $\mathcal{FOUR} = \{\mathbf{f}, \mathbf{t}, \bot, \top\}$, introduced by Belnap ([3]) to allow the representation of *inconsistency* (denoted $\top$) as well. This process of enlarging the set of truth-values culminated with Fitting's progressive work (e.g. [8,9]) on giving meaning to logic programs by relying on *bilattices* [12]. Bilattices, where $\mathcal{FOUR}$ is the simplest non-trivial one, play an important role in logic programming, and in knowledge representation in general, allowing in particular reasoning with inconsistency and uncertainty (see e.g. [1,4,6,14,15,16,17,18]). Fitting proposed an extension of the stable models family by extending the Gelfond-Lifschitz transform from logic programming over two- or three-valued logics to logic programming over bilattices [8]. As a consequence, that work proposed an extension to that setting of the well-founded semantics as well, as being the least fixpoint, with respect to a given knowledge order, of the extension of the Gelfond-Lifschitz transform.

The primary goal of this study is to show, in the quite general setting of bilattices as space of truth-values, that neither this separation of positive and negative information is necessary nor any program transformation is required to characterize the well-founded semantics. Indeed, we show that it can be defined as a simple, natural and epistemic extension of the Kripke-Kleene semantics. Informally, we view the CWA as an additional source of information to be used for information completion, or more precisely, as a carrier for falsehood, to be considered cumulatively to the Kripke-Kleene semantics. To this end, given a logic program $\mathcal{P}$ and an interpretation I representing our current knowledge about some intended model of $\mathcal{P}$, we define the notion of *support provided by the CWA to the program $\mathcal{P}$ with respect to I.* The support provided by the CWA to a program $\mathcal{P}$ with respect to an interpretation I extends the notion of unfounded

set [22] from three-valued logics to bilattices, and determines in a principled way how much *false* knowledge, i.e. how much knowledge provided by the CWA, can "safely" be joined to I with respect to the program $\mathcal{P}$. Then, we define the set of *supported models* to be the set of those models that can not be completed anymore by the CWA, i.e. that contains their own support. We provide different characterizations of that family of models in terms of an operator managing negation classically, i.e. in terms of $\Phi_{\mathcal{P}}$. Finally, we show that stable models are supported models and that the well-founded semantics coincides with the least supported model with respect to the knowledge order.

As a consequence, we propose alternative, both epistemic and fixpoint-based, characterizations of the well-founded semantics to the well-known, widely applied and long studied technique based on the separation of positive and negative information, by reverting to the classical interpretation of negation, i.e. we characterize negation-as-failure as standard negation. While Fitting treats negation-as-failure in a special way and unlike other connectives, our approach is an attempt to relate the semantics of logic programs to a standard model-theoretic account of rules. We emphasize the possibility to analyze logic programs using standard logical means as the notion of interpretation and information ordering, i.e. knowledge ordering. Therefore, our approach in principle does not depend on the presence of any specific connective, such as negation-as-failure, nor on any specific syntax of rules. Moreover our approach lights, in the general setting of logic programming over bilattices, the role of the CWA in the well-founded semantics. Due to the generality and the purely algebraic nature of our results, as just monotone operators over bilattices are postulated, the epistemic characterization of the well-founded semantics given in this study can be applied in other contexts as well (e.g. uncertainty and/or paraconsistency in logic programming, or nonmonotonic logics).

The remaining of the paper is organized as follows. In order to make the paper self-contained, in the next section, we will briefly recall definitions and properties of bilattices and logic programs. Section 3 is the main part of this work, where we present our characterizations of the well-founded semantics, while Section 4 concludes. Due to lack of space, proofs are omitted and are available from the authors' on-line version.

2 Preliminaries

2.1 Bilattices

The simplest non-trivial bilattice, called $\mathcal{FOUR}$, is due to Belnap ([3]), who introduced a logic intended to deal with incomplete and/or inconsistent information – see also [2]. $\mathcal{FOUR}$ already illustrates many of the basic properties concerning bilattices. Essentially, $\mathcal{FOUR}$ extends the classical truth set $\{\mathtt{f},\mathtt{t}\}$ to its power set $\{\{\mathtt{f}\},\{\mathtt{t}\},\emptyset,\{\mathtt{f},\mathtt{t}\}\}$, where we can think that each set indicates the amount of information we have in terms of truth: so, $\{\mathtt{f}\}$ stands for *false,* $\{\mathtt{t}\}$ for *true* and, quite naturally, $\emptyset$ for lack of information or *unknown,* and $\{\mathtt{f},\mathtt{t}\}$ for *inconsistent* information (for ease, we use $\mathtt{f}$ for $\{\mathtt{f}\}$, $\mathtt{t}$ for $\{\mathtt{t}\}$, $\bot$ for $\emptyset$ and

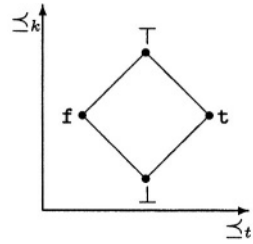


Fig. 1. The logic $\mathcal{FOUR}$.

$\top$ for $\{\mathtt{f},\mathtt{t}\}$). $\mathcal{FOUR}$ has two quite intuitive and natural 'orthogonal' orderings, $\preceq_k$ and $\preceq_t$ (see Figure 1), each giving to $\mathcal{FOUR}$ the structure of a complete lattice. One is the so-called *knowledge ordering,* denoted $\preceq_k$, and is based on the subset relation, that is, if $x \subseteq y$ then x represents 'more information' than y (e.g. $\bot = \emptyset \subseteq \{\mathtt{t}\} = \mathtt{t}$, i.e. $\bot \preceq_k \mathtt{t}$). The other ordering is the so-called *truth ordering,* denoted $\preceq_t$. Here $x \preceq_t y$ means that x is 'at least as false as y is, and y is at least as true as x is', i.e. $x \cap \{\mathtt{t}\} \subseteq y \cap \{\mathtt{t}\}$ and $y \cap \{\mathtt{f}\} \subseteq x \cap \{\mathtt{f}\}$ (e.g. $\bot \preceq_t \mathtt{t}$).

Formally [12,9], a *bilattice* is a structure $\langle \mathcal{B}, \preceq_t, \preceq_k \rangle$ where $\mathcal{B}$ is a non-empty set and $\preceq_t$ and $\preceq_k$ are both partial orderings giving $\mathcal{B}$ the structure of a *complete lattice* with a top and bottom element. *Meet (or greatest lower bound)* and *join (or least upper bound)* under $\preceq_t$, denoted $\wedge$ and $\vee$, correspond to extensions of classical conjunction and disjunction. On the other hand, *meet and join under* $\preceq_k$ are denoted $\otimes$ and $\oplus$. $x \otimes y$ corresponds to the maximal information x and y can agree on, while $x \oplus y$ simply combines the information represented by x with that represented by y. *Top and bottom under* $\preceq_t$ are denoted $\mathtt{t}$ and $\mathtt{f}$, and *top and bottom under* $\preceq_k$ are denoted $\top$ and $\bot$, respectively. We will assume that bilattices are *infinitary distributive bilattices* in which all distributive laws connecting $\wedge, \vee, \otimes$ and $\oplus$ hold. We also assume that every bilattice satisfies the *infinitary interlacing conditions,* i.e. each of the lattice operations $\wedge, \vee, \otimes$ and $\oplus$ is monotone w.r.t. both orderings. Finally, we assume that each bilattice has a *negation,* i.e. an operator $\neg$ that reverses the $\preceq_t$ ordering, leaves unchanged the $\preceq_k$ ordering, and verifies $\neg\neg x = x$.

2.2 Logic Programs and Models

We recall here the definitions given in [8]. This setting is as general as possible, so that the results proved in this paper will be widely applicable.

Logic programs. Consider an alphabet of predicate symbols, of constants, of function symbols and variable symbols. A *term,* t, is either a variable x, a constant c or of the form $f(t_1, \ldots, t_n)$, where f is an n-ary function symbol and all t_i are terms. An *atom,* A, is of the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate symbol and all t_i are terms. A literal, l, is of the form A or $\neg A$, where

A is an atom. A *formula,* φ, is an expression built up from the literals and the members of a bilattice $\mathcal{B}$ using $\wedge, \vee, \otimes, \oplus, \exists$ and $\forall$. Note that members of the bilattice may appear in a formula, e.g. in $\mathcal{FOUR}$, $(p \wedge q) \oplus (r \otimes \mathtt{f})$ is a formula. A *rule* is of the form $p(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n)$, where p is an n-ary predicate symbol and all x_i are variables. The atom $p(x_1, \ldots, x_n)$ is called the *head,* and the formula $\varphi(x_1, \ldots, x_n)$ is called the *body.* It is assumed that the free variables of the body are among $x_1, \ldots, x_n$. Free variables are thought of as universally quantified. A *logic program,* denoted with $\mathcal{P}$, is a finite set of rules. The *Herbrand universe* of $\mathcal{P}$ is the set of *ground* (variable-free) terms that can be built from the constants and function symbols occurring in $\mathcal{P}$, while the *Herbrand base* of $\mathcal{P}$ (denoted $B_{\mathcal{P}}$) is the set of ground atoms over the Herbrand universe.

Given $\mathcal{P}$, the set $\mathcal{P}^*$ is constructed as follows; *(i)* put in $\mathcal{P}^*$ all ground instances of members of $\mathcal{P}$ (over the Herbrand base); *(ii)* if a ground atom A is not head of any rule in $\mathcal{P}^*$, then add the rule $A \leftarrow \mathtt{f}$ to $\mathcal{P}^*$; [1] and *(iii)* replace several ground rules in $\mathcal{P}^*$ having same head, $A \leftarrow \varphi_1$, $A \leftarrow \varphi_2$, ...with $A \leftarrow \varphi_1 \vee \varphi_2 \vee \ldots$.[2] Note that in $\mathcal{P}^*$, each ground atom appears in the head of *exactly one* rule.

Interpretations. An *interpretation of a logic program* on the bilattice $\langle \mathcal{B}, \preceq_t, \preceq_k \rangle$ is a mapping from ground atoms to members of $\mathcal{B}$. An interpretation I is extended from atoms to formulae as follows: *(i)* for $b \in \mathcal{B}$, $I(b) = b$; *(ii)* for formulae φ and φ', $I(\varphi \wedge \varphi') = I(\varphi) \wedge I(\varphi')$, and similarly for $\vee, \otimes, \oplus$ and $\neg$; and *(iii)* $I(\exists x \varphi(x)) = \bigvee \{I(\varphi(t)) : t$ ground term$\}$, and similarly for universal quantification[3]. The truth and knowledge orderings are extended from $\mathcal{B}$ to the set $\mathcal{I}(\mathcal{B})$ of all interpretations as follows: *(i)* $I_1 \preceq_t I_2$ iff $I_1(A) \preceq_t I_2(A)$, for every ground atom A; and *(ii)* $I_1 \preceq_k I_2$ iff $I_1(A) \preceq_k I_2(A)$, for every ground atom A. We define $(I_1 \wedge I_2)(\varphi) = I_1(\varphi) \wedge I_2(\varphi)$, and similarly for the other operations. With $\mathtt{I_f}$ and $\mathtt{I}_\perp$ denote the bottom interpretations under $\preceq_t$ and $\preceq_k$ respectively (they map any atom into $\mathtt{f}$ and $\perp$, respectively). It is easy to see that $\langle \mathcal{I}(\mathcal{B}), \preceq_t, \preceq_k \rangle$ is an infinitary interlaced and distributive bilattice as well.

Classical setting. Note that in a *classical logic program* the body is a conjunction of literals. Therefore, if $A \leftarrow \varphi \in \mathcal{P}^*$, then $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$ and $\varphi_i = L_{i_1} \wedge \ldots \wedge L_{i_n}$. Furthermore, a *classical total interpretation* is an interpretation over $\mathcal{FOUR}$ such that an atom is mapped into either $\mathtt{f}$ or $\mathtt{t}$. A *partial classical interpretation* is a classical interpretation where the truth of some atom may be left unspecified. This is the same as saying that the interpretation maps all atoms into either $\mathtt{f}, \mathtt{t}$ or $\perp$. For a set of literals X, with $\neg.X$ we indicate the set $\{\neg L : L \in X\}$, where for any atom A, $\neg\neg A$ is replaced with A. Then, a classical interpretation (total or partial) can also be represented as a set of literals, and inversely.

---

[1] It is a standard practice in logic programming to consider such atoms as *false.*

[2] There could be infinitely many grounded rules with same head, but the semantics behavior is unproblematic.

[3] The bilattice is complete w.r.t. $\preceq_t$, so existential and universal quantification are well-defined.

Models. An interpretation I is a *model* of a logic program $\mathcal{P}$, denoted by $I \models \mathcal{P}$, if and only if for each rule $A \leftarrow \varphi$ in $\mathcal{P}^*$, $I(\varphi) \preceq_t I(A)$.

Fitting ([8,9]) identifies a set of models which obeys the so-called *Clark-completion* procedure [5], by mainly replacing each occurrence of $\leftarrow$ in $\mathcal{P}^*$ with $\leftrightarrow$: an interpretation I is a *Clark-completion model, cl-model* for short, of a logic program $\mathcal{P}$, denoted by $I \models_{cl} \mathcal{P}$, iff for each rule $A \leftarrow \varphi$ in $\mathcal{P}^*$, $I(A) = I(\varphi)$. It can easily be shown that Clark-completion models have also an alternative characterization given by $I \models_{cl} \mathcal{P}$ iff $I = \min_{\preceq_t}\{J : J \models \mathcal{P}[I]\}$, where $\mathcal{P}[I] = \{A \leftarrow I(\varphi) : A \leftarrow \varphi \in \mathcal{P}^*\}$.

Program k-completions. We introduce here the notion of program *knowledge completion,* or simply, k-completion with I, denoted $\mathcal{P} \oplus I$. The idea is to enforce any model J of $\mathcal{P} \oplus I$ to contain at least the knowledge determined by $\mathcal{P}$ and I. That is, the *k-completion* of $\mathcal{P}$ with I, is the program obtained by replacing any rule of the form $A \leftarrow \varphi \in \mathcal{P}$ by $A \leftarrow \varphi \oplus I(A)$.

2.3 Semantics of Logic Programs

The semantics of a logic program $\mathcal{P}$ is usually determined by selecting a particular model, or a set of models, of $\mathcal{P}$. We recall the definitions of the three most popular and widely studied semantics for logic programs with negation in increasing order of knowledge.

The Kripke-Kleene semantics. The Kripke-Kleene semantics [10] has a simple, intuitive and epistemic characterization. The *Kripke-Kleene model* of $\mathcal{P}$, denoted $KK(\mathcal{P})$, is the $\preceq_k$-least cl-model of $\mathcal{P}$, i.e. $KK(\mathcal{P}) = \min_{\preceq_k}(\{I : I \models_{cl} \mathcal{P}\})$.

The Kripke-Kleene semantics has also an alternative, and better known, fixpoint characterization, by relying on the well-known $\Phi_{\mathcal{P}}$ immediate consequence operator.

Definition 1 ($\Phi_{\mathcal{P}}$). *The* immediate consequence operator $\Phi_{\mathcal{P}} : \mathcal{I}(\mathcal{B}) \rightarrow \mathcal{I}(\mathcal{B})$ *is defined as follows: for any ground atom A such that $A \leftarrow \varphi$ occurs in $\mathcal{P}^*$, $\Phi_{\mathcal{P}}(I)(A) = I(\varphi)$.*

$\Phi_{\mathcal{P}}$ is a generalization of the Van Emden-Kowalski's immediate consequence operator $T_{\mathcal{P}}$ [7] to bilattices under Clark's program completion. Interesting properties of $\Phi_{\mathcal{P}}$ are that *(i)* $\Phi_{\mathcal{P}}$ relies on the classical evaluation of negation, i.e. the evaluation of a negative literal $\neg A$ is given by the negation of the evaluation of A; *(ii)* the cl-models of a program $\mathcal{P}$ coincide with the fixpoints of $\Phi_{\mathcal{P}}$; and *(iii)* $\Phi_{\mathcal{P}}$ is monotone with respect to the knowledge ordering and, thus, has a $\preceq_k$-least fixpoint, which coincides with the Kripke-Kleene semantics of $\mathcal{P}$. Note that $\Phi_{\mathcal{P}}(I) = \min_{\preceq_t}\{J : J \models \mathcal{P}[I]\}$.

As a consequence, *all definitions and properties given in this paper in terms of $\Phi_{\mathcal{P}}$ and/or cl-models may be given in terms of models as well.* As $\Phi_{\mathcal{P}}$ is a well-known operator, for ease of presentation we will continue to rely on it.

Well-founded semantics and stable models. The original definition of the well-founded semantics [22] was formulated for classical logic programs. That

definition relies on the immediate consequence operator $T_{\mathcal{P}}$ over the set of partial interpretations, defined by $T_{\mathcal{P}}(I) = \{A \mid \exists A \leftarrow L_1, ..., L_n \in \mathcal{P} \ (\forall L_i \ (L_i \in I))\}$, for inferring positive information, and on the notion of *unfounded set* to complete that information with negative information. A set of instantiated atoms U is said to be unfounded with respect to a partial interpretation I if for all instantiated atoms $A \in U$ and for all rules $r \in P$ the following holds: $head(r) = A \Rightarrow \exists L \in body(r) \ (\neg L \in I \ \ or \ \ L \in U)$. Intuitively, a set of atoms is unfounded if every way to infer an atom in that set, i.e. every rule with such an atom in head, fails because some atom in the condition, i.e. in the body of the rule, is in contradiction with the current knowledge and/or is itself unfounded. The *well-founded semantics* of P is then defined to be the least fixpoint of the well-founded operator W_P, defined by $W_P(I) = T_P(I) \cup \neg.U_P(I)$, with respect to set inclusion, where $U_P(I)$ is the $\subseteq$-greatest unfounded set with respect to I.

The extension of the notions of the well-founded semantics to the context of bilattices is due to Fitting [8], through the fact that the least partial stable model coincides with the well-founded semantics [20]. He proposes a generalization of the Gelfond-Lifschitz transformation [11,20], based on the same basic principle that consists in separating the roles of positive and negative information. Formally, let I and J be two interpretations in $\langle \mathcal{I}(\mathcal{B}), \preceq_t, \preceq_k \rangle$. The *pseudo-interpretation* $I \triangle J$ over the bilattice is defined as follows: for a pure ground atom A, $(I \triangle J)(A) = I(A)$ and $(I \triangle J)(\neg A) = \neg J(A)$. Pseudo-interpretations are extended to non-literals in the obvious way. The *immediate consequence operator* $\Psi_{\mathcal{P}}: \mathcal{I}(\mathcal{B}) \times \mathcal{I}(\mathcal{B}) \rightarrow \mathcal{I}(\mathcal{B})$ is then defined by: for any ground atom A such that $A \leftarrow \varphi$ occurs in $\mathcal{P}^*$, $\Psi_{\mathcal{P}}(I, J)(A) = (I \triangle J)(\varphi)$. Using the monotonicity of $\Psi_{\mathcal{P}}$ in its first argument with respect to $\preceq_t$, Fitting defined the *stability operator* $\Psi'_{\mathcal{P}}: \mathcal{I}(\mathcal{B}) \rightarrow \mathcal{I}(\mathcal{B})$ as follows: $\Psi'_{\mathcal{P}}(I)$ is the $\preceq_t$-least fixpoint of the operator $\lambda x.\Psi_{\mathcal{P}}(x, I)$, i.e. $\Psi'_{\mathcal{P}}(I) = \mathrm{lfp}_{\preceq_t}(\lambda x.\Psi_{\mathcal{P}}(x, I))$.

Finally, following Fitting's formulation, a *stable model* for a logic program $\mathcal{P}$ is a fixpoint of $\Psi'_{\mathcal{P}}$. The operator $\Psi'_{\mathcal{P}}$ is monotone in the $\preceq_k$ ordering and, thus, has a $\preceq_k$-least fixpoint, denoted $WF(\mathcal{P})$. By definition, $WF(\mathcal{P})$ is known as the *well-founded model* of $\mathcal{P}$.

3 Well-Founded Semantics Revisited

In the previous sections we have seen that, while for the Kripke-Kleene semantics there is an intuitive epistemic and model theory-based characterization, for the well-founded semantics on bilattices this is likely not the case. In the following, we present our contribution, which mainly consists in defining both epistemic and fixpoint-based characterizations of the well-founded semantics over bilattices. We contribute, thus, to an alternative view of well-founded semantics over bilattices, to the well-known and long studied separation of positive and negative information in Fitting's computation.

We will rely on the following running example.

Example 1 (running example) *Consider the following logic program $\mathcal{P}$ with the following rules.*

$$p \leftarrow p$$
$$q \leftarrow \neg r$$
$$r \leftarrow \neg q \wedge \neg p$$

In the following table, we report the different interpretations and models studied in this paper: cl-models, supports, Kripke-Kleene (K) and well-founded (W) semantics, stable and supported models of $\mathcal{P}$.

	I_i			$s_{\mathcal{P}}(I_i)$					*stable*	*supported*
$I_i \models_{cl} \mathcal{P}$	p	q	r	p	q	r	K	W	*models*	*models*
I_1	$\bot$	$\bot$	$\bot$	f	$\bot$	$\bot$	•			
I_2	$\bot$	t	f	f	$\bot$	f				
I_3	f	$\bot$	$\bot$	f	$\bot$	$\bot$		•	•	•
I_4	f	f	t	f	f	$\bot$			•	•
I_5	f	t	f	f	$\bot$	f			•	•
I_6	f	$\top$	$\top$	f	f	f			•	•
I_7	t	t	f	f	$\bot$	f				
I_8	$\top$	t	f	f	$\bot$	f				•
I_9	$\top$	$\top$	$\top$	f	f	f				•

3.1 Support

First we introduce the notion of *support,* denoted $s_{\mathcal{P}}(I)$, provided by the CWA to a logic program $\mathcal{P}$ with respect to a given interpretation I. Intuitively, we regard I as what we already know about an intended model of $\mathcal{P}$. On the basis of both the current knowledge I and the information expressed by the rules of $\mathcal{P}$, we want to complete our available knowledge I, by using the CWA. We regard the CWA as an additional source of information for *falsehood.* The support $s_{\mathcal{P}}(I)$ of $\mathcal{P}$ w.r.t. I determines in a principled way the maximal amount of falsehood provided by the CWA that can be "safely" joined to I. The main principle underlying safe interpretations can be explained as follows. For ease, let us consider $\mathcal{FOUR}$. Consider an interpretation I, which is our current knowledge about $\mathcal{P}$. Let us assume that the interpretation J, with $J \preceq_k \mathtt{I_f}$, indicates which atoms may be assumed as f. For any ground atom A, *J(A)* is the default 'false' information provided by J to the atom A. The completion of I with J is the interpretation $I \oplus J$. In order to accept this completion, we have to ensure that the assumed false knowledge about A, *J(A),* is entailed by $\mathcal{P}$ w.r.t. the completed interpretation $I \oplus J$. In other words, for $A \leftarrow \varphi \in \mathcal{P}^*$, assuming that A is false is safe if φ is false as well with respect to the current knowledge I completed by the assumed falsehood J, i.e. $J(A) \preceq_k (I \oplus J)(\varphi)$ should hold.

Definition 2 (safe interpretation). *Let $\mathcal{P}$ and I be a logic program and an interpretation, respectively. An interpretation J is* safe *w.r.t. $\mathcal{P}$ and I iff:*

1. $J \preceq_k \mathtt{I_f}$;
2. $J \preceq_k \Phi_{\mathcal{P}}(I \oplus J)$.

In the above definition, the first item dictates that any safe interpretation is a carrier of falsehood, i.e. a part of the CWA. If $J = \mathtt{I_f}$, then every ground atom is false. But, given I and $\mathcal{P}$, not necessarily all atoms can be considered as false (e.g., some atoms may be inferred true from the program) and we have to consider some weaker assumption $J \preceq_k \mathtt{I_f}$ of falsehood. The second item dictates that a safe interpretation is *cumulative,* i.e. as we proceed in deriving more precise approximations of an intended model of $\mathcal{P}$, the accumulated falsehood should be preserved. To illustrate that concept, consider the interpretation I_2 of our running example. I_2 dictates that p is unknown, q is true and that r is false. Consider the interpretations J_i defined as follows:

J_i	p	q	r
J_1	$\bot$	$\bot$	$\bot$
J_2	$\mathtt{f}$	$\bot$	$\bot$
J_3	$\bot$	$\bot$	$\mathtt{f}$
J_4	$\mathtt{f}$	$\bot$	$\mathtt{f}$

It is easy to verify that all the J_is are safe. The $\preceq_k$-least safe interpretation is J_1, while the $\preceq_k$-greatest one is $J_4 = J_1 \oplus J_2 \oplus J_3$. J_4 dictates that under I_2, we can 'safely' assume that both p and r are false. Note that if we join J_4 to I_2 we obtain the stable model I_5, where $I_2 \preceq_k I_5$. J_4 improves the knowledge expressed by I_2. One might wonder why we do not consider q false as well. Indeed, if we consider p, q and r false, after joining to I and applying $\Phi_\mathcal{P}$, we have that q becomes true, which is *knowledge-incompatible* with q's previous knowledge status (q is false). So, q's falsehood is not preserved, i.e. cumulative.

To give another intuitive reading of that notion, as anticipated, safe interpretations have an interesting reading once we restrict our attention to the classical framework of logic programming: indeed, the concept of safe interpretation reduces to that of unfounded set.

Theorem 1. *Let $\mathcal{P}$ and I be a classical logic program and a classical interpretation, respectively. Let X be a subset of $B_\mathcal{P}$. Then X is an unfounded set of $\mathcal{P}$ w.r.t. I iff $\neg.X \preceq_k \Phi_\mathcal{P}(I \oplus \neg.X)$* [4]*, i.e. $\neg.X$ is safe w.r.t. $\mathcal{P}$ and I.*

The safe assumptions are parts of the CWA that can be added to the current knowledge to complete it, thus among all possible safe interpretations, we privilege the maximal one under $\preceq_k$ in order to infer as much knowledge as possible.

Definition 3 (support). *Let $\mathcal{P}$ be a logic program and I an interpretation. The* support provided by the CWA *to $\mathcal{P}$ w.r.t. I, or simply* support *of $\mathcal{P}$ w.r.t. I, denoted $s_\mathcal{P}(I)$, is the $\preceq_k$-greatest safe interpretation w.r.t. $\mathcal{P}$ and I.*

It is easy to show that the support is a well-defined concept: given two safe interpretations J and J', then $J \oplus J' \preceq_k \mathtt{I_f}$ and, from the monotonicity of $\Phi_\mathcal{P}$ under $\preceq_k$, $J \oplus J' \preceq_k \Phi_\mathcal{P}(I \oplus J \oplus J')$ and, thus, $J \oplus J'$ is safe. Therefore, $s_\mathcal{P}(I) = \bigoplus\{J: J \text{ is safe w.r.t. } \mathcal{P} \text{ and } I\}$. We give also an alternative fixpoint characterization of the support to the model-theoretical one, and thus an effective method to compute it.

[4] Note that this condition can be rewritten as $\neg.X \subseteq \Phi_\mathcal{P}(I \cup \neg.X)$.

Theorem 2. *Consider the iterated sequence of interpretations defined by: for any* $i \geq 0$,

$$F_0^I = \mathtt{I_f}, \text{ and}$$
$$F_{i+1}^I = \mathtt{I_f} \otimes \Phi_{\mathcal{P}}(I \oplus F_i^I).$$

The sequence F_i^I *is (i) monotone non-increasing under* $\preceq_k$ *and, thus, reaches a fixpoint* F_ω^I*, for a limit ordinal* ω*; and (ii) monotone non-decreasing under* $\preceq_t$*. Furthermore,* $s_{\mathcal{P}}(I) = F_\omega^I$ *holds.*

Note that, for classical logic programs, the notions of support and greatest unfounded set are closely related as the set of (negative) literals corresponding to $s_{\mathcal{P}}(I)$ coincides with the negation of the greatest unfounded set $U_{\mathcal{P}}(I)$. It follows that the above theorem leads to a new fixpoint computation of $U_{\mathcal{P}}(I)$ based on $\Phi_{\mathcal{P}}$ only. Indeed, $\neg.U_{\mathcal{P}}(I)$ coincides with the limit of the sequence:

$$F_0^I = \neg.B_{\mathcal{P}}$$
$$F_{i+1}^I = \neg.B_{\mathcal{P}} \cap \Phi_{\mathcal{P}}(I \cup F_i^I)$$

Note also that the support can be seen as a monotone operator over the space of interpretations w.r.t. $\preceq_k$.

Theorem 3. *The support operator* $s_{\mathcal{P}}$ *is monotone w.r.t.* $\preceq_k$*. Moreover, if* $I \preceq_k J$*, then* $s_{\mathcal{P}}(J) \preceq_t s_{\mathcal{P}}(I)$.

Theorem 3 has an intuitive reading: the more knowledge we have about a ground atom *A*, the more we *know* (the more precise and informative we are) about *A*'s falsehood, i.e. the more falsehood can be provided by the CWA to *A*.

3.2 Supported Models

Among all possible models of a program $\mathcal{P}$, we will be especially interested in the models I that integrate their own support ($s_{\mathcal{P}}(I) \preceq_k I$). Such models tell us that we have reached the point where the additional source for falsehood provided by the CWA can not contribute further to improve our knowledge about $\mathcal{P}$.

Definition 4 (supported model). *Consider a logic program* $\mathcal{P}$*. An interpretation I is a* supported model *of* $\mathcal{P}$ *iff* $I \models_{cl} \mathcal{P}$ *and* $s_{\mathcal{P}}(I) \preceq_k I$.

Restricting our attention to classical logic programs, a partial interpretation is a supported models of $\mathcal{P}$ iff I is a cl-model containing the negation of its greatest unfounded set, i.e. $I \models_{cl} \mathcal{P}$ and $\neg.U_{\mathcal{P}}(I) \subseteq I$. Supported models have interesting properties :

Theorem 4. *The following are equivalent:*

1. I *is a supported model of* $\mathcal{P}$*;*
2. $I = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}(I)$*;*
3. $I \models_{cl} \mathcal{P} \oplus s_{\mathcal{P}}(I)$*;*
4. $I = \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}(I))$.

The above theorem states in different ways the same concept: supported models contain the amount of knowledge expressed by the program and their support. Thus, from a fixpoint characterization point of view, the set of supported models can be identified by the fixpoints of the monotone immediate consequence operator $\Pi_{\mathcal{P}}$ defined by $\Pi_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}(I))$ (an equivalent definition, in terms of fixpoints, is $\Pi_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}(I)$, which could be seen as an extension to bilattices of the well-founded operator $W_{\mathcal{P}}$). Thus, the set of supported models is a complete lattice under $\preceq_k$, and we show finally that the notion of supported models provide new characterizations of the well-founded semantics.

Theorem 5. *Stable models of $\mathcal{P}$ are supported models of $\mathcal{P}$, and the $\preceq_k$-least supported model of $\mathcal{P}$ coincides with the well-founded semantics of $\mathcal{P}$.*

Therefore $WF(\mathcal{P})$ can be computed by iterations of $\Pi_{\mathcal{P}}$ starting from $\mathbf{I}_{\perp}$. This lights the role of the CWA that can be seen as a source of falsehood used to complete the Kripke-Kleene semantics. Indeed, the well-founded semantics can be obtained by iterating $\Phi_{\mathcal{P}}$ from $\mathbf{I}_{\perp}$ (similarly to the Kripke-Kleene semantics), but adding to each iteration some knowledge provided by the CWA. Apart obtaining alternative epistemic characterization and computation method of the well-founded semantics to $\Psi'_{\mathcal{P}}$, the above theorem highlights the fact that, in the general context of logic programming over bilattices, neither a separation of positive and negative information is necessary, nor any program transformation is required for defining and computing the well-founded semantics.

4 Conclusions

In this study we have presented alternative formulations of the well-founded semantics. Our approach is purely based on algebraic and semantical aspects of informative monotone operators over bilattices. In this sense, we talk about epistemic foundation of the well-founded semantics. The main concept we rely on is based on the fact that we regard the closed world assumption as an additional source for falsehood and identify with the *support* the amount of falsehood carried on by the closed world assumption. We have shown that the well-founded semantics can be characterized as the knowledge minimal model containing its own support, i.e. $WF(\mathcal{P}) = \min_{\preceq_k}(\{I : I \models \mathcal{P} \text{ and } s_{\mathcal{P}}(I) \preceq_k I\})$. This indicates that the support may be seen as the added-value to the Kripke-Kleene semantics and lights the role of the CWA in the well-founded semantics over bilattices. It also shows that neither a separation of positive and negative information nor any program transformation is necessary.

References

1. O. Arieli. Paraconsistent declarative semantics for extended logic programs. *Annals of Mathematics and Artificial Intelligence,* 36(4):381–417, 2002.
2. O. Arieli and A. Avron. The value of the four values. *Artificial Intelligence Journal,* 102(1):97–141, 1998.

3. N. D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern uses of multiple-valued logic,* pages 5–37. Reidel, Dordrecht, NL, 1977.
4. H. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science,* 68:135–154, 1989.
5. K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and data bases,* pages 293–322. Plenum Press, New York, NY, 1978.
6. C. V. Damásio and L. M. Pereira. A survey of paraconsistent semantics for logic programs. In D. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems,* pages 241–320. Kluwer, 1998.
7. M. H. Van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM),* 23(4):733–742, 1976.
8. M. C. Fitting. The family of stable models. *Journal of Logic Programming,* 17:197–225, 1993.
9. M. C. Fitting. Fixpoint semantics for logic programming - a survey. *Theoretical Computer Science,* 21(3):25–51, 2002.
10. M. Fitting. A Kripke-Kleene-semantics for general logic programs. *Journal of Logic Programming,* 2:295–312, 1985.
11. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. Bowen, editors, *Proc. of the 5th Int. Conf. on Logic Programming,* pages 1070–1080. The MIT Press, 1988.
12. M. L. Ginsberg. Multi-valued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence,* 4:265–316, 1988.
13. K. Kunen. Negation in logic programming. *Journal of Logic Programming,* 4(4):289–308, 1987.
14. Y. Loyer and U. Straccia. Uncertainty and partial non-uniform assumptions in parametric deductive databases. In *Proc. of the 8th European Conf. on Logics in Artificial Intelligence (JELIA-02),* LNCS 2424, pages 271–282. Springer-Verlag, 2002.
15. Y. Loyer and U. Straccia. The approximate well-founded semantics for logic programs with uncertainty. In *28th Int. Sym. on Mathematical Foundations of Computer Science (MFCS-2003),* LNCS 2747, pages 541–550. Springer-Verlag, 2003.
16. Y. Loyer and U. Straccia. Default knowledge in logic programs with uncertainty. In *Proc. of the 19th Int. Conf. on Logic Programming (ICLP-03),* LNCS 2916, pages 466–480. Springer Verlag, 2003.
17. T. Lukasiewicz. Fixpoint characterizations for many-valued disjunctive logic programs with probabilistic semantics. In *In Proc. of the 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-01),* LNAI 2173, pages 336–350. Springer-Verlag, 2001.
18. R. Ng and V.S. Subrahmanian. Stable model semantics for probabilistic deductive databases. In Z. W. Ras and M. Zemenkova, editors, *Proc. of the 6th Int. Sym. on Methodologies for Intelligent Systems (ISMIS-91),* LNAI 542, pages 163–171. Springer-Verlag, 1991.
19. T. C. Przymusinski. Extended stable semantics for normal and disjunctive programs. In D. H. D. Warren and P. Szeredi, editors, *Proc. of the 7th Int. Conf. on Logic Programming,* pages 459–477. MIT Press, 1990.
20. T. C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae,* 13(4):445–463, 1990.
21. R. Reiter. On closed world data bases. In Hervé Gallaire and Jack Minker, editors, *Logic and data bases,* pages 55–76. Plenum Press, New York, NY, 1978.
22. A. van Gelder, K. A. Ross, and J. S. Schlimpf. The well-founded semantics for general logic programs. *Journal of the ACM,* 38(3):620–650, 1991.

Structural Model Checking for Communicating Hierarchical Machines

Ruggero Lanotte[1], Andrea Maggiolo-Schettini[2], and Adriano Peron[3]

[1] Dipartimento di Scienze della Cultura, Politiche e dell'Informazione, Università dell'Insubria, Via Valleggio 11, Como I-22100, Italy
[2] Dipartimento di Informatica, Università di Pisa, Via Buonarroti 2, I-56127 Pisa, Italy
[3] Dipartimento di Scienze Fisiche, Università di Napoli Federico II, Via Cintia, I-80126 Napoli, Italy

Abstract. In this paper we consider the problem of model checking for well structured Communicating Hierarchical Machines (CHMs), i.e. Finite State Machines with additional features of hierarchy and concurrency. For model checking we follow the automaton theoretic approach, defining an algorithm which solves the problem of model checking without flattening the structure of CHMs.

1 Introduction

The feature of hierarchy together with the ability of representing parallel composition got prominence in Finite State Machines (FSMs for short) with the introduction of the specification formalism Statecharts (see [7]). Hierarchical structuring of states, parallelism and communication among parallel components support modular and top-down development of specifications. The features mentioned have been adopted in many specification formalisms, such as Modecharts and Rsml and are a central component of various object oriented software development methodologies (e.g. see the Unified Modelling Language [4]).

From a theoretical view point the impact of concurrency and communication on FSMs has been studied in [8], where concurrent communicating FSMs are shown to be exponentially more succinct than sequential FSMs. In [2] it has been studied also the impact of hierarchy in state machines. The formalism considered in that paper is called Communicating Hierarchical State Machines (CHMs for short). A CHM is an ordered collection of FSMs with the additional ability of refining a state of an FSM with another FSM and of composing FSMs in parallel. One advantage of this approach (with resp. to Statecharts, for instance) is that one can specify a component only once and use it in different contexts. It has been shown that allowing both hierarchy and concurrency in CHMs gives double exponential succinctness with respect to FSMs. In the same paper it is also shown, however, that the problem of reachability, in the general case, is difficult being EXSPACE-complete, even though, for the subclass of *well structured* CHMs, the complexity of reachability is PSPACE-complete.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 525–536, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The traditional approach adopted to solve reachability problems for FSMs with hierarchy and/or parallelism, is that of flattening structure and applying standard techniques on the corresponding flat sequential FSM (this may lead to a double exponential increase of the representation). Recent papers (see [1, 3]) show that the reachability problem for sequential hierarchical machines can be solved efficiently without flattening hierarchy. In this paper we tackle the problem of model checking for well structured CHMs by following the same approach, i.e. by preserving structure.

For expressing CHMs properties, we introduce a variant of Propositional Linear Temporal Logic (PLTL), called Hierarchical PLTL (HPLTL). It differs from PLTL in the ability of contextualizing in the modular structure of a CHM the standard properties one can express in PLTL. For instance, given a couple of parallel components, one can require that a property ϕ holds on the first component *until* the property ψ holds in the second, or one can require that a property is satisfied by the machine refining the one considered. This allows taking advantage of the benefits of modularity and refinement also for the specification of properties. For verification, we adopt the well known automaton theoretic approach (e.g. see [10]): to prove that any execution of a FSM M satisfies a PLTL formula ϕ, one constructs a FSM (actually a Büchi automaton) M' which accepts all the models for the formula $\neg\phi$, then suitably synchronizes M and M', and finally checks the resulting FSM for emptiness.

We present and exploit the structure of the model to be checked. Instead of flattening a CHM M, into an ordinary sequential machine, we define an algorithm that, given a HPLTL formula (i.e. a formula with explicit reference to hierarchy and modularity), constructs a CHM 'accepting' all the models of the formula. The problem of model checking is then reduced to the problem of emptiness for the CHM obtained by suitably synchronizing a couple of CHMs. Since we consider well structured CHMs, the reachability problem is *PSPACE* complete (see [2]). We prove that, for the acceptance criteria we adopt in this paper, the emptiness problem maintains the same complexity.

The paper is organized as follows: in Section 1 and 2 we recall the syntax and semantics of well structured CHMs; in Section 3 we introduce syntax and semantics of Hierarchical PLTL; in Section 4 we study equivalences and syntactical restrictions for HPLTL; in Section 6 we give the model checking algorithm and we assess its correctness.

2 Communicating Hierarchical State Machines

In this section we provide a (syntactical) variant of the formalism of the subclass of *well structured* CHMs presented in [2]. A CHM is an ordered collection of FSMs augmented with the additional ability of refining a state with a collection of FSMs. Well structured CHMs restrict CHM by allowing communication only between parallel components which refine the same state.

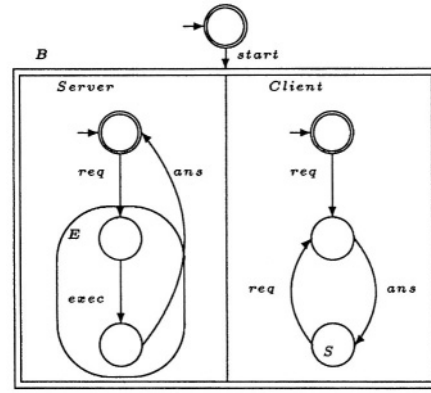
Definition 1. *Let Σ be an alphabet of input symbols. A Communicating Hierarchical Machine (CHM for short) is a collection of* sequential machines $\mathcal{A} = A_1, \ldots, A_n$, *where* A_i *has the form* $\langle N_i, B_i, Y_i, En_i, Ex_i, T_i, F_i \rangle$, *where*

- N_i *is the set of* nodes *and* B_i *is the set of* boxes *(assume* $N_i \cap B_i = \emptyset$*);*
- $Y_i : B_i \to \{A_{i+1}, \ldots, A_n\}^+$ *is the* refinement function;
- $En_i \subseteq N_i$ *and* $Ex_i \subseteq N_i$ *are the* entering *and* exiting *states, respectively;*
- $T_i \subset Src \times \Sigma \times Tgt$, *is the set of* transitions, *where Src is the set of source states* $N_i \cup \bigcup_{b \in B_i} (\{b\} \times Ex_{i_1^b} \times \ldots \times Ex_{i_k^b})$ *and Tgt is the set of target states* $N_i \cup \bigcup_{b \in B_i} (\{b\} \times En_{i_1^b} \times \ldots \times En_{i_k^b})$ *with* $Y_i(b) = A_{i_1^b} \cdot \ldots \cdot A_{i_k^b}$;
- $F_i = \Phi_{i_1}, \ldots, \Phi_{i_k}$, *with* $\Phi_{i_j} \subseteq N_i \cup B_i$, *for* $1 \leq j \leq k$ *is the collection of* acceptance sets.

We assume that $(N_i \cup B_i) \cap (N_j \cap B_j) = \emptyset$, *for all* $1 \leq i < j \leq n$. *The set of states of* $\mathcal{A}$ *is* $\mathcal{S}_\mathcal{A} = \bigcup_{i=1}^{n} B_i \cup N_i$.

The set of states of a sequential machine is partitioned into a set of nodes and a set of boxes. Performing a transition to a box b can be interpreted as a (parallel) procedure call which simultaneously activates a collection of sequential machines (the non-empty list of sequential machines given by the the refinement function Y_i applied to b). Since Y_i gives a list of sequential machine names, a (parallel) procedure call can activate different copies of the same machine. Moreover, the requirement that the machines in the list given by Y_i have indexes greater than i avoids recursive procedure calls. Notice that a transition leading to a box specifies the entering state (initial state) of each activated machine. All the machines, which are simultaneously activated, run (synchronously) in parallel, whereas the calling machine suspends its activity waiting for the return of the (parallel) procedure call. The return of a parallel procedure call to a box b is represented by a transition from b that specifies an exiting state for each sequential machine activated by the procedural call (a synchronous return from all the activated sequential machines). Parallel components which refine the same box behave synchronously: if a parallel component takes a transition labelled by an input symbol a, then all the other components must be able to take a transition labelled by the same input symbol.

Example 1. The figure depicts a simple system composed by a Client and a Server. The Client, after a request to the Server for a service (*req*), waits for the Server to satisfy the request (*ans*). The system is obtained by composition/refinement of four machines which are, in the order: $M_1 = System$ consisting of an entering state and a box *B;* $M_2 = Server$ consisting of an entering state and a box *E;* $M_3 = Client$ consisting of nodes only; and M_4, the sequential



machine inside box E. Notice that the refinement function Y_1 of M_1 is such that $Y_1(B) = M_2 \cdot M_3$ (the parallel composition of *Server* and *Client*).

3 Semantics of *CHMs*

The operational semantics of CHMs is given in terms of Labelled Transition Systems (LTSs) defined in the well know style of Structured Operational Semantics ([9]). From now on, we refer to a *CHM* $\mathcal{A} = A_1, \ldots, A_n$, with A_i of the form $\langle N_i, B_i, Y_i, En_i, Ex_i, T_i, F_i \rangle$. Since boxes can be refined by a parallel composition of FSMs, a configuration is described by a tree-like structure of states.

The set of *configurations* of a sequential machine A_i, denoted by $Conf(A_i)$, is inductively defined as follows:

1. $q \in Conf(A_i)$, for any $q \in N_i$;
2. $b.(c_1, \ldots, c_k) \in Conf(A_i)$, for $b \in B_i$ and $c_j \in Conf(A_{h_j})$ with $Y_i(b) = A_{h_1} \cdot \ldots \cdot A_{h_k}$, $1 \leq j \leq k$.

The set of configurations of $\mathcal{A}$, written $Conf(\mathcal{A})$, is $Conf(A_1)$. A configuration is initial if it has the form q, with $q \in En_1$.

A point of a configuration c is described by a string $s \in \{1, \ldots, n\}^*$: the empty string ϵ represents the root of the configuration, i represents the i-th son of the root (from left to right -if any-), $j.i$ is the i-th son of the j-th son of the root, and so on. The set of points $Pt(c)$ of a configuration c, is defined as follows:

1. $Pt(c) = \{\epsilon\}$ if $c = q$ for a node q;
2. $Pt(c) = \{\epsilon, 1, \ldots, k\} \cup \bigcup_{j \in [1,k]} \bigcup_{s \in Pt(c_j)} \{j \cdot s\}$ if $c = b.(c_1, \ldots c_k)$.

In the definition above we assume that, for each point s, s and $s.\epsilon$ denote the same point. Given a configuration c, the function $state_c : Pt(c) \rightarrow \mathcal{S}_{\mathcal{A}}$, gives the node or the box associated with a point of the configuration, more precisely:

1. $state_c(\epsilon) = q$ if c has either the form q or the form $q.(c_1, \ldots, c_s)$;
2. $state_c(i.w) = state_{c_i}(w)$ if c has the form $q.(c_1, \ldots c_s)$, with $1 \leq i \leq s$.

We can define now the LTS for $\mathcal{A}$. As usual, the states of the LTS are the configurations of $\mathcal{A}$. Each transition is labelled by a pair (W, a), where W is the set of configuration points where actually the transition is performed, and a is a symbol of Σ. So, the LTS for the a CHM $\mathcal{A}$ is the triple $\langle Conf(\mathcal{A}), 2^{Pt_{\mathcal{A}}} \times (\Sigma \cup \{\tau\}), \rightarrow_{\mathcal{A}} \rangle$, where $Pt_{\mathcal{A}}$ is the set of all possible points of configurations in $Conf(\mathcal{A})$ and $\rightarrow_{\mathcal{A}} \subseteq Conf(\mathcal{A}) \times 2^{Pt_{\mathcal{A}}} \times (\Sigma \cup \{\tau\}) \times Conf(\mathcal{A})$ is the transition relation defined by the following set of rules.

$$\text{R.1 } toConf(s) \xrightarrow{(\{\epsilon\}, a)} toConf(s'), \text{ with } \langle s, a, s' \rangle \in T_i \text{ and } a \in \Sigma$$

($toConf(s) = s$ if s is a node and $toConf(s) = b.(q_1, \ldots, q_n)$ if $s = (b, q_1, \ldots, q_n)$)

$$\text{R.2 } \frac{c_1 \xrightarrow{(\{\epsilon\}, a)} c'_1 \ldots c_k \xrightarrow{(\{\epsilon\}, a)} c'_k, \text{ with } a \in \Sigma}{b.(c_1, \ldots, c_k) \xrightarrow{(\{1, \ldots, k\}, \tau)} b.(c'_1, \ldots, c'_k)}$$

$$\text{R.3}\ \frac{c_h \overset{(W,\tau)}{\longrightarrow} c'_h}{b.(c_1,\ldots,c_h,\ldots,c_k) \overset{(\{h\cdot w : w\in W\},\tau)}{\longrightarrow} b.(c_1,\ldots,c'_h,\ldots,c_k)}$$

The Axiom of Rule R.1 deals with the basic case of FSM transitions. Rule R.2 forces transition synchronization in all the FSM refining a box b (notice that the resulting transition is labelled by the internal non-observable symbol τ). Rule R.3 allows a FSM, having as current state a box b refined with k FSMs, to perform a transition internal to the h-th refining FSM. Notice that the transition label keeps track of the position of the component performing the internal transition.

A path for the LTS for $\mathcal{A}$ is a sequence $c_0 \overset{(s_0,a_0)}{\rightarrow} c_1 \overset{(s_1,a_1)}{\rightarrow} \ldots$ such that c_0 is an initial configuration and $\langle c_i,(s_i,a_i),c_{i+1}\rangle \in \rightarrow_{\mathcal{A}}$, for all $i \geq 0$. We introduce a notion of acceptance similar to that adopted by Generalized Büchi Automata (see [10]). Recalling that a machine A_i has a collection of acceptance sets $F_i = \Phi_{i_1},\ldots,\Phi_{i_k}$, we require that any machine A_i, which is active infinite often in a path, has, for each acceptance set, at least one state occurring infinitely often in that path. More precisely, for a path $\mathcal{R}$, let $inf(\mathcal{R}) \subseteq \mathcal{S}_{\mathcal{A}}$ be the set of states which occur infinitely often in $\mathcal{R}$. The path $\mathcal{R}$ is *successful* if, for any $1 \leq i \leq n$, such that $inf(\mathcal{R}) \cap (N_i \cup B_i) \neq \emptyset$ it holds that, for all $1 \leq j \leq k$, $inf(R) \cap \Phi_{i_j} \neq \emptyset$.

In [2] it has been shown that the problem of reachability for well structured CHMs is PSPACE complete. The following result shows that the acceptance condition adopted in this paper for CHMs allows checking the emptiness problem (i.e. whether there exists a successful path) in polynomial space.

Theorem 1. *The problem of emptiness for CHMs is PSPACE.*

4 Hierarchical PLTL

Hierarchical PLTL (HPLTL) is a syntactical extension of the standard Propositional Linear Time Logic. PLTL allows reasoning about infinite sequence of (unstructured) states whereas HPLTL allows reasoning about infinite sequences of tree-like structured states (i.e. configurations of CHMs). HPLTL enriches PLTL with a set of atomic propositions $\Pi = \{Root, Leaf, First, Last\}$ which hold on the root of a configuration, on a leaf, at the first and last component of a parallel composition, respectively. Moreover, once fixed a point in a configuration, there are operators for reasoning about a linear time property ϕ which has to be satisfied at the ancestor of that point (in symbols, $\Uparrow(\phi)$), or at the i-th refining component ($\downarrow_i(\phi)$), or at the left or right parallel component ($\leftarrow(\phi)$ and $\rightarrow(\phi)$, respectively). The operator introduced can be freely composed with standard operators: the operator next $\bigcirc$, the operator until U, and the dual operator $\overline{\mathbf{U}}$.

Definition 2. *Given a set of propositional symbols $\mathcal{P}$, an HPLTL formula is inductively defined as follows:*

- $p, \neg p \in HPLTL$, *for any* $p \in \mathcal{P} \cup \Pi$;
- *True, False, Stop* $\in HPLTL$;
- $\bigcirc\phi \in HPLTL$, *for any* $\phi \in HPLTL$;

- $\downarrow_n(\phi), \Uparrow(\phi), \leftarrow(\phi), \rightarrow(\phi) \in HPLTL$, *for any* $\phi \in HPLTL$ *and* $n \geq 1$;
- $\phi \mathbf{U} \psi, \phi \overline{\mathbf{U}} \psi$ *for any* $\phi, \psi \in HPLTL$.

We shall consider also two derived operators, which are of very common use:

- $\Diamond\phi$ which stands for $True\mathbf{U}\phi$ (ϕ will be true at some point of the future);
- $\Box\phi$ which stands for $False\overline{\mathbf{U}}\phi$ (ϕ will be true at all points of the future).

The semantics of HPLTL formulas is defined with respect to sequences of configurations having nodes labelled over $2^{\mathcal{P} \cup \Pi}$, i.e. a labelled sequence of pairs

$$\mathcal{C} = (c_0, V_0), (c_1, V_1) \ldots, (c_i, V_i) \ldots$$

where c_i is a configuration and $V_i : Pt(c_i) \rightarrow 2^{\mathcal{P} \cup \Pi}$, for each $i \geq 0$. In the following, we shall consider only sequences of configurations corresponding to executions of a CHM $\mathcal{A}$ (i.e. paths of the corresponding LTS), and the labelling of configurations will be induced by a *labelling function* $L_{\mathcal{A}} : \mathcal{S}_{\mathcal{A}} \rightarrow 2^{\mathcal{P}}$.

More precisely, we shall consider sequences of labelled transitions satisfying the two following constraints:

1. $c_0 \stackrel{(s_0,a_0)}{\rightarrow} c_1 \stackrel{(s_1,a_1)}{\rightarrow}, \ldots, c_i, \ldots$ is a path in the LTS of $\mathcal{A}$;
2. $V_i(w) \cap \mathcal{P} = L_{\mathcal{A}}(state_{c_i}(w))$, for all $w \in Pt(c_i)$, for all $i \geq 0$;
3. $Root \in V_i(w)$ iff $w = \epsilon$;
4. $Leaf \in V_i(w)$ iff $w \cdot 1 \notin Pt(c_i)$;
5. $First \in V_i(w)$ iff either $w = \epsilon$ or has the form $w' \cdot 1$;
6. $Last \in V_i(w)$ iff either $w = \epsilon$ or has the form $w' \cdot j$ and $w' \cdot j + 1 \notin Pt(c_i)$.

The labelled sequence induced by a path $\mathcal{R}$ is denoted by $\mathcal{C}_{\mathcal{R}}$.

A crucial point for defining the semantics of an HPLTL formula is determining what is the next state for a given state in a configuration. Given a point $\langle c_i, w \rangle$ of the i-th configuration of the sequence $\mathcal{C}_{\mathcal{R}}$, we could choose a 'global notion' of next state saying that the next state of $\langle c_i, w \rangle$ is $\langle c_{i+1}, w \rangle$, namely the corresponding point (if any) in the $i+1$-th configuration of the sequence. The drawback of this solution is that it might happen that the transition leading from c_i to c_{i+1} does not affect the state at point w (i.e. there is no real change of state at point w). The semantics we have chosen enforces a stronger, 'local' notion of next state: the next point of a configuration state $\langle c_i, w \rangle$ is the first state $\langle c_j, w \rangle$, with $j > i$, which has been affected by a transition in the path considered. Notice that in absence of parallelism the global and local notion of next state coincide.

Given a path $c_0 \stackrel{(s_0,a_0)}{\rightarrow} c_1 \stackrel{(s_1,a_1)}{\rightarrow}, \ldots, c_i, \ldots$ in the LTS of $\mathcal{A}$, and a pair $\langle c_i, w \rangle$ with $i \geq 0$ and $w \in Pt(c_i)$, we denote by $Next(c_i, w)$ the configuration c_j (if any), with $j > i$, such that $w \in s_j$, and, such that, for each $i < h < j$, there is no element $w' \in s_h$ which is a prefix of w.
$Next^*(c_i, w)$ denotes the set of configurations inductively defined as follows:

- $c_j \in Next^*(c_i, w)$ if $c_j = Next(c_i, w)$;
- $c_j \in Next^*(c_i, w)$ if $c_j = Next(c_k, w)$ and $c_k \in Next^*(c_i, w)$.

We give the rules for assigning the truth value of a formula with respect to a point of a configuration of a labelled sequence $\mathcal{C}_{\mathcal{R}}$. The rules are the following:

- $\langle c_i, w\rangle \models True$, and it is not the case that $\langle c_i, w\rangle \models False$, for any c_i and any $w \in Pt(c_i)$;
- $\langle c_i, w\rangle \models p$ iff $p \in V_i(w)$, for any $p \in \mathcal{P} \cup \Pi$;
- $\langle c_i, w\rangle \models \neg p$ iff $p \notin V_i(w)$, for any $p \in \mathcal{P} \cup \Pi$;
- $\langle c_i, w\rangle \models Stop$ iff $Next^*(c_i, w) = \emptyset$;
- $\langle c_i, w\rangle \models \uparrow(\phi)$ iff $w = w' \cdot j$ and $\langle c_i, w'\rangle \models \phi$;
- $\langle c_i, w\rangle \models \downarrow_j(\phi)$ iff $w \cdot j \in Pt(c_i)$ and $\langle c_i, w \cdot j\rangle \models \phi$;
- $\langle c_i, w\rangle \models \leftarrow(\phi)$ iff $w = w'.j$ with $j > 1$ and $\langle c_i, w' \cdot j - 1\rangle \models \phi$;
- $\langle c_i, w\rangle \models \rightarrow(\phi)$ iff $w = w' \cdot j$ and $w' \cdot j + 1 \in Pt(c_i)$ and $\langle c_i, w' \cdot j + 1\rangle \models \phi$;
- $\langle c_i, w\rangle \models \bigcirc(\phi)$ iff there is $j > i$ such that $c_j = Next(c_i, w)$ and $\langle c_j, w\rangle \models \phi$;
- $\langle c_i, w\rangle \models \phi \mathbf{U} \psi$ iff there are $j_1 < \ldots < j_k$ with $i = j_1$ and $k \geq 1$, such that $\langle c_{j_k}, w\rangle \models \psi$, and, for each $h = 1, \ldots, k-1$, it holds that $Next(c_{j_h}, w) = c_{j_{h+1}}$, and $\langle c_{j_h}, w\rangle \models \phi$;
- $\langle c_i, w\rangle \models \phi \overline{\mathbf{U}} \psi$ iff for all $c_j \in Next^*(c_i, w)$, if it is not the case that $\langle c_j, w\rangle \models \psi$, then there exists $c_h \in Next^*(c_i, w)$ with $i \leq h < j$ such that $\langle c_h, w\rangle \models \phi$.

Given a labelled sequence $\mathcal{C}_{\mathcal{R}} = c_0 \overset{(s_0,a_0)}{\rightarrow} c_1 \overset{(s_1,a_1)}{\rightarrow} \ldots$, we say that $\mathcal{C}_{\mathcal{R}}$ *satisfies* a formula ϕ, written $\mathcal{C}_{\mathcal{R}} \models \phi$, if $\langle c_0, \epsilon\rangle \models \phi$. Moreover a CHM $\mathcal{A}$ *satisfies* a formula ϕ, written $\mathcal{A} \models \phi$, iff for any path $\mathcal{R}$ of the LTS for $\mathcal{A}$ it holds that $\mathcal{C}_{\mathcal{R}} \models \phi$.

In the logic we have defined, negation is only applied to atomic propositions in $\mathcal{P} \cup \Pi$. Assuming that $\langle c_i, s\rangle \models \neg\phi$ iff it is not the case that $\langle c_i, s\rangle \models \phi$, the syntactical restriction can be lifted by observing that

- $\langle c_i, s\rangle \models \neg(\phi \mathbf{U} \psi)$ iff $\langle c_i, s\rangle \models \neg\phi \overline{\mathbf{U}} \neg\psi$;
- $\langle c_i, s\rangle \models \neg(\phi \overline{\mathbf{U}} \psi)$ iff $\langle c_i, s\rangle \models \neg\phi \mathbf{U} \neg\psi$;
- $\langle c_i, s\rangle \models \neg \bigcirc(\phi)$ iff $\langle c_i, s\rangle \models \bigcirc(\neg\phi) \vee Stop$;
- $\langle c_i, s\rangle \models \neg Stop$ iff $\langle c_i, s\rangle \models \bigcirc(True)$;
- $\langle c_i, s\rangle \models \neg\uparrow(\phi)$ iff $\langle c_i, s\rangle \models \uparrow(\neg\phi) \vee Root$;
- $\langle c_i, s\rangle \models \neg\downarrow_j(\phi)$ iff $\langle c_i, s\rangle \models \downarrow_j(\neg\phi) \vee Leaf \vee \bigvee_{l=1}^{j-1}(\downarrow_l(True) \wedge Last)$;
- $\langle c_i, s\rangle \models \neg\leftarrow(\phi)$ iff $\langle c_i, s\rangle \models \leftarrow(\neg\phi) \vee First$;
- $\langle c_i, s\rangle \models \neg\rightarrow(\phi)$ iff $\langle c_i, s\rangle \models \rightarrow(\neg\phi) \vee Last$.

Example 2. The system of Example 1 satisfies the following properties:

- $\bigcirc(\downarrow_1 (\Box(\downarrow_1 (\Diamond Stop) \Rightarrow\rightarrow (\bigcirc(S)))))$, namely, it always holds that, if the service is completed by the Server, then eventually Client receives the answer;
- $\bigcirc(\downarrow_1 (\Box(E \Rightarrow \Diamond(\downarrow_1 (Stop)))))$, namely, it holds that, whenever the Server is going to execute a service it will be completed;
- $\bigcirc(\downarrow_1 (\Box(\neg Leaf \Rightarrow (\Diamond(Leaf \wedge \Diamond \downarrow_1 (Stop))))))$, namely, it always holds that the Server satisfies infinitely many times the service requested.

5 Equivalences of Formulas and Syntactical Restrictions

In this section we introduce some equivalences of HPLTL formulas and a syntactical restriction of HPLTL forcing to express properties of a CHM in a top-down (with respect to refinement) and left-to-right (with respect to parallelism) manner. The following proposition is a consequence of the HPLTL semantics.

Proposition 1. *The following equivalences hold true:*

1. $\bigcirc$, $\leftarrow$, $\rightarrow$, $\downarrow_i$ *and* $\uparrow$ *distribute with respect to both the connectives* $\vee$ *and* $\wedge$;
2. $\bigcirc$ *commutes with respect to* $\rightarrow$ *and* $\leftarrow$;
3. $\rightarrow(\leftarrow(\phi)) \equiv \phi \wedge \rightarrow(True)$ *and* $\leftarrow(\rightarrow(\phi)) \equiv \phi \wedge \leftarrow(True)$;
4. $\rightarrow(\uparrow(\phi)) \equiv \uparrow(\phi) \wedge \rightarrow(True)$ *and* $\leftarrow(\uparrow(\phi)) \equiv \uparrow(\phi) \wedge \leftarrow(True)$;
5. $\downarrow_i(\uparrow(\phi)) \equiv \phi \wedge \downarrow_i(True)$ *and* $\uparrow(\downarrow_i(\phi)) \equiv \phi \wedge \uparrow(True)$;
6. $\bigcirc(\uparrow(\phi)) \equiv \bigcirc(True) \wedge \uparrow(\phi)$;
7. $\downarrow_i(\rightarrow(\phi)) \equiv \downarrow_{i+1}(\phi)$ *and* $\downarrow_i(\phi) \equiv \downarrow_1(\rightarrow_i(\phi))$;
8. $\rightarrow$ *and* $\leftarrow$ *commute with respect to* $\mathbf{U}$ *and* $\overline{\mathbf{U}}$ *(for instance,* $\rightarrow(\phi_1 \mathbf{U} \phi_2) \equiv \rightarrow(\phi_1)\mathbf{U}\rightarrow(\phi_2)$*),*

where $\rightarrow_i(\phi)$ *is the formula* $\rightarrow(\phi)$ *if* $i = 1$, *and* $\rightarrow(\rightarrow_{i-1}(\psi))$, *otherwise.*

Notice that $\uparrow$ does not commute with respect to $\mathbf{U}$ and $\overline{\mathbf{U}}$. Moreover, if there is no occurrence of $\downarrow_i$ in ϕ_1 (resp. ϕ_2) we have

$$(\uparrow(\phi_1) \vee \phi_2)\mathbf{U}\phi_3 \equiv (\uparrow(\phi_1) \wedge (True\mathbf{U}\phi_3)) \vee (\phi_2\mathbf{U}\phi_3) \tag{1}$$

$$(\text{resp.: } \phi_1\mathbf{U}(\uparrow(\phi_2)) \wedge \phi_3 \equiv \uparrow(\phi_2) \wedge (\phi_1\mathbf{U}\phi_3)). \tag{2}$$

The equivalences $\uparrow(\phi_1)\mathbf{U}\phi_2 \equiv (\uparrow(\phi_1) \wedge (True\mathbf{U}\phi_2)) \vee \phi_2$ and $\phi_1\mathbf{U}\uparrow(\phi_2) \equiv \uparrow(\phi_2)$ are particular case of Eq.1 and Eq.2. $\overline{\mathbf{U}}$ enjoys analogous equivalences.

It is easy to verify that, if a formula ϕ fulfils the requirement that there is no occurrence of $\downarrow_i$, $\leftarrow$ and $\rightarrow$ in the scope of $\uparrow$, then ϕ can be rewritten into a formula ϕ' in a so called *restricted form,* where one is enabled to speak about properties of hierarchical machines only in a top-down (as concerns hierarchy) and left-to-right (as concerns parallelism) direction.

Definition 3. *A formula* ϕ *is in* restricted form *if it holds that:*

1. *There is no occurrence of* $\uparrow$, *of* $\downarrow_i$ *with* $i > 1$, *and of* $\leftarrow$;
2. *Each occurrence of a formula* $\rightarrow_i(\psi)$ *in* ϕ *in the scope of a formula* $\rightarrow_j(\psi')$ *is in the scope of the operator* $\downarrow_1$.

Proposition 2. *For any formula* ϕ *having no occurrence of* $\downarrow_i$, $\leftarrow$ *and* $\rightarrow$ *in the scope of* $\uparrow$, *there is a formula* ϕ' *in restricted form equivalent to* ϕ.

6 From Hierarchical PLTL to CHMs

In this section we shall synthesize CHMs for the fragment of formulas in restricted form. The problem we consider is the following: given an HPLTL formula ϕ, construct a CHM whose successful paths are exactly those that satisfy ϕ. The construction we propose in this section is modelled on the well known construction for PLTL formulas (see [10] for a tutorial). Even if, in the case of PLTL many improvements can be found in the literature for the basic technique (e.g., removal of redundancies, construction of the automaton by need etc), in this paper, for simplicity, we present only an extension of the basic technique leaving possible optimizations to future work.

The idea is that, given a formula ϕ, we start looking for the linear properties expressed by ϕ concerning only the local sequential module the formula is concerned with. For instance, if we consider the formula $\bigcirc(\downarrow_1 (\Box(\downarrow_1 (\Diamond Stop) \Rightarrow \rightarrow (\bigcirc(S)))))$ discussed in the example 2, this can be seen as an atomic formula $\bigcirc(P)$, since it essentially speaks of the properties of a refinement of the current module. Therefore, we start with constructing a sequential component M_1 for the formula P. To construct a sequential component we exploit a simple variant of the standard algorithm. The state of M_1 where the atomic formula P holds has to be refined with a CHMs for the formula $\Box(\downarrow_1 (\Diamond Stop) \Rightarrow \rightarrow (\bigcirc(S)))$. By applying the same idea, we have that the local linear property has then the form $\Box(Q \Rightarrow R)$, and hence we create a machine M_2 for that formula. Since R has the form $\rightarrow (\bigcirc(S))$ we create a machine M_3 for $\bigcirc(S)$ which runs in parallel synchronously with M_2 and which communicates with M_2 when the property $\bigcirc(S)$ holds. Therefore, in M_2 the atomic predicate R can be assumed to hold only when a synchronization with M_3 communicates $\bigcirc(S)$.

We define now the *closure* of ϕ, written $cl(\phi)$, i.e. the subset of subformulas of ϕ which are required in order to synthesize the sequential machine for ϕ:

- $\phi \in cl(\phi)$;
- $\phi_1 \wedge \phi_2 \in cl(\phi)$ implies $\phi_1, \phi_2 \in cl(\phi)$;
- $\phi_1 \vee \phi_2 \in cl(\phi)$ implies $\phi_1, \phi_2 \in cl(\phi)$;
- $\bigcirc(\phi_1) \in cl(\phi)$ implies $\phi_1 \in cl(\phi)$;
- $\phi_1 \mathbf{U} \phi_2 \in cl(\phi)$ implies $\phi_1, \phi_2 \in cl(\phi)$;
- $\phi_1 \overline{\mathbf{U}} \phi_2 \in cl(\phi)$ implies $\phi_1, \phi_2 \in cl(\phi)$.

We denote by $Comp_{\rightarrow}(\phi)$ the set $\{\phi' : \phi' \in cl(\phi), \phi'$ has the form $\rightarrow(\phi_1)\}$; by $Comp_{\rightarrow_i}(\phi)$ the set $\{\rightarrow_i (\phi') : \rightarrow_i (\phi') \in Comp_{\rightarrow}(\phi)\}$; by $Comp_{\downarrow_1}(\phi)$ the set $\{\phi' : \phi' \in cl(\phi), \phi'$ has the form $\downarrow_1(\phi_1)\}$. Notice that subformulas of ϕ belonging to $Comp_{\rightarrow_i}(\phi) \cup Comp_{\downarrow_1}(\phi)$ are considered as indivisible formulas. Subformulas in $cl(\phi)$ will be exploited for synthesizing a sequential component of a hierarchical machine. The other sequential components will be synthesized from the set of formulas belonging to $Comp_{\rightarrow}(\phi) \cup Comp_{\downarrow_1}(\phi)$. We construct now a finite state machine $A_\phi = \langle V_\phi, V_{0,\phi}, \delta_\phi, \mathcal{F}_\phi \rangle$ for a formula ϕ. Following the standard construction (see [10]), the idea is to use $2^{cl(\phi)}$ as state set (the transition relation over states is not labeled).

More precisely the automaton for ϕ is $A_\phi = \langle V_\phi, V_{0,\phi}, \delta_\phi, \mathcal{F}_\phi \rangle$ with:

- $V_\phi \subseteq 2^{cl(\phi)}$ is the set of elements $s \in 2^{cl(\phi)}$ such that:
  - $False \notin s$ and $True \in s$;
  - if $\phi_1 \wedge \phi_2 \in s$, then $\phi_1, \phi_2 \in s$;
  - if $\phi_1 \vee \phi_2 \in s$, then either $\phi_1 \in s$ or $\phi_2 \in s$;
  - p and $\neg p$ are mutually exclusive, for all $p \in \mathcal{P} \cup \Pi$;
  - If $Stop \in s$, then $\bigcirc(\phi_1) \notin s$, for any formula ϕ_1;
  - If $Stop \in s$ and $\phi_1 \mathbf{U} \phi_2 \in s$, then $\phi_2 \in s$.
- $V_{0,\phi} = \{s \in V_\phi : \phi \in s\}$;
- $\delta_\phi \subseteq V_\phi \times V_\phi$ is the least set of couples of the form $\langle s, s' \rangle$ such that:
  - $Stop \notin s$;
  - $\bigcirc(\phi') \in s$ implies $\phi' \in s'$;
  - $\phi_1 \mathbf{U} \phi_2 \in s$ implies either $\phi_2 \in s$, or $\phi_1 \in s$ and $\phi_1 \mathbf{U} \phi_2 \in s'$;
  - $\phi_1 \overline{\mathbf{U}} \phi_2 \in s$ implies that either $\phi_2 \in s$ or $\phi_1 \in s$ and $\phi_1 \overline{\mathbf{U}} \phi_2 \in s'$.
- $\mathcal{F}_\phi = \{\Phi_1, \ldots, \Phi_m\}$ where the eventualities appearing in $cl(\phi)$ are $e_1(\phi_1), \ldots, e_m(\phi_m)$ (with $e_i(\phi_i)$ a formula of the form $\phi' \mathbf{U} \phi_i$, for some formula ϕ') and $\Phi_i = \{s \in V_\phi :$ either $e_i, \phi_i \in s$ or $e_i \notin s\}$.

We define now a recursive procedure *Cst* which takes as input a formula ϕ and returns a CHM for ϕ.
$Cst(\phi)$ is recursively defined as follows: If $m = max\{i : \rightarrow_i(\psi) \in Comp_\rightarrow(\phi)\}$, then $Cst(\phi) = M_1 \cdot M_2 \cdot \ldots \cdot M_m$, where $\phi_1 = \phi$ and

$$\phi_i = \Box(True \vee \bigvee_{\rightarrow_i(\psi) \in Comp_{\rightarrow_i}(\phi)} \psi), \text{ with } 2 \leq i \leq m,$$

and $M_i = \langle N_i, B_i, Y_i, En_i, Ex_i, T_i, F_i \rangle$ is the sequential machine induced by $A_{\phi_i} = \langle V_{\phi_i}, V_{0,\phi_i}, \delta_{\phi_i}, \mathcal{F}_{\phi_i} \rangle$:

- $N_i = \{s \in V_{\phi_i} : s \cap Comp_{\downarrow_1}(\phi_i) = \emptyset\}$, $B_i = \{s \in V_{\phi_i} : s \cap Comp_{\downarrow_1}(\phi_i) \neq \emptyset\}$;
- $Y_i = \{\langle s, Cst(\psi_1 \wedge \ldots \wedge \psi_n) \rangle : s \in B, s \cap Comp_{\downarrow_1}(\phi_i) = \{\downarrow_1(\psi_1), \ldots, \downarrow_1(\psi_n)\}\}$;
- $En_i = V_{0,\phi_i} \cap N_i$ and $Ex_i = \{s \in V_{\phi_i} : Stop \in s\} \cap N_i$;
- $T_1 = \{\langle \overline{s}, (\overline{s} \cap Comp_\rightarrow(\phi)), \overline{s}' \rangle : \langle s, s' \rangle \in \delta_\phi, \overline{s} \in R_1(s), \overline{s}' \in R_2(s')\}$, and for $i > 1$, $T_i = \{\langle \overline{s}, \{\rightarrow_j(\psi) \in Comp_\rightarrow(\phi) : j = i, \psi \in s \text{ or } j \neq i\}, \overline{s}' \rangle : \langle s, s' \rangle \in \delta_\phi, \overline{s} \in R_1(s), \overline{s}' \in R_2(s')\}$ where, for any $n \in N_i$, it holds that $R_1(n) = R_2(n) = \{n\}$ and, for any $b \in B_i$ with $Y(b) = M'_1 \cdot \ldots \cdot M'_l$, it holds that $R_1(b) = \{(b, q_1, \ldots, q_l) : q_j \in Ex_{M'_j}$ and $\rightarrow_j(\psi) \in q_1$ implies $\psi \in q_j$, for any $1 \leq j \leq l\}$, and, $R_2(b) = \{b\} \times En_{M'_1} \times \ldots \times En_{M'_l}$;
- $F_i = \mathcal{F}_{\phi_i}$;

The following theorem proves the correctness of the construction.

Theorem 2. *For any CHM $\mathcal{A}$ with labelling function $L_\mathcal{A}$, there exists a path $\mathcal{R}$ such that $\mathcal{C}_\mathcal{R} \models \phi$ iff there exists a successful path $\mathcal{Q}$ of CHM_ϕ with initial configuration r_0 such that $\phi \in state_{r_0}(\epsilon)$.*
Moreover, assuming that $\mathcal{C}_\mathcal{R}$ has the form $\langle c_0, V_0 \rangle, \langle c_1, V_1 \rangle, \ldots$ and that $\mathcal{C}_\mathcal{Q}$ has the form $\langle r_0, V'_0 \rangle, \langle r_1, V'_1 \rangle, \ldots$ it holds that:

1. for any $w \in Pt(c_i) \cap Pt(r_i)$, $V_i(w) = state_{r_i}(w) \cap (\mathcal{P} \cup \Pi)$;

2. *for any* $w \in Pt(r_i) \backslash Pt(c_i)$, w *has the form* $w' \cdot j \cdot w''$ *with* $j \geq 1$, $w' \cdot 1 \in Pt(c_i)$ *and* $\rightarrow_j (\phi) \notin state_{r_i}(w' \cdot 1)$.

We give now an evaluation of the size of CHM_ϕ. Let $G_0(\phi) = \{\phi\}$ and $G_i(\phi) = \{\psi : \rightarrow_i (\psi) \in Comp_{\rightarrow}(\phi)\}$, for $i \geq 1$. Let $m(\phi)$ be the least $i \in \mathbb{N}$ such that $G_j(\phi) = \emptyset$, for any $j > i$ (number of component of this level). We have that

$$size(CHM_\phi) 2^{|Comp_{\rightarrow}(\phi)|} \cdot \sum_{i=1}^{m(\phi)} 2^{|cl(G_i(\phi))|} + size(CHM_{\left(\bigwedge_{\downarrow_1(\psi) \in Comp_{\downarrow_1} G_i(\phi)} \psi\right)}).$$

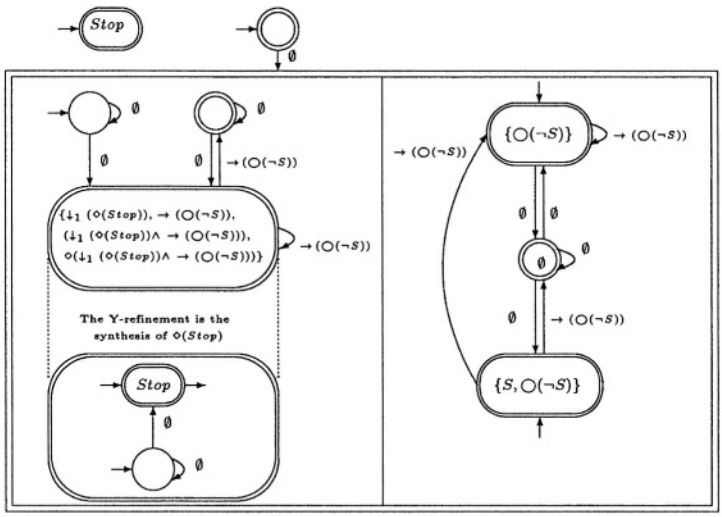


Fig. 1. $Cst(\neg\phi)$

Example 3. We consider now the synthesis of the formula $\neg\phi$, where ϕ is the formula

$$\bigcirc(\downarrow_1 (\Box(\downarrow_1 (\Diamond Stop) \Rightarrow \rightarrow (\bigcirc(S)))))$$

discussed in the example 1. Hence, $\neg\phi$ is equivalent to the formula

$$Stop \vee \bigcirc(\downarrow_1 (\Diamond(\downarrow_1 (\Diamond Stop) \wedge \rightarrow (\bigcirc(\neg S))))).$$

In figure 1 is depicted the CHM $Cst(\neg\phi)$. We have merged some states for readability purposes. The first CHM synthesizes the formula $\Diamond(\downarrow_1 (\Diamond(Stop)) \wedge \rightarrow (\bigcirc(\neg S)))$ where $\downarrow_1 (\Diamond Stop)$ and $\rightarrow (\bigcirc(\neg S))$ are considered as atomic propositions. The CHM synthesizing the formula $\Diamond Stop$ refines the states having an occurrence of $\downarrow_1 (\Diamond(Stop))$. Moreover, the truth of atomic proposition $\rightarrow (\bigcirc(\neg S))$ is communicated by the CHM synthesizing $\bigcirc(\neg S)$ (namely, if $\bigcirc(\neg S)$ is true, then we have a synchronization with the symbol $\rightarrow (\bigcirc(\neg S))$).

In order to complete the algorithm of model checking we have to introduce a notion of synchronous product between a CHM $\mathcal{A}$ and the CHM for a formula ϕ. This product, denoted with $Prd(\mathcal{A}, CHM_\phi)$, exploits the cartesian product between components of $\mathcal{A}$ and CHM_ϕ with additional checks aimed at verifying that states that are synchronized have compatible structure. The formal definition of the synchronous product is skipped here due to paper length constraints and can be found at http://www.di.unipi.it/~lanotte/pub.html in the extended version of the paper. The following theorem states the correctness of our model checking approach.

Theorem 3. $\mathcal{A} \models \phi$ *iff* $Prd(\mathcal{A}, CHM_{\neg\phi})$ *has no successful path.*

The first step of the future work will be devoted to precisely compare, from the complexity point of view, our structural approach with the traditional approach which uses flat FSMs. The emptiness problem can be checked in linear time in flat FSMs, whereas it requires exponential time (but PSPACE complete) for CHMs. Conversely, CHMs (and then also the CHM for a formula ϕ) may be double exponential more succinct than the corresponding flat FSMs (the synchronous product of two CHMs has polynomial size with respect to the sizes of the two operands). We believe (and we want to prove) that our structural approach, taking advantage of double exponential saving of succinctness of CHMs (both for the specification machine and also for the CHM for $\neg\phi$), gives a computational advantage with respect to the traditional approach.

References

1. R. Alur, M. Yannakakis, Model checking of hierarchical state machines. In Proc. 6th ACM Symp. on Found of Software Engineering, 1998, 175–188.
2. R. Alur, S. Kannan, and M. Yannakakis, Communicating hierarchical state machine, In ICALP'99, Springer LNCS 1644, 1999, 169–178.
3. R. Alur, K. Etessami, and M. Yannakakis, Analysis of Recursive State Machines. In Proc. Computer Aided Verification (CAV'01)6, Springer LNCS pp. 207-220.
4. G. Booch, L. Jacobson, and J. Rumbaugh, Unifying Modeling Language user guide, Addison-Wesley, Boston, 1997.
5. A. Boujjani, J. Esparza, and O. Maler, Reachability Analysis of Pushdown automata: Applications to Model Checking. In Proc. Concur'97, Springer LNCS, 1997, pp. 135–150.
6. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon, Efficient algorithms for model checking pushdown systems. In Proc. Computer Aided Verification (CAV'00), Springer LNCS 1855, 2000, 232–247.
7. D. Harel, Statecharts: A visual formalism for complex systems, Science of Computer Programming **8**, 1987, 231–274.
8. D. Drusinsky, D. Harel, On the power of bounded concurrency I: finite automata, Journal of ACM, 41(3), 1994, pp.517–539.
9. G. Plotkin, A structural approach to operational semantics, Technical report DAIMI FN–19, University of Aarhus, Denmark, 1981.
10. P. Wolper, Cstucting automata from temporal logic formulas: A tutorial, Lectures on formal methods and performance analysis: first EEF/Euro Summer School on Trends in Computer Science, Springer, 2002, pp. 261–277.

Compositional Verification: Decidability Issues Using Graph Substitutions

Olivier Ly

LaBRI — Bordeaux I University

Abstract. This paper deals with the *compositional verification* of sequential programs. This consists in deciding whether or not a given set of local structural properties of the functions of a program implies a given global behavioural property of the program. Here we consider properties expressed in monadic second-order logic dealing with the control flow of the program and the function calls occuring during its execution. This problem has been investigated in relation with the security of open multi-application smart cards. We show that the compositionality is *decidable* for sequential programs whose control-flow graphs are of tree-width less than a fixed integer value, which includes in particular structured programs. Formally, we prove the decidability of MSO theories of families of hypergraphs obtained by uniform substitution of hyperedges by hypergraphs specified by MSO formulas.

Keywords: Compositional Verification, Tree Automata, Monadic Second-Order Logic

Introduction

This paper deals with the *compositional verification* of sequential programs. This consists in deciding whether or not a given set of local structural properties of the functions of a program implies a given global behavioural property of the program. This aims at reducing the verification of a global property of a program to the verifications of some independent local properties of its components.

Compositional verification is a long standing problem in the area of concurrent systems (see *e.g.* [1,2,3]). Here we rather consider sequential programs and we restrict our study to behavioural and structural properties dealing with the control flow of the program and the function calls occurring during execution. This framework relies on the language-independent model of programs introduced in [4] in order to catch some classical security properties. It has been studied in [5,6,7] (see also [8]) for compositional reasoning about security of open multi-application smart cards. Like *e.g.* [9,10,11] for concurrent systems, a proof system dealing with modal μ-calculus has been set up in [5] for the framework that we consider. But the question of decidability was left open. Then a decision procedure has been proposed in [7] by restricting the study to properties expressed in simulation logic. Here we consider properties expressed in *monadic*

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 537–549, 2004.
© Springer-Verlag Berlin Heidelberg 2004

second-order logic (see *e.g.* [12]) and we prove that for any fixed integer value k, the compositionality is a *decidable* problem for sequential programs whose control-flow graphs are of tree-width less than k.

This contributes to the studies mentioned above seeing that monadic second-order logic contains modal μ-calculus and *a fortiori* simulation logic. The limitation of our solution concerns the tree-width of the control-flow graphs. Roughly speaking, the tree-width is an integer value measuring how far a graph is from being a tree (see [13]). We claim that this limitation is reasonable, because reasonable programs are indeed of bounded tree-width as it has been shown in [14]: the tree-width of the control-flow graph of any goto-free C program is at most 6. Moreover, a study reported in [15] showed that the tree-width for control-flow graphs of the functions of the Java API source code does not exceed 5, and the average of it is only 2.7.

All the paper is devoted to the proof of the decidability result (Theorem 1), which relies on a variation of the classical link between monadic second-order logic and *automata* (see *e.g.* [16]). First, we remark that the compositionality problem as described above can be reformulated into the satisfiability problem for families of infinite hypergraphs (actually HR-equational hypergraphs) obtained by uniform substitution of hyperedges by hypergraphs specified by MSO formulas, which we call *MSO uniform expansions.* These families actually are the sets of transition systems which encode the behaviours of the programs made of functions satisfying a given set of structural properties. Second, we prove the decidability of MSO theories of MSO uniform expansions in the case of bounded tree-width. The first step of the proof is to translate the problem into a tree language problem, introducing for that a new kind of automata on infinite trees called *composite automata.* These automata encode MSO uniform expansions via the the concept of syntactic expressions, which is possible because of the bounded tree-width. Finally we prove the decidability of the emptyness of the intersection of the languages of a composite automaton and a Rabin automaton. We do that by using a variation of the classical point view of runs of automata in terms of *games* (see *e.g.* [16]). This result allows us to give out a bound for the length of solutions, and therefore to solve the problem by an enumeration method.

1 Formal Definitions

Hypergraphs. A *hypergraph,* also called *relational structure,* is a tuple of the form $(V, L, \{R_\ell\}_{\ell \in L})$ where V is the set of vertices; L is the finite set of labels, and for each $\ell \in L$, R_ℓ is a relation over V, i.e., a subset of V^n for some strictly positive integer n. An element of R_ℓ of form $(v_1, ..., v_n)$ defines a *hyperedge* whose vertex list is $v_1, ..., v_n$; n is called the arity of the hyperedge. Let us note that the labelling of vertices is encoded by hyperedges of arity 1. *Graphs* are the hypergraphs whose hyperedges are all of arity 1 or 2. Here we actually deal with hypergraphs with *sources* which generalise pointed graphs; such a hypergraph

is a hypergraph provided with a finite ordered list of pairwise distinct vertices. These distinguished vertices are called the sources of the hypergraph, the other ones are said to be internal. The sequence of sources of a hypergraph H is denoted by $src(H)$. The number of sources of a hypergraph is called its *type*. Let G be a hypergraph. A *tree-decomposition* is a pair (U, f) where U is an undirected tree and $f : V_U \to 2^{V_G}$ such that: $V_G = \bigcup_{i \in V_U} f(i)$; if a list of vertices $v_1 \dots v_n$ of G are connected by a hyperedge, then there exists $i \in V_U$ such that $v_1, \dots, v_n \in f(i)$; if $i, j, k \in V_U$ are such that j belongs to the shortest path between i and k, then $f(i) \cap f(k) \subseteq f(j)$. The *tree-width* (see [13]) of such a decomposition is defined to be $\max\{\mathrm{card}(f(i)) \mid i \in V_U\} - 1$. The *tree-width* of G, denoted by $twd(G)$, is the minimum of the tree-widths of all its tree-decompositions.

HR-equational Graphs. The concept of *HR-equational hypergraph* extends those of context-free graphs (see [17]). It has been introduced in [18]. Such a hypergraph, possibly infinite, is defined according to a deterministic *hyperedge replacement graph grammar* (see [19]). Such a grammar is defined according to a finite set of symbols L which is divided into two subsets L_1 and L_2 whose elements are respectively called *terminal* symbols and *non-terminal* symbols. A deterministic hyperedge replacement grammar[1] is a tuple $(\ell_0, \{H_\ell\}_{\ell \in L_2})$ where $\ell_0 \in L_2$ is the initial symbol and for each $\ell \in L_2$, H_ℓ is a finite L-labelled hypergraph with source, such that for each $\ell \in L$, the arities of all the ℓ-labelled hyperedges of any $H_{\ell'}$ are equal; and if $\ell \in L_2$, this arity must be equal to the type of H_ℓ. The construction of the HR-equational hypergraph generated by such a grammar is as follows: First, one starts from H_{ℓ_0}. Then, according to the grammar, one replaces each hyperedge labelled by a non-terminal symbol by the associated hypergraph, gluing the sources of the hypergraph in place of the vertices of the hyperedge. One gets a new hypergraph, with possibly new hyperedges labelled by a non-terminal symbols. One replaces again these last ones as above, and so on, possibly infinitely many times, until there is no more such hyperedge.

Graph Grammar Compositionality Problem. We consider *monadic second-order formulæ* on hypergraphs (MSO-formulæ for short, see *e.g.* [12]). These formulæ are constructed using individual variables and set variables, taking their values in the set of vertices. Atomic formulæ are of the forms $x \in A$, $A \subset B$, or else $\mathbf{edg}_\eta(x_1, \dots, x_n)$ which encodes the fact that $x_1, \dots, x_n$ are connected by an η-labelled hyperedge. Syntax is not restricted: we allow existential and universal quantifiers over individual and set variables, conjunctions and negations. The set of finite models (respectively infinite) of a formula φ is denoted by $\mathcal{M}(\varphi)$ (respectively $\mathcal{M}^\infty(\varphi)$). We shall also use the set $\mathcal{M}_{\mathrm{twd}<k}(\varphi)$ (respectively $\mathcal{M}^\infty_{\mathrm{twd}<k}(\varphi)$) of finite (respectively infinite) models of φ

[1] Seeing that the grammars that we consider are deterministic, i.e., it associates to each non-terminal symbol one and only one hypergraph, a production of form $\ell \to H_\ell$ is just denoted by H_ℓ.

whose tree-width is less than a fixed value k. The MSO-theory of a family $\mathfrak{F}$ of hypergraphs, i.e., the set of MSO-formulæ which are satisfied by all the graphs of $\mathfrak{F}$, shall be denoted by $\mathcal{T}h_{\text{MSO}}(\mathfrak{F})$. Let us consider a finite set of symbols $\mathcal{M}$ given with an *arity* mapping $\alpha : \mathcal{M} \to \mathbb{N}$. Let us call *$\mathcal{M}$-vector of typed-graph languages* any tuple $(\mathfrak{L}_m)_{m\in\mathcal{M}}$ of sets of finite graphs such that for each $m \in \mathcal{M}$, all the graphs of $\mathfrak{L}_m$ are of type $\alpha(m)$. Any pair $\mathfrak{S} = (m_0, (\mathfrak{L}_m)_{m\in\mathcal{M}})$ made of a distinguished symbol $m_0 \in \mathcal{M}$ and a $\mathcal{M}$-vector of typed-graph languages $(\mathfrak{L}_m)_{m\in\mathcal{M}}$ defines a family of deterministic graph grammars as follows: $\mathcal{G}r(\mathfrak{S}) = \{(m_0, \{g_m\}_{m\in\mathcal{M}}) \mid \forall m \in \mathcal{M} : g_m \in \mathfrak{L}_m\}$. Let us now consider the set of all the HR-equational hypergraphs generated by any such graph grammar:

$$\mathcal{E}xp(\mathfrak{S}) = \{G \mid \text{there exists } (m_0, \{g_m\}_{m\in\mathcal{M}}) \in \mathcal{G}r(\mathfrak{S}) \text{ which generates } G\}$$

Informally, a graph of $\mathcal{E}xp(\mathfrak{S})$ is obtained by picking up one graph g_m in each $\mathfrak{L}_m$, once for all, and gluing the g_m's to each others infinitely many times. Such a graph shall be said to be obtained by *uniform expansion* from $\mathfrak{S}$ and the family $\mathcal{E}xp(\mathfrak{S})$ shall be called the uniform expansion of $\mathfrak{S}$. *The compositionality problem consists in determining the theory of $\mathcal{E}xp(\mathfrak{S})$ from the theories of the $\mathfrak{L}_m$'s.*

The $\mathfrak{L}_m$'s are *a priori* infinite[2]. A way to give a constructive version of the problem is to consider the case where $\mathcal{M}$-vectors of typed-graph languages are defined by logical formulæ, i.e., $\mathcal{M}$-vectors of the form $(\mathcal{M}(\varphi_m))_{m\in\mathcal{M}}$, where the φ_m's are MSO-formulæ. The family of hypergraphs obtained by uniform expansion from such a $\mathcal{M}$-vector shall be called the *MSO uniform expansion* of the $\mathcal{M}$-vector $(\varphi_m)_{m\in\mathcal{M}}$ according to m_0, it shall be denoted by $\mathcal{E}xp(m_0, (\varphi_m)_{m\in\mathcal{M}})$. It is easy to see that in general theories of MSO uniform expansions are not computable. However, if we bound the tree-width of the considered graphs, i.e., if we consider $\mathcal{M}$-vectors of the form $(\mathcal{M}_{\text{twd}<k}(\varphi_m))_{m\in\mathcal{M}}$ for some fixed k, one gets uniform expansions of form $\mathcal{E}xp(m_0, (\mathcal{M}_{\text{twd}<k}(\varphi_m))_{m\in\mathcal{M}})$, called *bounded-tree-width MSO uniform expansions,* whose MSO-theories are decidable; this is the main point of this paper:

Theorem 1. *Let $\mathcal{M}$ be a set of symbols, $m_0 \in \mathcal{M}$, k a positive integer and $(\varphi_m)_{m\in\mathcal{M}}$ an $\mathcal{M}$-vector of MSO-formulæ,*
then $\mathcal{T}h_{MSO}(\mathcal{E}xp(m_0, (\mathcal{M}_{\text{twd}<k}(\varphi_m)_{m\in\mathcal{M}})))$ is effectively computable.

In other words, seeing that MSO logic is closed by negation, the following problem is effectively decidable:

Problem 1 (Bounded Tree-Width MSO-Compositionality). Given an integer k, a distinguished symbol m_0, a vector of MSO-formulae $(\varphi_m)_{m\in\mathcal{M}}$ and a MSO-formula Ψ, does there exist $G \in \mathcal{E}xp(m_0, (\mathcal{M}_{\text{twd}<k}(\varphi_m))_{m\in\mathcal{M}})$ such that $G \models \Psi$?

[2] If all the $\mathfrak{L}_m$'s are finite, then the problem relies to the MSO-satisfiability of HR-equational hypergraphs, which is already known to be decidable (see [18]).

2 Motivation: Program Models

Here we define program models. They generalise[3] the concepts introduced in [5] which was inspired by [4]. In particular, this model allows us to catch security properties considered in [5], see also [4,6,7].

Let $\mathcal{M}$ be a set of symbols whose elements are the *names* of functions. A *function* is a pair (m, G_m) where $m \in \mathcal{M}$ is the name of the function and G_m is its *control-flow graph,* i.e., a tuple $(V_m, \rightarrow_m, \lambda_m)$ such that V_m is the finite set of *program points* of m. The relation $\rightarrow_m \subset V_m \times V_m$ encode the *transfer edges* of m. $\lambda_m : V_m \rightarrow \{\mathsf{entry}, \mathsf{seq}, \mathsf{ret}\} \cup \{\mathsf{call}\, m\}_{m \in \mathcal{M}}$ associates to each program point of m a program point type. The entry points are the entry points of the function. The call points are the points where the function calls an other function. The ret points are the returning points of the function. The seq points are all the other ones, which are not distinguished. A *program* $\mathcal{P}$ consists of a set of functions $\{(m, G_m)\}_{m \in \mathcal{M}}$. Let $V_\mathcal{P}$ denote the set of all the program points of all the functions of $\mathcal{P}$. A *state* of $\mathcal{P}$ is a pair (c, σ) consisting of a program point, i.e., an element of $V_\mathcal{P}$, and a call stack $\sigma \in V_\mathcal{P}^*$. A program $\mathcal{P}$ induces a *labelled transition system,* i.e., a graph $\mathcal{T}_\mathcal{P} = (\mathcal{S}_\mathcal{P}, \mathcal{L}_\mathcal{P}, \rightarrow_\mathcal{P})$ defined as follows: The set of *states* $\mathcal{S}_\mathcal{P}$ is the set of the states of $\mathcal{P}$ as defined above. The set of *edge labels* $\mathcal{L}_\mathcal{P}$ is defined to be $\{\tau, \mathsf{call}, \mathsf{ret}\}$. The call-labelled edges encode function calls. The ret-labelled edges encode function returns. Any other transition of the system is labelled by τ which is called the silent action (see [5]).

For each function name $m \in \mathcal{M}$, we consider the labelled transition system $\mathcal{T}_{\mathcal{P},m}$ defined to be the subgraph of all the vertices of $\mathcal{T}_\mathcal{P}$ which are accessible from a state of the form (c_i, ε) where c_i is an entry point of m.

Lemma 1. *$\mathcal{T}_{\mathcal{P},m}$ is a HR-equational graph.*

Remark 1. This result could actually be refined seeing that $\mathcal{T}_{\mathcal{P},m}$ is of finite degree: this implies that it is a context-free graph. Therefore, it can be constructed as the configuration graph of a pushdown automaton (see [17]), recovering the result of [5].

Any $\mathcal{M}$-vector $(\varphi_m)_{m \in \mathcal{M}}$ of MSO-formulæ specifies a set of programs
$\mathfrak{P}((\varphi_m)_{m \in \mathcal{M}}) = \{(m, G_m)_{m \in \mathcal{M}} \mid \forall\, m \in \mathcal{M} : G_m \models \varphi_m\}$.

Problem 2 (Program Models Compositionality). Given a set of formulæ $(\varphi_m)_{m \in \mathcal{M}}$, a method name m_0, and a formula Ψ, does there exist[4] a program
$\mathcal{P} \in \mathfrak{P}((\varphi_m)_{m \in \mathcal{M}})$ such that $\mathcal{T}_{\mathcal{P},m_0} \models \Psi$?

[3] We kept the formalism used in [5] to encode control-flow graphs and transition systems. However these concepts can be easily defined as relational structures.

[4] We state the Compositionality problem in an existential form for convenience. In our framework, this is equivalent to the classic statement seeing that the monadic second-order logic is closed by negation of formulæ.

This problem is Turing reductible to $\mathcal{T}h_{\text{MSO}}(\mathcal{E}xp(m_0, (\varphi_m)_{m\in\mathcal{M}}))$; and actually, it is also undecidable. However, if we restrict the study to control flow graph of bounded tree-width, we get a problem which is Turing reductible to Problem 1, i.e. to the theory of a bounded-tree-width MSO uniform expansion, and therefore decidable.

3 Decidability of Compositionality Problem

Here we introduce some tools for dealing with decidability issues concerning monadic second-order logic: HR-syntactic expressions and tree automata.

HR-syntactic expressions are the expressions based on the following operations on hypergraphs (see [18] for complete definitions): the *redefinition of the sources* of a hypergraph, and the *fusion of the sources* of a hypergraph. These operators are typed by the types of hypergraphs considered (see [18] for details). We consider the set $\mathfrak{T}_{L,k}$ (respectively $\mathfrak{T}^{\infty}_{L,k}$) made of well-formed *finite terms* (respectively *infinite*) on the set of operators dealing with hypergraphs of type bounded by k, and L-labelled hyperedges. The finite hypergraph denoted by a finite term of $\mathfrak{T}_{L,k}$, i.e. the *value* of this term, is defined inductively according to the definitions of the semantics of each operator which has been given above (see [18]). The value of an infinite term $t \in \mathfrak{T}^{\infty}_{L,k}$, which is an infinite hypergraph, can be defined in two ways: as the inductive limit of a sequence of finite hypergraphs defined from an increasing sequence of sub-terms of t, or as the quotient of a graph constructed from the set of leaves of t (see [18]). The value of a term is denoted by $\overline{t}$. *Any* finite hypergraph is the value of a term of some $\mathfrak{T}_{L,k}$, and *any* HR-equational hypergraph[5] is the value of a (regular) term of some $\mathfrak{T}^{\infty}_{L,k}$ (cf. [18]). Besides, the concept of syntactic expressions allows to give an alternative definition of the *tree-width*: the tree-width of a hypergraph is the smallest k such that it is the value of a term of $\mathfrak{T}_{L,k}$ (see [18]). The syntactic expression operators are of arity 0, 1 or 2. Therefore, the terms on these operators can be encoded by possibly infinite ordered binary trees labelled by operators. Here we use automata on such trees: *top-down tree automata* (see [20]) and *Rabin automata* (see [21]). The classical link between automata and monadic second-order logic applies in this framework according to the following result (see [18]): For any formula φ and any integer k, there exists a top-down automaton (respectively a Rabin automaton) $\mathcal{A}$ such that $\mathcal{M}_{\text{twd}<k}(\varphi) = \{\overline{t} \mid t \in \mathfrak{T}_{L,k} \text{ and } t \in \mathcal{L}(\mathcal{A})\}$ (respectively $\mathcal{M}^{\infty}_{\text{twd}<k}(\varphi) = \{\overline{t} \mid t \in \mathfrak{T}^{\infty}_{L,k} \text{ and } t \in \mathcal{L}(\mathcal{A})\}$). One can compute $\mathcal{A}$ from φ and conversely. Therefore Problem 1 can be reformulated in terms of automata as follows:

Problem 3 (Automata Compositionality). Given a distinguished symbol m_0, a $\mathcal{M}$-vector of top-down automata $(\mathcal{A}_m)_{m\in\mathcal{M}}$ and a Rabin automaton $\mathcal{A}'$, does there exist $G \in \mathcal{E}xp(m_0, (\mathcal{L}(\mathcal{A}_m))_{m\in\mathcal{M}})$ such that $G = \overline{t}$ for some $t \in \mathcal{L}(\mathcal{A}')$?

[5] However, some infinite hypergraphs such as the infinite grid have no syntactic expression in any $\mathfrak{T}^{\infty}_{L,k}$ (see [18].)

3.1 Composite Automata

Here we introduce the concept of *composite automata.* It is intended to catch sets of form $\mathcal{E}xp(m_0, (\mathcal{L}(\mathcal{A}_m))_{m\in\mathcal{M}})$ used in Problem 3 (see Lemma 2 below).

As in Problem 3, let us consider a pair $\mathfrak{S} = (m_0, (\mathcal{A}_m)_{m\in\mathcal{M}})$. Without loss of generality, one can assume that each $\mathcal{A}_m$ is such that no transition of it has an initial state in its target vector. Let us consider the automaton $\mathcal{A}(\mathfrak{S})$ constructed from the $\mathcal{A}_m$'s as follows:
1) First, one takes the disjoint union of all the $\mathcal{A}_m$'s. **2**) The terms recognized by $\mathcal{A}_m$ represents elements of $\mathcal{M}(\varphi_m)$. In such terms, the hyperedges encoding gluings regarding the grammar are encoded by sub-terms of form $E_{m'}$ for some m'. Since these last ones are of arity 0, these sub-terms actually are leaves of the complete term, when this last one is seen as a tree. Therefore, if we consider a run ρ of $\mathcal{A}_m$ on such a term, then ρ associates a terminal state to any sub-term of form $E_{m'}$. Let us consider such a terminal state q_f. Without loss of generality, one can assume that $\mathcal{A}_m$ is such that in any of its run, all the sub-terms to which q_f is associated are of the form $E_{m'}$ for a particular fixed m' depending only of q_f. One deletes q_f and replaces its occurrences in the target vector of any transition by an initial state of $\mathcal{A}_{m'}$, duplicating the transition in question for each initial state. **3**) The initial states are those of $\mathcal{A}_{m_0}$. **4**) The accepting condition for $\mathcal{A}(\mathfrak{S})$ shall be defined in Definition 1 below.

We distinguish in $\mathcal{A}(\mathfrak{S})$ all the states which are initial states of some $\mathcal{A}_m$; such a state is called a *boundary state.* $\mathcal{A}(\mathfrak{S})$ is called a *composite automaton.*

Induced Partitions. Roughly speaking, a run of a composite automaton on a given infinite tree induces a partition of the tree into finite subtrees, each of them being recognized by a composant of the automaton, i.e., the states coming from a particular $\mathcal{A}_m$. Such a run is accepting if, during the whole run, each composant of the automaton recognizes always the same finite tree (see Definition 1 below). Formally, let T be an infinite tree. A *partition* of T is a set $\{t_i\}_{i\in I}$ of finite subtrees of T such that the t_i's are pairwise disjoint and $T = \bigcup_i t_i$. Let $\{t_i\}_{i\in I}$ be such a partition. For any $i \in I$, let r_i denote the root of t_i. Let us note that I has a canonical non-ordered tree structure in which, for any pair $i, i' \in I$, i is the father of i' if and only if $r_{i'}$ is the son of some node of t_i. The set $\{r_i\}_{i\in I}$ is called the *internal boundary* of $\{t_i\}_{i\in I}$, it is denoted by $\mathcal{F}(\{t_i\}_{i\in I})$. We say that $\{t_i\}_{i\in I}$ is a *tiling* if there exists a finite subset $K \subset I$, assumed to be minimal, such that there exists a mapping $\lambda : I \to K$ such that for any $i \in I$, there exists an isomorphism $\sigma_i : t_i \to t_{\lambda(i)}$. The t_k's are called the *tiles.* $|K|$ is called the *index* of the tiling, it is denoted by *idx*(*K*). Such a tiling is said to be *deterministic*[6] if for any $k \in K$ there exists a mapping $\delta_k : t_k \to (K \cup \perp)^*$ where $\perp$ is a new symbol not belonging to K defined as follows: for any $\bar{x} \in t_k$, the n-th element of $\delta_k(\bar{x})$, denoted by $\delta_k(\bar{x})_n$, is equal to $\perp$ if and only if for any $i \in \lambda^{-1}(k)$ and

[6] A tree provided with a deterministic tiling is not necessarily deterministic it-self. Indeed, several patterns could occur inside a particular tile.

any $x \in \sigma_i^{-1}(\bar{x})$: the n-th son of x still is in t_i. And $\delta_k(\bar{x})_n = k'$ if and only if for any $i \in \lambda^{-1}(k)$ and any $x \in \sigma_i^{-1}(\bar{x})$, the n-th son of x does not belong to t_i and is equal to some $r_{i'}$ with $\lambda(i') = k'$. Let $d_{\max} = \max_{k,x}\{|\delta_k(x)|\}$. Let us assume that there exists an run $\rho : T \to Q_{\mathcal{A}(\mathfrak{S})}$ of $\mathcal{A}(\mathfrak{S})$ on T such that there exists a partition made of finite trees defined by the property that $\mathcal{F}(\mathfrak{P}(\rho))$ is the inverse image by ρ of the set of boundary states of $\mathcal{A}(\mathfrak{S})$ This partition is called the *induced partition* of T regarding ρ, it is denoted by $\mathfrak{P}(\rho)$. Let us consider the mapping $\tau : Q_{\mathcal{A}(\mathfrak{S})} \to \mathcal{M}$ defined by $\tau(q) = m$ if and only if q belongs to $\mathcal{A}_m$.

Definition 1 (Rigid Accepting Condition). *A run $\rho : T \to Q_{\mathcal{A}(\mathfrak{S})}$ is said to be* accepting *if $\mathfrak{P}(\rho)$ is a tiling $(K, \lambda, \{\sigma_i\}_{i\in I})$, and for any pair $i_1, i_2 \in I$, $\tau(\rho(r_{i_1})) = \tau(\rho(r_{i_2}))$ implies that $\lambda(i_1) = \lambda(i_2)$. The language of $\mathcal{A}(\mathfrak{S})$, denoted by $\mathcal{L}^\infty(\mathcal{A}(\mathfrak{S}))$, is defined to be the set of trees for which such a run exists.*

Lemma 2. *A graph G belongs to $\mathcal{E}xp(m_0, (\mathcal{L}(\mathcal{A}_m))_{m\in\mathcal{M}})$ if and only if it has a syntactic expression tree $T \in \mathcal{L}^\infty(\mathcal{A}(\mathfrak{S}))$.*

Simplification of Accepting Runs. Let $\rho : T \to Q_{\mathcal{A}(\mathfrak{S})}$ be an accepting run. Here we keep the notations of Definition 1. For any $m \in \mathcal{M}$, let $k_m \in K$ be such that $\tau(\rho(r_{k_m})) = m$. Let us note that k_m is unique because K is assumed to be minimal. In particular $idx(\mathfrak{P}(\rho)) \leq |\mathcal{M}|$. In order to simplify notations, let t_m (respectively r_m) denote t_{k_m} (respectively r_{k_m}). Let us define $\sigma : T \to \bigcup_m t_m$ the canonical extension of all the σ_i defined for any $n \in T$ by $\sigma(n) = \sigma_i(n)$ where t_i is the subtree of $\mathfrak{P}(\rho)$ which contains n. Let us now consider for any $m \in \mathcal{M}$ the mapping $\rho_m = \rho|_{t_m} : t_m \to Q_{\mathcal{A}_m} \subset Q_{\mathcal{A}(\mathfrak{S})}$. The family $\{\rho_m\}_{m\in\mathcal{M}}$ is denoted by Δ_ρ. Let $\lambda_\rho : I \to \mathcal{M}$ be defined by $\lambda_\rho(i) = \tau(\rho(r_i))$ for any $i \in I$. Finally, let us consider the mapping $\bar{\rho} : T \to Q_{\mathcal{A}(\mathfrak{S})}$ defined by $\forall n \in T : \bar{\rho}(n) = \rho_{\tau(\rho(n))}(\sigma(n))$.

Lemma 3. *$\bar{\rho}$ is an accepting run of $\mathcal{A}(\mathfrak{S})$, called a* simplified *accepting run.*

3.2 Tree Tilings and Rabin Automata

By Lemma 2, Problem 1 is equivalent to decide whether the intersection of the languages of a given composite automaton and a given Rabin automaton is empty or not. Here we study the runs of a Rabin automaton on trees which are recognized by some composite automaton. We focus on the proof of Lemma 4 below which aims at simplifying such runs. This is a corner-stone of the proof of the Theorem 1. Roughly speaking, it says that if a tree which has a deterministic tiling is recognized by a Rabin automaton, then there is a run which uses a bounded number of partial runs on each tile; the bound only depending on the number of tiles (and not on their sizes) and on the number of states of the automaton. This applies to trees which are simultaneously recognized by a composite automaton and a Rabin automaton.

Formally, let us consider a Rabin automaton $\mathcal{A}'$. Let $T \in \mathcal{L}(\mathcal{A}')$, let us assume that there exists a tiling $\mathcal{T} = \{t_i\}_{i\in I}$ of T defined by a tuple $(K, \lambda, (\sigma_i)_{i\in I})$. Let

$\rho' : T \to Q_{\mathcal{A}'}$ be an accepting run of $\mathcal{A}'$ on T. Let us consider the collection of mappings of the form $\rho' \circ \sigma_i^{-1} : t_{\lambda(i)} \to Q_{\mathcal{A}'}$ for $i \in I$. Since all the t_k's are finite and their number is also finite, this collection of mappings is actually finite. Let us index it by a finite set J and denote it by $\Delta'_{\mathcal{T},\rho'} = \{\rho'_j : t_{k_j} \to Q_{\mathcal{A}'}\}_{j \in J}$ where $k_j \in K$. According to this notation, let $\lambda'_{\mathcal{T},\rho'} : I \to J$ be the mapping such that for any $i \in I$: $\rho'|_{t_i} \equiv \rho'_{\lambda'_{\mathcal{T},\rho'}(i)} \circ \sigma_i$. $|\Delta'_{\mathcal{T},\rho'}|$ is called the *regularity index* of ρ' regarding $\mathcal{T}$, it is denoted by $reg(\mathcal{T}, \rho')$.

Lemma 4. *Let $\mathcal{A}'$ be a Rabin automaton and let $T \in \mathcal{L}(\mathcal{A}')$ having a deterministic tiling $\mathcal{T}$. Then there exists an accepting run $\rho' : T \to Q_{\mathcal{A}'}$ on T such that $reg(\mathcal{T}, \rho')$ is bounded by $2^{2 \times |Q_{\mathcal{A}'}| \times |\Sigma_{\mathcal{A}'}| \times \max\{idx(K), |Q_{\mathcal{A}'}|\}^{d_{\max}} \times 2^{|Q_{\mathcal{A}'}|}}$.*

The Game Point of View. Let us be given with $T \in \mathcal{L}(\mathcal{A}')$ and a deterministic tiling $\mathcal{T} = \{t_i\}_{i \in I}$ defined by a tuple $(K, \lambda, \{\sigma_i\}_{i \in I}, \{\delta_k\}_{k \in K})$. Let us consider a finite set of mappings $\Delta = \{\rho'_j : t_{k_j} \to Q_{\mathcal{A}'}\}_{j \in J}$ where $k_j \in K$ such that for any $j \in J$, ρ'_j is a partial run of $\mathcal{A}'$ on t_{k_j}. For each $j \in J$ and for each node x of t_{k_j}, let us consider the tuple $\xi_{j,x} = (q, \eta, w, p) \in Q_{\mathcal{A}'} \times \Sigma_{\mathcal{A}'} \times (K \cup Q_{\mathcal{A}'})^{*<d_{\max}} \times \mathcal{P}(Q_{\mathcal{A}'})$ defined as follows: $q = \rho'_j(x)$ and η is the label of x. The word w is obtained as follows: by definition of $\delta_{k_j}(x)$, for any position n such that $\delta_{k_j}(x)_n = \perp$, the n-th son of x is defined in t_{k_j}, let us denote it by x'. Then one puts $\rho'_j(x')$ at the n-th position in $\delta_{k_j}(x)$ in the place of $\perp$. Finally $p = \{\rho'_j(y) \mid y \in \alpha\}$ where α denotes the path of t_{k_j} starting at the root and ending at x. Let ξ_j denote the set of all the $(q, \eta, w, p) = \xi_{j,x}$ such that $x \in t_{k_j}$ and w contains at least one element of K.

For any transition $\tau = (q, \eta, (q_1, \ldots, q_{n'}))$ of $\mathcal{A}'$ and any $w \in (K \cup Q_{\mathcal{A}'})^{*<d_{\max}}$, let $\chi_{\tau,w} \subset J^{*<d_{\max}}$ be the set of words of form $j_0 \ldots j_n$ where n is the number of positions where some element of K occurs in w, and for any $\ell \in [1...n]$, $j_\ell \in J$ is such that k_{j_ℓ} is equal to the ℓ-th element of K occurring in w and $(q_1, \ldots, q_{n'})$ is obtained from w by replacing the ℓ-th element of K occurring in w by $\rho'_{j_\ell}(r_{k_{j_\ell}})$.

Let $\mathcal{G}_\Delta$ be the finite two-players game defined by the finite game graph $\Gamma_\Delta = (V_0, V_1, E, c, C)$ and the winning ω-language W where $V_0 = Q_{\mathcal{A}'} \times \Sigma_{\mathcal{A}'} \times (K \cup Q_{\mathcal{A}'})^{*<d_{\max}} \times \mathcal{P}(Q_{\mathcal{A}'})$ is the set of positions for player_1, $V_1 = J^{*<d_{\max}}$ is the set of positions for player_0, the set of transitions E groups transitions of the form $(q, \eta, w, p) \to w'$ such that $w' \in \chi_{\tau,w}$ where τ is a transition of $\mathcal{A}'$ of origin (q, η), and transitions of the form $j_0 \ldots j_n \to (q, \eta, w, p)$ such that $\exists e \in [1...n]$ such that $(q, \eta, w, p) \in \xi_{j_e}$, the colour set is $C = Q_{\mathcal{A}'}{}^*$ and $c : V \to C$ is defined by $c(v) = \varepsilon$ for any $v \in V_1$, and $c(q, \eta, w, p) = \tilde{p}$, where $\tilde{p} \in Q_{\mathcal{A}'}{}^*$ is defined to be the ordered sequence of all the elements of p according to a linear ordering of $Q_{\mathcal{A}'}$ fixed in advance once for all, finally, W is defined to be the language of infinite words accepted according to the acceptance condition of $\mathcal{A}'$ and starting at some initial state of $\mathcal{A}'$. Following the standard schema, we identify runs and strategies: Let ρ' be a run of $\mathcal{A}'$ on T. Let $\Delta'_{\mathcal{T},\rho'}$ and $\lambda'_{\mathcal{T},\rho'}$ be as defined page 545. Then ρ' defines a strategy $\mathcal{S}_{\rho'}$ in the game $\mathcal{G}_{\Delta'_{\mathcal{T},\rho'}}$ associated to $\Delta'_{\mathcal{T},\rho'}$. Here we define $\mathcal{S}_{\rho'}$ by describing how it drives a play: **1)** The first move of player_0 is $j_{0,0} = \rho'(root(T))$. **2)** Next, the first move of player_1 is defined as follows: According to the definition of $\mathcal{G}_{\Delta'_{\mathcal{T},\rho'}}$,

player$_1$ picks a node $\bar{x}_1 \in t_{k_{j_{0,0}}}$ such that $\xi_{j_{0,0},\bar{x}_1} \in \xi_{j_{0,0}}$; and he plays the tuple $(q_1, \eta_1, w_1, p_1) = \xi_{j_{0,0},\bar{x}_1}$. **3)** Let us consider the play at the turn number $\ell \geq 1$: let $\gamma_\ell = j_{0,0}|\ (q_1, \eta_1, w_1, p_1)|\ \cdots\ j_{\ell-1,0}j_{\ell-1,1}\cdots j_{\ell-1,n_{\ell-1}}|\ (q_\ell, \eta_\ell, w_\ell, p_\ell)$ be the successive positions which have appeared before. By induction on $\ell \geq 0$, let us assume that **a)** there exists a sequence $t_{i_0}, \ldots, t_{i_{\ell-1}}$ of ℓ elements of $\mathfrak{P}(\rho)$ such that i_0 is the root of I according to the natural tree structure of I, i.e., r_{i_0} is the root of T (cf. page 543), and for any $h \in [1...\ell-2]$: i_h is the father of i_{h+1}; in particular, the concatenation of all the α_h's is also a path of T; **b)** there exists a sequence of paths $\alpha_1 \subset t_{i_0}, \ldots, \alpha_\ell \subset t_{i_{\ell-1}}$ such that for any $h \in [1...\ell]$, α_h starts at the root $r_{i_{h-1}}$ of $t_{i_{h-1}}$ and ends at a node x_h which is the father of r_{i_h} if $h \leq \ell - 1$; **c)** for any $h \in [0...\ell-1]$: there exists $e_h \in [1...n_h]$ such that $(q_{h+1}, \eta_{h+1}, w_{h+1}, p_{h+1}) = \xi_{j_{h,e_h},\sigma_{i_h}(x_{h+1})}$.

The move of player$_0$. The strategy $\mathcal{S}_{\rho'}$ defined by ρ' is defined here: player$_0$ plays a word $j_{\ell,0}j_{\ell,1}\cdots j_{\ell,n_\ell}$ defined as follows: n_ℓ is the number of elements of K occurring in w_ℓ. For any $h \in [1...n_\ell]$, let l_h be the position of the h-th element of K occurring in w_ℓ, and let $s_h \in I$ be such that r_{s_h} is h-th son of x_ℓ. Then for any $h \in [1...n_\ell]$, we define $j_h = \lambda'_{\mathcal{T},\rho'}(s_h)$. This move is correct: Let τ be the transition of $\mathcal{A}'$ used by ρ' at x_ℓ. Then by construction we have that η_ℓ is the label of x_ℓ and $q_\ell = \rho'(x_\ell)$. Therefore (q_ℓ, η_ℓ) is indeed the origin of τ. By a detailed checking of the definition of χ_{τ,w_ℓ}, one also can verify that $j_{\ell,0}j_{\ell,1}\cdots j_{\ell,n_\ell} \in \chi_{\tau,w_\ell}$. The move is indeed correct.

The move of player$_1$. According to the definition of $\mathcal{G}_{\Delta'_{\mathcal{T},\rho'}}$, player$_1$ picks an integer $e_\ell \in [1...n_\ell]$ and a node $\bar{x}_{\ell+1} \in t_{k_{j_{\ell,e_\ell}}}$ such that $\xi_{j_{\ell,e_\ell},\bar{x}_{\ell+1}} \in \xi_{j_{\ell,e_\ell}}$; and he plays the tuple $(q_{\ell+1}, \eta_{\ell+1}, w_{\ell+1}, p_{\ell+1}) = \xi_{j_{\ell,e_\ell},\bar{x}_{\ell+1}}$.

Construction of $\alpha_{\ell+1}$ *and* t_{i_ℓ}. Let $\bar{\alpha}_{\ell+1}$ be the path going from the root of $t_{k_{j_{\ell,e_\ell}}}$ to $\bar{x}_{\ell+1}$. Let e'_ℓ be position in w_ℓ of the e_ℓ-th element of K occurring in it. Then i_ℓ is defined by the condition that r_{i_ℓ} is the e'_ℓ-th son of x_ℓ. Let us note that we have $\lambda(i_\ell) = (w_\ell)_{e'_\ell}$. Finally $\alpha_{\ell+1}$ is defined to be $\sigma_{i_\ell}^{-1}(\bar{\alpha})$. And $x_{\ell+1} = \sigma_{i_\ell}^{-1}(\bar{x}_{\ell+1})$. The induction hypothesis are obviously verified by construction.

Lemma 5. For *any run* ρ' *of* $\mathcal{A}'$, *if* ρ' *is an accepting run then* $\mathcal{S}_{\rho'}$ *is a winning strategy.*

Let us turn to the converse. Let $\mathcal{S}$ be a strategy in $\mathcal{G}_\Delta$. Let $j_0 = \mathcal{S}(\varepsilon)$. Let us suppose that $\rho'_{j_0}(r_{k_{j_0}})$ is an initial state of $\mathcal{A}'$. Such a strategy is said to *have the property* $\mathcal{P}_{init}$. Assuming that, $\mathcal{S}$ defines a run $\rho'_\mathcal{S}$ of $\mathcal{A}'$ such that $\Delta'_{\mathcal{T},\rho'_\mathcal{S}} \subset \{\rho'_j \in \Delta \mid j \in \mathcal{D}om(\mathcal{S})\}$. To define $\rho'_\mathcal{S}$, we shall construct the mapping $\lambda'_{\mathcal{T},\rho'_\mathcal{S}} : I \to J$ associated to $\rho'_\mathcal{S}$.

Let $i \in I$. Let β be the path starting at the root of T and ending at r_i, the root of t_i. Let $t_{i_0}, \ldots, t_{i_\theta}$ be the sequence of elements of $\mathfrak{P}(\rho)$ across which β goes; β ends at r_{i_θ}. For each $h \in [1...\theta]$, let $x_h \in t_{i_{h-1}}$ be the father of r_{i_h}. Then $x_h \in \beta$. Let e_h be such that r_{i_h} is the e_h-th son of x_h which does not belong to $t_{i_{h-1}}$; and let $e'_h \in [1...n_h]$ be such that r_{i_h} is the e'_h-th son of x_h. Let us consider the partial play defined as follows: According

to $\mathcal{S}$, the first move of player_0 is $j_{0,0} = \mathcal{S}(\varepsilon)$. Then, inductively, let us consider the play at the turn number $\ell \leq \theta$: let $\gamma_\ell = j_{0,0}| \ (q_1, \eta_1, w_1, p_1)| \ \ldots \ |(q_\ell, \eta_\ell, w_\ell, p_\ell)| \ j_{\ell,0} j_{\ell,1} \ldots j_{\ell,n_\ell}$ be the successive positions which have appeared before. We assume that the successive game positions $j_{0,0}| \ j_{1,0} j_{1,1} \ldots j_{1,n_1}| \ \ldots \ |j_{\ell,0} j_{\ell,1} \ldots j_{\ell,n_\ell}$ have been played by player_0 according to $\mathcal{S}$. If $\ell = \theta$, then the construction is finite. Otherwise, we define the next moves of player_0 and player_1: First, let us deal with the next move of player_1. Let us note that $\lambda(i_\ell) = k_{j_{\ell,e_\ell}}$. Therefore $\bar{x}_{\ell+1} = \sigma_{i_\ell}(x_{\ell+1})$ belongs to $t_{k_{j_{\ell,e_\ell}}}$. Then we define the next move of player_1 to be $(q_{\ell+1}, \eta_{\ell+1}, w_{\ell+1}, p_{\ell+1}) = \xi_{j_{\ell,e_\ell}, \bar{x}_{\ell+1}}$. Second, player_0 plays according $\mathcal{S}$: $j_{\ell+1,0} j_{\ell+1,1} \ldots j_{\ell+1,n_{\ell+1}} = \mathcal{S}((q_1, \eta_1, w_1, p_1) \ldots (q_{\ell+1}, \eta_{\ell+1}, w_{\ell+1}, p_{\ell+1}))$ By induction, we construct a play γ_θ. And finally, we set: $\lambda'_{\mathcal{T},\rho'}(i) = j_{\theta,e_\theta}$.

Lemma 6. *For any strategy $\mathcal{S}$ of player_0 in $\mathcal{G}_\Delta$, if $\mathcal{S}$ is a winning strategy and has the property $\mathcal{P}_{init}$, then $\rho'_{\mathcal{S}}$ is an accepting run.*

Lemma 7 (Strategy Reduction). *Let $\mathcal{G}$ be a game defined by a game graph $\Gamma = (V_0, V_1, E, c, C)$ with $|V_1|$ and $|C|$ finite. Let us suppose that there exists a winning strategy $\mathcal{S}$ for player_0 in $\mathcal{G}$. Then there exists an other strategy $\mathcal{S}'$ for player_0 in $\mathcal{G}$ such that $|\mathcal{S}'(\mathcal{D}om(\mathcal{S}'))| \leq |c(V_1)| \times 2^{2\times|V_0|}$. Moreover, $\mathcal{S}'$ can be computed in an effective way from $\mathcal{S}$.*

Proof of Lemma 4. Let us be given with an accepting run $\tilde{\rho}'$ on $T \in \mathcal{L}(\mathcal{A}')$. By Lemma 5, $\tilde{\rho}'$ defines a winning strategy $\mathcal{S}_{\tilde{\rho}'}$ for player_0 in $\mathcal{G}_{\Delta'_{\mathcal{T},\tilde{\rho}'}}$. Moreover, $\mathcal{S}_{\tilde{\rho}'}$ has the property $\mathcal{P}_{\text{init}}$. By Lemma 7, we can construct from $\mathcal{S}_{\tilde{\rho}'}$ a new winning strategy $\mathcal{S}'$ for player_0 such that $|\mathcal{S}'(\mathcal{D}om(\mathcal{S}'))|$ is bounded by $2^{2\times|Q_{\mathcal{A}'}|\times|\Sigma_{\mathcal{A}'}|\times\max\{idx(K),|Q_{\mathcal{A}'}|\}^{d_{\max}}\times 2^{|Q_{\mathcal{A}'}|}}$. Taking into account Remark 8, one can assume that $\mathcal{S}'$ has the property $\mathcal{P}_{\text{init}}$. By Lemma 6, $\mathcal{S}'$ defines an accepting run $\rho_{\mathcal{S}'}$ which satisfies the conditions of Lemma 4. □

3.3 Decidability

Decidability results from a pumping property obtained from Lemma 4: if the langages of a composite automaton and a Rabin automaton are not disjoint, then there is a tree in their intersection which has a deterministic tiling with *bounded* tiles (Lemma 8 below). Deciding if such a tree exists is therefore possible by enumeration.

Let $(\mathcal{A}_m)_{m\in\mathcal{M}}$ and $\mathcal{A}(\mathfrak{S})$ be such as in Section 3.1. Let $T \in \mathcal{L}^\infty(\mathcal{A}(\mathfrak{S}))$. Let $\rho : T \to Q_{\mathcal{A}(\mathfrak{S})}$ be an accepting run of $\mathcal{A}(\mathfrak{S})$ on T, and let $\mathfrak{P}(\rho) = \{t_i\}_{i\in I}$ be the induced partition of T associated to ρ. $\mathfrak{P}(\rho)$ is a deterministic tiling, let $(K, \lambda, \{\sigma_i\}_{i\in I}, \{\delta_k\}_{k\in K})$ be a tuple associated to it. We also consider the families $\{t_m\}_{m\in\mathcal{M}}$ and $\Delta_\rho = \{\rho_m : t_m \to Q_{\mathcal{A}_m}\}_{m\in\mathcal{M}}$, and the mapping $\lambda_\rho : I \to \mathcal{M}$ as defined at page 544. Let $\mathcal{A}'$ be a Rabin automaton, let us assume that $T \in \mathcal{L}(\mathcal{A}')$. Let $\rho' : T \to Q_{\mathcal{A}'}$ be an accepting run satisfying Lemma 4 regarding the deterministic tiling defined by $\mathfrak{P}(\rho)$. We keep the notations of Section 3.2.

Let us consider $\Delta'_{\mathfrak{P}(\rho),\rho'} = \{\rho'_j : t_{k_j} \to Q_{\mathcal{A}'}\}_{j\in J}$ defined in Section 3.2 and the associated mapping $\lambda'_{\mathfrak{P}(\rho),\rho'} : I \to J$.

Path Collapsing. Let $\alpha = x_0...x_n$ be a depth-increasing path in T; x_0 is the higher node and x_n the lower one. *Collapsing* α in T consists in deleting T_{x_0} and gluing T_{x_n} in place of it in T. The resulting tree is denoted by $\zeta_\alpha(T)$. Let $\pi_\alpha : T \to \zeta_\alpha(T)$ be the associated projection. Let $\rho : T \to Q'_{\mathcal{A}}$ be a run. The operation of collapsing α in T is said to be *compatible* with ρ if $\rho(x_0) = \rho(x_n)$. If this holds, then ρ induces a run on $\zeta_\alpha(T)$ which shall be denoted by $\zeta_\alpha(\rho)$. For any family $\mathcal{P}$ of pairwise disjoint paths of T, possibly infinite, the operation of collapsing all the paths of $\mathcal{P}$ simultaneously in T, denoted by $\zeta_\mathcal{P}$, is well defined by considering the equivalence relation made of the union of all the equivalence relations (seen as subsets of $T \times T$) associated to the paths of $\mathcal{P}$: $\mathcal{R}_\mathcal{P} = \bigcup_{\alpha\in\mathcal{P}} \mathcal{R}_\alpha$. Let $\pi_\mathcal{P} : T \to \zeta_\mathcal{P}(T)$ denote the associated projection. If ζ_α is compatible with ρ for any $\alpha \in \mathcal{P}$, then one consider the canonical run $\zeta_\mathcal{P}(\rho)$ of $\mathcal{A}'$ on $\zeta_\mathcal{P}(T)$. Let α be a path of t_m for some fixed m. Such a path defines a family of pairwise disjoint paths in T defined by $\sigma^{-1}(\alpha) = \{\sigma_i^{-1}(\alpha) \subset T \mid \lambda_\rho(i) = m\}$. Let us consider the simultaneous collapsing of all the paths of $\sigma^{-1}(\alpha)$ in T, i.e., $\zeta_{\sigma^{-1}(\alpha)}$; let it be denoted by ζ_α^∞. We also consider the associated projection $\pi_\alpha^\infty : T \to \zeta_{\sigma^{-1}(\alpha)}(T)$ and the associated run $\zeta_\alpha^\infty(\rho)$. Similarly, ζ_α^∞ is compatible with ρ' if and only if ζ_α is compatible with ρ'_j for any $j \in J$ such that $\lambda_\rho(k_j) = m$. In this case $\zeta_\alpha^\infty(\rho')$ still is a run of $\mathcal{A}'$ on $\zeta_\alpha^\infty(T)$, but non necessarily accepting.

Lemma 8. *Let* $\mathcal{B}(\mathcal{M},\mathcal{A}') = 2^{2\times|Q_{\mathcal{A}'}|\times|\Sigma_{\mathcal{A}'}|\times\max\{|\mathcal{M}|,|Q_{\mathcal{A}'}|\}^{d_{\max}}\times 2^{|Q_{\mathcal{A}'}|}}$. *If the height of* t_m *for some* $m \in \mathcal{M}$ *is greater than* $(\mathcal{B}(\mathcal{M},\mathcal{A}')|Q_{\mathcal{A}'}| + 2)|Q_{\mathcal{A}_m}||Q_{\mathcal{A}'}|^{\mathcal{B}(\mathcal{M},\mathcal{A}')}$ *then there exists a depth-increasing path* $\alpha \subset t_m$ *such that* ζ_α^∞ *is compatible with* ρ *and* ρ', *and such that* $\zeta_\alpha^\infty(\rho')$ *is a Rabin-accepting run of* $\mathcal{A}'$ *on* $\zeta_\alpha^\infty(T)$.

This lemma gives us a bound for enumeration and thus, allows us to decide Problem 3 in an effective way. This achieves the proof of Theorem 1.

Acknowledgements. The author thanks B. Courcelle and G. Sénizergues for many helpful comments and suggestions.

References

1. Grumberg, O., Long, D.: Model Checking and Modular Verification. ACM Trans. on Prog. Lang. & Syst. **16** (1994) 843–871
2. Abadi, M., Lamport, L.: Composing Specifications. ACM Transactions on Prog. Lang. and Systems (TOPLAS) **15** (1993) 73–132
3. M., C.E., Long, D.E., McMillan, K.L.: Compositional Model Checking. In: Proc. of the 4th Symp. on Logic in Comp. Sci. (LICS'89. (1989) 353–362
4. Jensen, T., Le Métayer, D., Thorn, T.: Verifying Security Properties of Control-Flow Graphs. In: Proc. of the 20th Symposium on Security and Privacy, Berkeley, IEEE Computer Society Press (1999) 89–103

5. Barthe, G., Gurov, D., Huisman, M.: Compositional Verification of Secure Applet Interactions. In: Fundamental Approaches to Soft. Eng. (FASE'02). Volume LNCS 2306. (2002) 15–32
6. Barthe, G., Courtieu, P., Dufay, G., Huisman, M., Mello de Sousa, S., Chugunov, G., Fredlund, L.A., Gurov, D.: Temporal Logic and Toolset for Applet Verification: Compositional Reasoning, Model Checking, Abstract Interpretation. Technical report, VERIFICARD Project, `http://www.verificard.org/` (Sept 2002) Deliverable 4.1.
7. Sprenger, C., Gurov, D., Huisman, M.: Simulation Logic, Applets and Compositional Verification. Technical Report 4890, INRIA (2003)
8. Dam, M., Gurov, D.: Compositional Verification of CCS processes. In: Proceedings of PSI'99. Volume LNCS 1755. (1999) 247–256
9. Stirling, C.: A complete compositional modal proof system for a subset of CCS. LNCS **194** (1985) 475–486
10. Andersen, H.R., Stirling, C., Winskel, G.: A compositional proof system for the modal mu-calculus. In: 9th Symp. on Logic in Comp. Sci. (LICS'94), IEEE Comp. Soc. Press (1994) 144–153
11. Namjoshi, K.S., Trefler, R.J.: On the completeness of compositional reasoning. In: Proc. of the 12th Int. Conference on Computer Aided Verification (CAV'00). Number 1855, Springer-Verlag (2000) 139–153
12. Gurevich, Y.: Monadic Second-Order Theories. In Barwise, J., Feferman, S., eds.: Model Theoretic Logic, Springer (1985) 479–506
13. Robertson, N., Seymour, P.: Some New Results on the Well-Quasi Ordering of Graphs. Annals of Discrete Math. **23** (1984) 343–354
14. Thorup, M.: All structured programs have small tree-width and good register allocation. Inf. and Comp. **142** (1998) 159–181
15. Gustedt, J., Maehle, O., Telle, J.: The treewidth of java programs. In: Proc. of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02), San Francisco. (2002) To appear in LNCS.
16. Thomas, W.: Languages, automata, and logic. In: Rozenberg and Salomaa ed., Handbook of Formal Languages. Springer Verlag. Volume 3. (1997) 389–455
17. Muller, D.E., Schupp, P.E.: The theory of ends, pushdown automata, and second-order logic. Theoretical Comp. Sci. **37** (1985) 51–75
18. Courcelle, B.: The Monadic Second-order Logic of Graphs II : Infinite Graphs of Bounded Width. Math. Syst. Theory **21** (1989) 187–221
19. Drewes, F., Kreowski, H.J., A., H.: Hyperedge Replacement Graph Grammars. In Rozenberg, G., ed.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1., World Scientific (1997) 95–162
20. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications. Technical report, LIFL – France (2003) `http://www.grappa.univ-lille3.fr/tata/`.
21. Rabin, M.O.: Decidability of Second-Order Theories and Automata on Infinite Trees. Trans. of the A.M.S. **141** (1969) 1–35

Event Structures for Resolvable Conflict*

Rob van Glabbeek and Gordon Plotkin

Laboratory for Foundations of Computer Science
School of Informatics, University of Edinburgh
Edinburgh EH9 3JZ, UK
rvg@cs.stanford.edu, gdp@inf.ed.ac.uk

Abstract. We propose a generalisation of Winskel's event structures, matching the expressive power of arbitrary Petri nets. In particular, our event structures capture resolvable conflict, besides disjunctive and conjunctive causality.

1 Introduction

Event structures were introduced in NIELSEN, PLOTKIN & WINSKEL [8] as abstract representations of the behaviour of safe Petri nets. They describe a concurrent system by means of a set of events, representing action occurrences, and for every two events d and e it is specified whether one of them is a prerequisite for the other, whether they exclude each other, or—the remaining case—whether they may happen concurrently. A formal definition can be found in Fig. 2. The behaviour of an event structure is formalised by associating to it a family of *configurations,* these being sets of events that occur during (partial) runs of the represented system. A configuration x can also be understood as a state of the represented system, namely the state reached after performing all events in x.

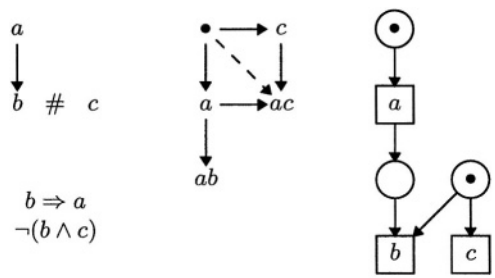


Fig. 1. An event structure as in [8] and its family of configurations, together with a transition relation between the configurations indicating how one can move from one state to another by concurrently performing some events. The same system is also represented by means of a propositional theory [2] and a Petri net.

WINSKEL later proposed a variety of other notions of event structure. In [9,10] he presented event structures where instead of indicating which events are prerequisites of other events, it is indicated which *sets* of events X are *possible* prerequisites of events e, written $X \vdash e$. This enables one to model *disjunctive causality* (cf. Fig. 3), the phenomenon that an event is causally dependent on a disjunction of other events occurring in the same system run. The event struc-

* This work was supported by EPSRC under grant number GR/S22097.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 550–561, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The event structures of NIELSEN, PLOTKIN & WINSKEL are triples $\mathrm{E} = \langle E, \leq, \# \rangle$ where
- E is a set of *events*,
- $\leq \subseteq E \times E$ is a partial order, the *causality relation*,
- $\# \subseteq E \times E$ is an irreflexive, symmetric relation, the *conflict relation*, satisfying the *principle of conflict heredity*: $\forall d, e, f \in E.\ d \leq e \wedge d \# f \Rightarrow e \# f$.

The set $L(\mathrm{E})$ of configurations of such a structure consists of those $X \subseteq E$ which are
- *conflict-free*: $\# \restriction (X \times X) = \emptyset$,
- and *left-closed*: $\forall d, e \in E.\ d \leq e \in X \Rightarrow d \in X$.

The prime event structures of WINSKEL [10] are defined likewise, but additionally requiring the *principle of finite causes*: $\{d \in E \mid d \leq e\}$ is finite for all $e \in E$.

The event structures of WINSKEL [9] are defined as triples $\mathrm{E} = \langle E, Con, \vdash \rangle$ where
- E is a set of *events*,
- $Con \subseteq \mathcal{P}_{fin}(E)$, satisfying $\emptyset \in Con$ and $Y \subseteq X \in Con \Rightarrow Y \in Con$, and
- $\vdash \subseteq Con \times E$ is the *enabling relation*, which satisfies $X \vdash e \wedge X \subseteq Y \in Con \Rightarrow Y \vdash e$.

Such a structure is *stable* if $X \vdash e \wedge Y \vdash e \wedge Con(X \cup Y \cup \{e\}) \Rightarrow X \cap Y \vdash e$.

The set $S(\mathrm{E})$ of configurations of such a structure consists of those $X \subseteq E$ which are
- *consistent*: every finite subset of X is in Con,
- and *secured*: $\forall e \in X.\ \exists e_0, \ldots, e_n \in X.\ e_n = e \wedge \forall i \leq n.\ \{e_0, \ldots, e_{i-1}\} \vdash e_i$.

The event structures of WINSKEL [10] are defined likewise, except that the consistency predicate Con is generated by a given symmetric, irreflexive *conflict relation* $\# \subseteq E \times E$, through $Con(X) \Leftrightarrow (X \text{ finite}) \wedge \forall d, e \in X.\ \neg(d \# e)$.

The prime event structures of WINSKEL [9] are defined as triples $\mathrm{E} = \langle E, Con, \leq \rangle$ combining the requirements for Con from [9] with those for $\leq$ from [10], and additionally satisfying $\{e\} \in Con$ for all $e \in E$ and $d \leq e \in X \in Con \Rightarrow X \cup \{d\} \in Con$.

Fig. 2. Formal definitions of 5 types of event structures

tures in [9] moreover allow one to express for any finite set of events whether it is *in conflict,* i.e. can not happen in full in the same run; in [8,10] this can only be specified for sets with two events.

However, not every Petri net can be faithfully represented as an event structure from [8,9,10], due to the phenomena of *resolvable conflict* illustrated in Fig. 4. In order to capture this type of behaviour, we simply extend the notion of event structure from [9,10] by allowing enablings of the form $X \vdash Y$, with X and Y sets of events. The enablings $X \vdash Y$ do not place any restrictions on the occurrence of individual events in Y, but say that for *all* events in Y to occur, for some set X with $X \vdash Y$ the events in X have to happen first.

Definition 1. An *event structure* is a pair $\mathrm{E} = \langle E, \vdash \rangle$ with
- E a set of *events,*
- $\vdash \subseteq \mathcal{P}(E) \times \mathcal{P}(E)$, the *enabling relation.*

With this type of event structure we do not need a separate conflict or consistency relation; that a set X of events is in irresolvable conflict can be expressed by not

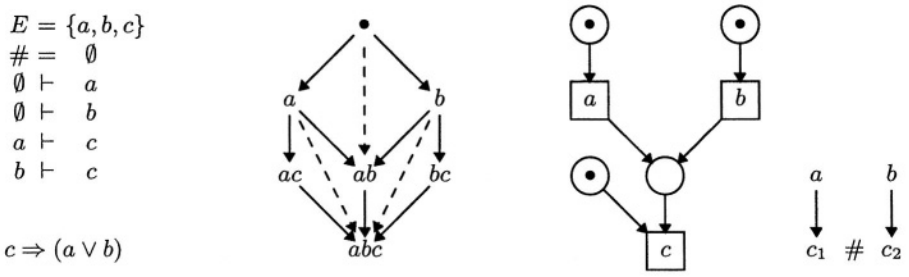


Fig. 3. A system with *disjunctive causality* represented as an event structure of [10], a family of configurations with transition relation, a propositional theory and a Petri net. The last picture is the best representation of the same system as an event structure from [8]. It requires the decomposition of the event c, which is causally dependent on the disjunction of a and b, into two events c_1 and c_2, only one of which may happen: c_1 being causally dependent only on a, and c_2 on b. Antoni Mazurkiewicz argued against the accuracy of this representation by letting a and b be £1 contributions of two school children to buy a present for their teacher. The act of buying the present, which only costs £1, is represented by c. Now the event structure from [8] has two maximal runs, representing that the present is bought from the contribution from either one child or the other. The event structure from [10] on the other hand has only one possible run, in which the buying of the present is caused by the disjunction of the two contributions. The latter would be a fairer description of the intended state of affairs.

having any enabling of the form $Y \vdash X$. When describing an event structure of [9,10] as one of ours, we have to omit enablings $X \vdash e$ with $\{e\} \notin Con$ and add enablings $\emptyset \vdash X$ for sets X with $|X| \neq 1$ and *Con(X),* and also for infinite X.

In Sect. 2 we discuss various forms of behavioural equivalence on these new event structures. In Sect. 3 we show how they include the classical event structures, thereby establishing their *generality.* In Sect. 4 we consider the relation

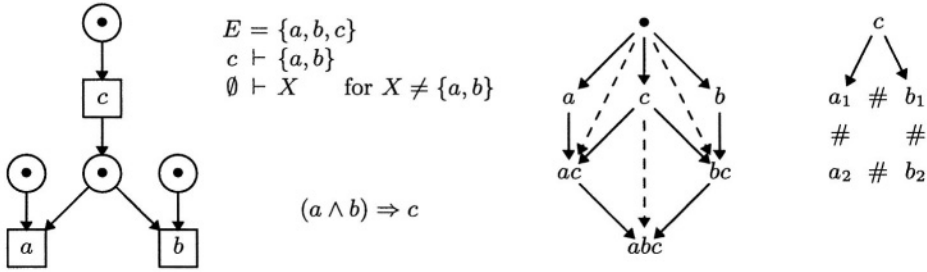


Fig. 4. A system with *resolvable conflict* represented as a Petri net, an event structure as introduced here, a family of configurations with transition relation, and a propositional theory. The events a and b are initially in conflict (only one of them may happen), but as soon as c occurs this conflict is resolved. The last picture is the best representation of the same system as an event structure from [8], again with arguable accuracy. It yields a system with two maximal runs, in one of which c causes just a, and in the other just b.

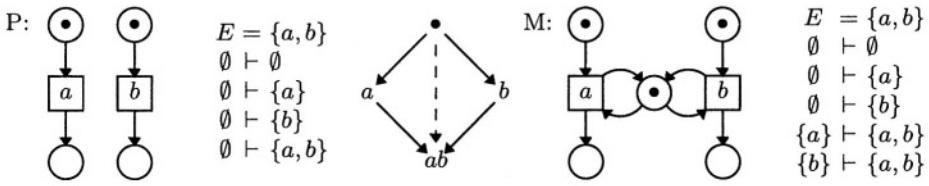


Fig. 5. Two systems P *(parallelism)* and M *(mutual exclusion)* represented as a Petri net and an event structure. The figure in the middle describes the configurations of P and M, as well as the transition relation of P. The dashed transition is lacking in Q. Even though P and M have the same configurations and single action transition relations, their behaviour is different, as witnessed by the transition $\emptyset \longrightarrow_{\mathrm{P}} \{a, b\}$.

between event structures and Petri nets, making use of infinitary prepositional theories to translate between them. We show that our new event structures enable us to represent any Petri net, thereby establishing their *universality.*

2 Configurations and Transitions

We formalise the dynamic behaviour of an event structure by defining a transition relation between sets of events. The idea here is that when X is the set of events that have happened so far, an additional set U of events can happen (concurrently) iff every subset of $X \cup U$ is enabled by a set of events that have happened before, i.e. a subset of X.

Definition 2. The *step transition relation* $\longrightarrow_{\mathrm{E}}$ between sets of events $X, Y \subseteq E$ of an event structure $\mathrm{E} = \langle E, \vdash \rangle$ is given by

$$X \longrightarrow_{\mathrm{E}} Y \Leftrightarrow (X \subseteq Y \wedge \forall Z \subseteq Y.\ \exists W \subseteq X.\ W \vdash Z).$$

For the *single action* transition relation we also require that $|Y - X| \leq 1$.
The set $L(\mathrm{E})$ of *(left-closed) configurations* of E is $L(\mathrm{E}) = \{X \subseteq E \mid X \longrightarrow_{\mathrm{E}} X\}$.
Two event structures E and F are *transition equivalent* if they have the same events and $\longrightarrow_{\mathrm{E}} = \longrightarrow_{\mathrm{F}}$.

Thus, $X \in L(\mathrm{E}) \Leftrightarrow \forall Y \subseteq X.\ \exists Z \subseteq X.\ Z \vdash Y$ and if $X \longrightarrow_{\mathrm{E}} Y$ then $X, Y \in L(\mathrm{E})$.
In Figs. 1, 3 and 4 we have indicated the single action transition relation with solid arrows, and the rest of the step transition relation with dashed ones. Figure 5 shows that the step transition relation can provide important information about an event structure which is included neither in its family of configurations nor in its single action transition relation.

2.1 Purity

We now introduce an important class of event structures whose step transition relation is completely determined by their family of configurations.

Definition 3. An event structure is *pure* if $X \vdash Y$ only if $X \cap Y = \emptyset$.

Proposition 1. *Let* $\mathrm{E} = \langle E, \vdash \rangle$ *be a pure event structure, and* $x, y \in L(\mathrm{E})$. *Then*

$$x \longrightarrow_{\mathrm{E}} y \quad \textit{iff} \quad x \subseteq y \wedge \forall Z (x \subseteq Z \subseteq y \Rightarrow Z \in L(\mathrm{E})).$$

Proof. "Only if" follows immediately from the definitions.

For "if" let $x \subseteq y$ and $\forall Z(x \subseteq Z \subseteq y \Rightarrow Z \in L(\mathrm{E}))$. Let $Z \subseteq y$. Then $x \subseteq x \cup Z \subseteq y$, so $x \cup Z \in L(\mathrm{E})$. Hence, by Definition 2, $\exists W \subseteq x \cup Z.\ W \vdash Z$. As E is pure, $W \cap Z = \emptyset$, hence $W \subseteq x$, which had to be proved. □

Corollary 1. *Two pure event structures* E *and* F *are transition equivalent iff they have the same events and* $L(\mathrm{E}) = L(\mathrm{F})$.

2.2 Reachability

It can be argued that only the reachable configurations and the reachable part of the step transition relation are semantically relevant.

Definition 4. A configuration x of an event structure E is *reachable* if there is a sequence $\emptyset = x_0 \longrightarrow_{\mathrm{E}} x_1 \longrightarrow_{\mathrm{E}} \ldots \longrightarrow_{\mathrm{E}} x_n = x$. Let $R(\mathrm{E})$ denote the set of reachable configurations of E. Two event structures E and F are *reachable transition equivalent* if they have the same events and $\longrightarrow_{\mathrm{E}} \restriction R(\mathrm{E}) = \longrightarrow_{\mathrm{F}} \restriction R(\mathrm{F})$.

Clearly, transition equivalence is finer than reachable transition equivalence. The following example shows that this is strictly so.

Example 1. Take as events of E the set $\mathbb{Q}$ of rational numbers and define $\vdash$ by $\emptyset \vdash X$ for any X with $|X| \neq 1$, and $X \vdash \{e\}$ iff $X = \{d \in \mathbb{Q} \mid d < e\}$. We have $R(\mathrm{E}) = \{\emptyset\}$, whereas $L(\mathrm{E})$ additionally contains representatives of all reals as well as extra copies of the rationals and $\mathbb{Q}$ itself (infinity). If F is $\langle \mathbb{Q}, \{\emptyset \vdash \emptyset\} \rangle$ then E and F are reachable transition equivalent, yet $L(\mathrm{E}) \neq L(\mathrm{F})$ (hence $\longrightarrow_{\mathrm{E}} \neq \longrightarrow_{\mathrm{F}}$).

The following shows that, unlike in the pure case, the reachable configurations of impure event structures, and thus also their step transition relations, are in general not determined by their left-closed configurations.

Example 2. Let $\mathrm{E} = \langle \{e\}, \{\emptyset \vdash \emptyset, \{e\} \vdash \{e\}\} \rangle$. Then $L(\mathrm{E}) = \{\emptyset, \{e\}\}$, whereas $R(\mathrm{E}) = \{\emptyset\}$. Let $\mathrm{F} = \langle \{e\}, \{\emptyset \vdash \emptyset, \emptyset \vdash \{e\}\} \rangle$. Then $L(\mathrm{E}) = L(\mathrm{F})$ but $R(\mathrm{E}) \neq R(\mathrm{F})$.

Using Prop. 1 we obtain a reachable analogue of Cor. 1. This result can be slightly strengthened as follows.

Definition 5. *Call an event structure* reachably pure *if* $X \vdash Y$ *only if either* $X \cap Y = \emptyset$ *or* $Y \subseteq X$.

The event structure E of Example 2 for instance is reachably pure, but not pure.

Proposition 2. *Two reachably pure event structures* E *and* F *are reachable transition equivalent iff they have the same events and* $R(\mathrm{E}) = R(\mathrm{F})$.

Proof. Enablings $X \vdash Y$ with $\emptyset \neq Y \subseteq X$ can be omitted while preserving the reachable configurations and the step transition relation between them.

3 Connecting with Classical Event Structures

In this section we define various properties of our event structures which, in suitable combinations, determine subclasses corresponding to the various event structures in [8,9,10]. We also show how our left-closed configurations generalise the configurations used for the event structures of [8] as well as for the prime event structures of [9,10], and we develop a notion of *secured configuration* that generalises the notion of configuration used for all event structures of [9,10].

Definition 6. Let $\mathrm{E} = \langle E, \vdash \rangle$ be an event structure. A set of events $X \subseteq E$ is *consistent,* written $Con(X)$, if $\forall Y \subseteq X.\ \exists Z \subseteq E.\ Z \vdash Y$. The *direct causality relation* $\prec \subseteq E \times E$ is given by $d \prec e \Leftrightarrow \forall X.\ (X \vdash \{e\} \Rightarrow d \in X)$. We take the *causality relation,* $\leq$, to be the reflexive and transitive closure of $\prec$. E is

- *rooted* if $\emptyset \vdash \emptyset$,
- *singular* if $X \vdash Y \Rightarrow X = \emptyset \vee |Y| = 1$,
- *conjunctive* if $X_i \vdash Y\ (i \in I \neq \emptyset) \Rightarrow \bigcap_{i \in I} X_i \vdash Y$,
- *locally conjunctive* if $X_i \vdash Y$ (for $i \in I \neq \emptyset$) $\wedge\, Con(\bigcup_{i \in I} X_i \cup Y) \Rightarrow \bigcap_{i \in I} X_i \vdash Y$,
- $\mathcal{L}$*-irredundant* if each event occurs in a configuration, i.e. $E = \bigcup L(\mathrm{E})$,
- $\mathcal{R}$*-irredundant* if $E = \bigcup R(\mathrm{E})$,
- and *cycle-free* if there is no chain $e_0 \prec e_1 \prec \cdots \prec e_n \prec e_0$

and has

- *finite causes* if $X \vdash Y \Rightarrow X$ finite,
- *finite conflict* if X infinite $\Rightarrow \emptyset \vdash X$
- and *binary conflict* if $|X| > 2 \Rightarrow \emptyset \vdash X$.

Clearly, conjunctivity implies local conjunctivity, $\mathcal{R}$-irredundancy implies $\mathcal{L}$-irredundancy and cycle-freeness, and binary conflict implies finite conflict.

3.1 Correspondence Through Left-Closed Configurations

For singular event structures, our notion of a left-closed configuration can be simplified as follows:

Observation 1. Let E be a singular event structure. Then

$$X \in L(\mathrm{E}) \Leftrightarrow Con(X) \wedge \forall e \in X.\ \exists Z \subseteq X.\ Z \vdash \{e\}.$$

When $d \leq e$, any configuration containing e also contains d. When $\mathrm{E} = \langle E, \vdash \rangle$ is conjunctive, for any consistent event $e \in E$ there is a smallest set $X \subseteq E$ with $X \vdash \{e\}$. Therefore, the part of the enabling relation consisting of enablings $X \vdash \{e\}$ is in essence completely determined by the causality relation $\leq$.

Observation 2. Let E be a singular, conjunctive event structure. Then

$$X \in L(\mathrm{E}) \Leftrightarrow Con(X) \wedge \forall d, e \in E.\ d \leq e \in X \Rightarrow d \in X.$$

An event structure as in [8], or a prime event structure as in [9,10], can be translated into our framework by defining enablings $\{d \mid d < e\} \vdash \{e\}$, as well as some enablings of the form $\emptyset \vdash X$. For prime event structures with a binary conflict relation # we take $\emptyset \vdash X$ whenever $|X| \neq 1, 2$, and when $X = \{d, e\}$ with $\neg(d \# e)$; for prime event structures as in [9] we take $\emptyset \vdash X$ when X is

infinite or $|X| \neq 1$ and $Con(X)$. Clearly, the resulting event structure is pure. Hence the dynamic behaviour of such an event structure, as given by its step transition relation, is fully determined by its left-closed configurations.

Using the translation given above, an event structure as in [8], or a prime event structure as in [9,10], maps to an event structure in our sense that is singular and conjunctive (as well as rooted and with finite conflict). By Obs. 2, it now follows that our notion of left-closed configuration generalises the notion of configuration employed in [8] and for prime event structures in [9,10].

3.2 Correspondence Through Secured Configurations

Definition 7. A set of events X is a *secured configuration* of an event structure E if there is an infinite sequence $\emptyset = x_0 \longrightarrow_{\mathrm{E}} x_1 \longrightarrow_{\mathrm{E}} \ldots$ with $X = \bigcup_{i=0}^{\infty} x_i$.

Computationally, a secured configuration can be understood by partitioning time in countably many successive stages S_n $(n \geq 1)$. The set $x_n - x_{n-1}$ contains the events that occur during stage S_n. These events must be enabled by events occurring in earlier stages. The set X contains all events that happen during such a run. The secured configurations include the reachable ones (just take $x_i = x_n$ for $i > n$).

Observation 3. Any event structure E with finite conflict satisfies $S(\mathrm{E}) \subseteq L(\mathrm{E})$.

For the event structures that result from mapping prime event structures as in [9,10] into our framework we find, using the principle of finite causes, that all left-closed configurations are secured. It follows that both the secured and the left-closed configurations can be understood as generalisations of the notion of configuration for prime event structures from [9,10].

Proposition 3. *Two singular event structures with finite conflict* E *and* F *are reachable transition equivalent iff they have the same events and* S(E) = S(F).

Proof. "only if" follows immediately from Definition 7.

"if": Singular event structures are always reachably pure. Using the proof of Prop. 2, we can restrict attention to the case that E and F are pure. For $\mathcal{F}$ one of L, R or S we define $\longrightarrow_{\mathcal{F}(\mathrm{E})}$ by

$$x \longrightarrow_{\mathcal{F}(\mathrm{E})} y \text{ iff } x \subseteq y \wedge \forall Z(x \subseteq Z \subseteq y \Rightarrow Z \in \mathcal{F}(\mathrm{E})).$$

By Definition 7 and Obs. 3 we have $R(\mathrm{E}) \subseteq S(\mathrm{E}) \subseteq L(\mathrm{E})$, hence

$$\longrightarrow_{\mathrm{E}} \restriction R(\mathrm{E}) \;=\; \longrightarrow_{R(\mathrm{E})} \;\subseteq\; \longrightarrow_{S(\mathrm{E})} \;\subseteq\; \longrightarrow_{L(\mathrm{E})} \;=\; \longrightarrow_{\mathrm{E}}.$$

As the reachable part of both $\longrightarrow_{R(\mathrm{E})}$ and $\longrightarrow_{L(\mathrm{E})}$ (defined the obvious way) is $\longrightarrow_{\mathrm{E}} \restriction R(\mathrm{E})$, the reachable part of $\longrightarrow_{S(\mathrm{E})}$ must also be $\longrightarrow_{\mathrm{E}} \restriction R(\mathrm{E})$, and so the latter is fully determined by S(E).

Using the translation given at the end of Sect. 1, an event structure as in [9,10] maps to an event structure in our sense that is singular and with finite causes and finite conflict. Hence the dynamic behaviour of such an event structure, as given by the reachable part of its step transition relation, is fully determined by its secured configurations.

Table 1. This table indicates in which way the event structures from [8,9,10] correspond with subclasses of our event structures. In all 7 cases we have that any event structure from [8,9,10] translates into one of ours with the listed properties, that has the same events and configurations; and vice versa, that for each of our event structures with the required properties an event structure from [8,9,10] can be found that has the same events and configurations. As indicated, we use the left-closed configurations for the event structures from [8], and the secured configurations for the ones from [9,10]. For the prime event structures from [9,10] the two notions of configuration coincide.

ev. str. [9]	rooted, singular, finite causes & finite conflict	S
stable [9]	same & locally conjunctive	S
prime [9]	same & conjunctive & $\mathcal{R}$-irredundant	S, L
ev. str. [10]	rooted, singular, finite causes & binary conflict	S
stable [10]	same & locally conjunctive	S
prime [10]	same & conjunctive & $\mathcal{R}$-irredundant	S, L
ev. str. [8]	rooted, singular, binary conflict, conjunctive, $\mathcal{L}$-irr. & cycle-free	L

The next proposition says that for such event structures the secured configurations in turn are completely determined by the finite secured ones. In addition, it provides a simplification of the notion of a secured configuration.

Proposition 4. *Let* E *be a singular event structure with finite conflict and finite causes. Then*

$$X \in S(\mathrm{E}) \Leftrightarrow \forall Y \subseteq_{fin} X.\ \exists Z \in S(\mathrm{E}).\ Z \text{ is finite} \wedge Y \subseteq Z \subseteq X,$$

i. e. the secured configurations are the directed unions *over the set of finite secured configurations. Moreover,*

$$X \in S(\mathrm{E}) \Leftrightarrow \begin{cases} Con(X) \wedge \\ \forall e \in X.\ \exists e_0, \ldots, e_n \in X.\ e = e_n \wedge \\ \forall k \leq n.\ \exists Y \subseteq \{e_0, ..., e_{k-1}\}.\ Y \vdash \{e_k\}. \end{cases}$$

The proof of this proposition will appear in [3]. It follows that our secured configurations generalise the configurations of [9,10]. Table 1 tells exactly how the various event structures of [8,9,10] can be regarded as subclasses of our event structures. Again, the proofs of the claims therein will be provided in [3],

4 Petri Nets and Propositional Theories

In this section we describe how any Petri net can be represented, in a behaviour preserving way, by a rooted event structure with finite conflict, and vice versa. We also show how to represent an event structure as a propositional theory.

4.1 From Nets to Event Structures

Definition 8. A *Petri net* is a tuple $\mathrm{N} = \langle S, T, F, I \rangle$ with
- S and T two disjoint sets of *places* (*Stellen* in German) and *transitions,*
- $F \subseteq S \times T \cup T \times S$, the *flow relation,*
- and $I : S \to \mathbb{N}$, the *initial marking.*

In [2] we described how any Petri net can be transformed, in a behaviour preserving way, into a *1-occurrence net,* this being a Petri net with the property that in any run each transition can fire at most once. The transformation replaces any transition by countably many copies, each of which is connected to the places of the net (through the flow relation) in the same way as the original transition. Each of the obtained transitions gets a private preplace, initially marked with 1 token. This ensures that whenever a transition could fire in the original net, one of its copies can fire in the transformed net—but each of the new transitions can fire only once. A formal account of the way in which this transformation is behaviour preserving would require the use of labelled Petri nets.

We now show how any 1-occurrence net can be represented as an event structure. Let $\mathrm{N} = \langle S, E, F, I\rangle$ be a 1-occurrence net. For any place $s \in S$ let $s^\bullet = \{t \in E \mid (s,t) \in F\}$ be its set of *posttransitions* and ${}^\bullet s = \{t \in E \mid (t,e) \in F\}$ its set of *pretransitions.* For any finite set $Y \subseteq s^\bullet$ of posttransitions of s, $|Y|$ is the number of tokens needed in place s for all transitions in Y to fire, so $|Y| \dot{-} I(s)$ is the number of tokens that have to arrive in place s before all transitions in Y can fire. Furthermore, let ${}^n s = \{X \subseteq {}^\bullet s \mid |X| = n\}$ be the collection of sets X of pretransitions of s, such that if all transitions in X fire, $n \in \mathbb{N}$ tokens will arrive in s. Write ${}^Y s$ for ${}^{|Y| \dot{-} I(s)} s$. One of the sets of transitions in ${}^Y s$ has to fire entirely before all transitions in Y can fire.

For any finite set of transitions $Y \subseteq E$, let S_Y be the set of places s with $Y \subseteq s^\bullet$ and $|Y| - I(s) > 0$. Now write $X \vdash_{\mathrm{N}} Y$ whenever $X = \bigcup_{s \in S_Y} X_s$ with $X_s \in {}^Y s$. We also write $\emptyset \vdash_{\mathrm{N}} Y$ whenever Y is infinite. The *event structure associated to* N is defined as $\mathcal{E}(\mathrm{N}) = \langle E, \vdash_{\mathrm{N}}\rangle$. Note that $\mathcal{E}(\mathrm{N})$ is rooted and with finite conflict. It can be shown that this event structure has the same *step transition relation* as N, at least when restricting to steps of finitely many events, although we didn't have space to formalise the latter notion for Petri nets here.

It is not hard to extend the above construction to nets with *arcweights* [3].

4.2 From Event Structures to Propositional Theories

With any event structure $\mathrm{E} = \langle E, \vdash\rangle$ we associate the (infinitary) propositional theory

$$T(\mathrm{E}) = \{\bigwedge X \Rightarrow \bigvee\{\bigwedge Y \mid Y \vdash X\} \mid X \subseteq E\}.$$

In this context, an event is regarded as the proposition that it has happened. The propositional formulae generated above give necessary and sufficient conditions for a set of events to be a left-closed configuration; it is not hard to see that $L(\mathrm{E})$ is exactly the set of models of $T(\mathrm{E})$ in the sense of propositional logic.

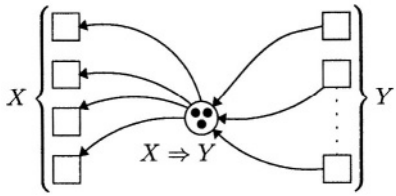
A propositional theory provides a pleasant alternative representation of an event structure; examples of this can be found in the figures of Sect. 1.

For any two subsets X,Y of E, let the *clause* $X \Rightarrow Y$ abbreviate the implication $\bigwedge X \Rightarrow \bigvee Y$. A theory consisting of a set of clauses is said to be in *conjunctive normal form.* Using the distributivity of $\bigvee$ over $\bigwedge$, and that $\varphi \Rightarrow \bigwedge_{i \in I} \psi_i$ is equivalent to $\bigwedge_{i \in I}(\varphi \Rightarrow \psi_i)$, the theory $T(\mathrm{E})$ can be turned into the conjunctive normal form $T_{\mathrm{CNF}}(\mathrm{E})$. We say that a propositional theory in conjunctive normal

form is *rooted* if it has no clauses of the form $\emptyset \Rightarrow X$, and that it has *finite conflict* if there are no clauses $X \Rightarrow Y$ with X infinite. Clearly, if E is a rooted and with finite conflict, then so is $T_{\mathrm{CNF}}(\mathrm{E})$.

4.3 From Propositional Theories to Petri Nets

Let $\mathrm{T} = \langle E, T \rangle$ be a propositional theory in conjunctive normal form that is rooted and with finite conflict. As in [2], we define the associated Petri net $\mathcal{N}(\mathrm{T})$ as follows. As transitions of the net we take the events from E. For every transition we add one place, containing one initial token, that has no incoming arcs, and with its only outgoing arc going to that transition. These *1-occurrence places* make sure that every transition fires at most once. For every clause $X \Rightarrow Y$ in T, we introduce a place in the net. This place has outgoing arcs to each of the transitions in X, and incoming arcs from each of the places in Y. Let n be the cardinality of X. As T is rooted and with finite conflict, $n \neq 0$ and n is finite. We finish the construction by putting $n - 1$ initial tokens in the created place:



The place belonging to the clause $X \Rightarrow Y$ does not place any restrictions on the firing of the first $n - 1$ transitions in X. However, the last one can only fire after an extra token arrives in the place. This can happen only if one of the transitions in Y fires first. The firing of more transitions in Y has no adverse effects, as each of the transitions in X can fire only once. Thus this place imposes the same restriction on the occurrence of events as the corresponding clause.

It should be intuitively clear that the dynamic behaviour of $\mathcal{N}(T_{\mathrm{CNF}}(\mathrm{E}))$ strongly resembles that of E, although it should be admitted that in the standard semantics of Petri nets only finitely many transitions may fire in one step, whereas Definition 2 allows infinite steps. Nevertheless, we have

Theorem 1. *Let* E *be a rooted event structure with finite conflict. Then* $\mathcal{E}(\mathcal{N}(T_{\mathrm{CNF}}(\mathrm{E})))$ *is transition equivalent to* E.

Proof sketch. It is straightforward to find mappings $\mathcal{T}$ from nets to propositional theories [3] and $\mathcal{E}'$ from propositional theories of the form $\mathcal{T}(\mathrm{N})$ with N a net to event structures, such that $\mathcal{E}(\mathrm{N}) = \mathcal{E}'(\mathcal{T}(\mathrm{N}))$ for all Petri nets, $\mathcal{T}(\mathcal{N}(\mathrm{T})) = \mathrm{T}$ for all propositional theories T in conjunctive normal form, and $\mathcal{E}'(T_{\mathrm{CNF}}(\mathrm{E})) \equiv \mathrm{E}$ for all event structures E, where $\equiv$ denotes transition equivalence.

In general this theorem depends intrinsically on the specific form of $T_{\mathrm{CNF}}(\mathrm{E})$; however, for pure event structures any conjunctive normal form of $T(\mathrm{E})$ (up to *logical equivalence,* i.e. having the same models) will do, as shown in [3].

Also note that the construction $\mathcal{N} \circ T_{\mathrm{CNF}} \circ \mathcal{E}$ converts any 1-occurrence net into an equivalent net without arcweights.

Table 2. Corresponding properties

Event structures	Propositional theories and nets	Petri nets
rooted	(>0, any)	(any, any)
singular	(1, any), (any, 0)	(≤ 1, any), (any, 0)
manifestly conjunctive	(any, ≤ 1)	(any, ≤ 1)
finite conflict	(finite, any)	(any, any)
binary conflict	(≤ 2, any)	(≤ 2, any)

4.4 Comparing Models

It is interesting to see how three important properties of event structures correspond with structural properties of Petri nets. Call an event structure *manifestly conjunctive* if for every set of events Y there is at most one set X with $X \vdash Y$. Every conjunctive event structure can be made manifestly conjunctive by deleting, for every set Y, all but the smallest X for which $X \vdash Y$. The property of conjunctivity implies that such a smallest X exists. This normalisation preserves transition equivalence, and all properties of Definition 6.

When E is an event structure satisfying any of the properties from the left column of Table 2, then $T_{\mathrm{CNF}}(\mathrm{E})$ satisfies the corresponding properties from the middle column. These are to be read as cardinality restrictions on the sets X and Y, respectively, in each of its clauses $X \Rightarrow Y$. For instance, if E is singular, $T_{\mathrm{CNF}}(\mathrm{E})$ has only clauses $X \Rightarrow Y$ with $|X| = 1$ or $|Y| = 0$. Furthermore, if T is a rooted propositional theory with finite conflict satisfying any of the properties of the middle column, then $\mathcal{N}(T)$ satisfies these same properties, but now they are cardinality restrictions on the number of outgoing and incoming arcs, respectively, for every place in $\mathcal{N}(T)$. Finally, any net satisfying some of the restrictions from the middle column, or even the weaker variants from the right column, translates to an event structure satisfying the corresponding restrictions on the left. This remains true if any place with n incoming arcs and k initial tokens is deemed to satisfy the restriction "($\leq k+n$, $\leq n$) or ($k+n+1$, 0)".

5 Related Work

A *bundle event structure,* as studied in LANGERAK [7], can in our framework best be understood as a propositional theory. Using the translation of Sect. 4 it maps to a special kind of stable event structure [10]. Langerak's notion of an *extended* bundle event structure on the other hand does not correspond to an event structure as in [9,10]. Here the symmetric binary conflict relation $\#$ is replaced by an asymmetric counterpart $\rightsquigarrow$. When $a \rightsquigarrow b$, the event b can happen regardless of a, and a is initially enabled as well; however, as soon as b happens, a is blocked. Asymmetric conflict $a \rightsquigarrow b$ can be translated into our framework as $\{b\} \vdash \{a, b\}$. (As this translation introduces impurity, for its correctness it is necessary to consider the transition relation of Definition 2.) Thus, extended bundle event structures are subsumed by our event structures too.

The same can be said for the *extended dual event structures* of KATOEN [6]. Here the crucial feature is the symmetric and irreflexive *interleaving relation,* modelling mutual exclusion of events, i.e. disallowing them to overlap in time. As in Fig. 5, this can be modelled in our framework as $\{a\} \vdash \{a,b\}$ $\{b\} \vdash \{a,b\}$. Using similar techniques, we believe it is also possible to embed the causal automata of GUNAWARDENA [4] in our framework.

BOUDOL [1] provides translations between a class of 1-occurrence nets, the *flow nets,* and a class of *flow event structures* that fall in expressive power between the prime and the stable event structures of [10]. His correspondence extends the correspondence between safe occurrence nets and prime event structures due to [8]. As Boudol's translations preserve the notions of event (=transition) and configuration, they are consistent with our approach. Our translations can be regarded as an extension of the work of [1] to general Petri nets.

Another translation between Petri nets and a new model of event structures has been provided in HOOGERS, KLEIJN & THIAGARAJAN [5], albeit for systems without autoconcurrency only. Their event structures are essentially families of configurations with a step transition relation between them. The translations of [5] are quite different from ours: even on 1-occurrence nets an individual transition may correspond to multiple events. We conjecture the two approaches are equivalent under a suitable notion of history preserving bisimulation.

References

1. G. BOUDOL (1990): *Flow event structures and flow nets.* In I. Guessarian, editor: *Semantics of Systems of Concurrent Processes, Proceedings LITP Spring School on Theoretical Computer Science,* La Roche Posay, France, LNCS 469, pp. 62–95.
2. R.J. VAN GLABBEEK & G.D. PLOTKIN (1995): *Configuration structures (extended abstract).* In D. Kozen, editor: Proc. LICS'95, San Diego, USA, pp. 199–209.
3. R.J. VAN GLABBEEK & G.D. PLOTKIN (2004): *Configuration structures, event structures and Petri nets.* Draft at `http://theory.stanford.edu/~rvg/event/`.
4. J. GUNAWARDENA (1992): *Causal automata.* TCS 101, pp. 265–288.
5. P.W. HOOGERS, H.C.M. KLEIJN & P.S. THIAGARAJAN (1996): *An event structures semantics for general Petri nets.* TCS 153, pp. 129–170.
6. J.-P. KATOEN (1996): *Quantitative and Qualitative Extensions of Event Structures,* PhD thesis, Department of Computer Science, University of Twente.
7. R. LANGERAK (1992): *Transformations and Semantics for LOTOS.* PhD thesis, Department of Computer Science, University of Twente.
8. M. NIELSEN, G.D. PLOTKIN & G. WINSKEL (1981): *Petri nets, event structures and domains, part I.* TCS 13(1), pp. 85–108.
9. G. WINSKEL (1987): *Event structures.* In W. Brauer, W. Reisig & G. Rozenberg, editors: *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proceedings of an Advanced Course,* Bad Honnef, September 1986, LNCS 255, Springer, pp. 325–392.
10. G. WINSKEL (1989): *An introduction to event structures.* In J.W. de Bakker, W.P. de Roever & G. Rozenberg, editors: REX *School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* Noordwijkerhout, The Netherlands, May/June 1988, LNCS 354, Springer, pp. 364–397.

Optimal Preemptive Scheduling for General Target Functions

Leah Epstein[1*] and Tamir Tassa[2]

[1] School of Computer Science, The Interdisciplinary Center, Herzliya, Israel.
lea@idc.ac.il

[2] Department of Mathematics and Computer Science, The Open University, Ramat Aviv, Tel Aviv, and Department of Computer Science, Ben Gurion University, Beer Sheva, Israel.
tamirta@openu.ac.il

Abstract. We study the problem of optimal preemptive scheduling with respect to a general target function. Given n jobs with associated weights and $m \leq n$ uniformly related machines, one aims at scheduling the jobs to the machines, allowing preemptions but forbidding parallelization, so that a given target function of the loads on each machine is minimized. This problem was studied in the past in the case of the makespan. Gonzalez and Sahni [7] and later Shachnai, Tamir and Woeginger [12] devised a polynomial algorithm that outputs an optimal schedule for which the number of preemptions is at most $2(m-1)$. We extend their ideas for general symmetric, convex and monotone target functions. This general approach enables us to distill the underlying principles on which the optimal makespan algorithm is based. More specifically, the general approach enables us to identify between the optimal scheduling problem and a corresponding problem of mathematical programming. This, in turn, allows us to devise a single algorithm that is suitable for a wide array of target functions, where the only difference between one target function and another is manifested through the corresponding mathematical programming problem.

1 Introduction

We are interested in the problem of optimal preemptive scheduling with respect to a general target function. The data in such problems consists of:

- n jobs, $\mathcal{J} = \{J_i\}_{1 \leq i \leq n}$, where job J_i has a weight w_i and $w_1 \geq w_2 \geq \cdots \geq w_n > 0$.
- m machines, $\mathcal{M} = \{M_j\}_{1 \leq j \leq m}$, $m \leq n$, where machine M_j has speed s_j and $1 = s_1 \geq s_2 \geq \cdots \geq s_m > 0$.

If a job of weight w runs on a machine of speed s, its processing time will be w/s. A non-preemptive schedule of the jobs to the machines is a function $\sigma : \mathcal{J} \to \mathcal{M}$.

* The research of this author supported in part by Israel Science Foundation grant 250/01.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 562–573, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In such schedules, once a job started its process on a given machine, it is executed continuously until completion. We, however, are interested here in preemptive schedules, where a job's execution may be terminated and then resumed later, possibly on a different machine.

Definition 1. *A preemptive schedule is a vector* $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)$ *where* σ_j *is the schedule on* M_j, $1 \leq j \leq m$. *The machine schedule* σ_j *takes the form of a pair of sequences,* $\sigma_j = (\tau_j, \eta_j)$, *where* τ_j *is a sequence of strictly increasing times,* $0 = \tau_{j,0} < \tau_{j,1} < \cdots < \tau_{j,k_j}$, *and* η_j *is a sequence of indices, i.e.,* $\eta_j = (\eta_{j,1}, \ldots, \eta_{j,k_j})$ *where* $\eta_{j,k} \in \{0, 1, \ldots, n\}$ *for all* $1 \leq k \leq k_j$. *Such a schedule means that* M_j *processes* $J_{\eta_{j,k}}$ *in time interval* $[\tau_{j,k-1}, \tau_{j,k})$, *for all* $1 \leq k \leq k_j$, *unless* $\eta_{j,k} = 0$ *in which case* M_j *is idle during the corresponding time interval.*

The schedule is *legal* if the same job is never scheduled to be processed at the same time by two different machines (namely, parallelization is forbidden). The schedule is *complete* if for every given job, the sum over all machines of its processed parts amounts to its weight, i.e.,

$$\sum_{(j,k):\ \eta_{j,k}=i} (\tau_{j,k} - \tau_{j,k-1}) \cdot s_j = w_i \quad \text{for all} \quad 1 \leq i \leq n \ . \tag{1}$$

Hereinafter we consider only complete and legal schedules.

For a given schedule, $\boldsymbol{\sigma}$, we let $\boldsymbol{\lambda}(\boldsymbol{\sigma}) = (\lambda_1, \ldots, \lambda_m)$ denote the corresponding vector of *loads,* where $\lambda_j := \tau_{j,k_j}$ is the *time* in which M_j finishes running under the schedule $\boldsymbol{\sigma}$. One usually seeks schedules that minimize the value of some target function of the loads,

$$f(\boldsymbol{\sigma}) = f(\boldsymbol{\lambda}(\boldsymbol{\sigma})) = f(\lambda_1, \ldots, \lambda_m) \ , \tag{2}$$

where f is typically a convex, symmetric and monotonically non-decreasing function with respect to its arguments.

For a given target function f, we let f_{opt} denote its optimal value, i.e., $f_{opt} = \min_{\boldsymbol{\sigma}} f(\boldsymbol{\sigma})$. The usual choice is the makespan, $f = \max$. This case was studied in [11,10,7,12,5]. Liu and Yang [11] introduced bounds on the cost of optimal schedules. Horvath, Lam and Sethi proved that the optimal cost is indeed the maximum of those bounds by constructing an algorithm that uses a large number of preemptions. Gonzalez and Sahni [7] devised a polynomial algorithm that outputs an optimal schedule for which the number of preemptions is at most $2(m-1)$. This number of preemptions was shown to be optimal in the sense that there exist inputs for which every optimal schedule involves that many preemptions. This algorithm was later generalized and simplified for jobs of limited splitting constraints by Shachnai, Tamir and Woeginger [12]. In this paper we extend the ideas of [12] for general symmetric, convex and monotone target functions. This general approach offers several benefits over the study of the particular makespan problem. By looking at the problem from a more general perspective, we are able to distill the underlying principles on which the algorithm of [12] is based. This approach enables us to identify between the

optimal scheduling problem and a problem of mathematical programming. This, in turn, allows us to devise a single algorithm that is suitable for a wide array of target functions, where the only difference between one target function and another is manifested through the corresponding mathematical programming problem. Lastly, this approach facilitates the presentation and analysis of the algorithm.

The paper begins with a study of properties of optimal schedules, Section 2. We show that when the target function is convex, symmetric and monotone, there always exist optimal schedules of a relatively simple structure. Specifically, there always exist optimal schedules where the loads on faster machines are greater than or equal to the loads on slower machines and there are no idle times, Proposition 1. As a consequence of this characterization of (some) optimal schedules, we define a mathematical program (i.e., a problem of minimizing a multivariate target function in a bounded polyhedron) whose solution is the set of machine loads of an optimal schedule, Theorem 1. Section 3 is then dedicated to the presentation and analysis of Algorithm 1. This algorithm receives as an input a set of machine loads from the polyhedron that corresponds to the equivalent mathematical program, and it outputs a complete and legal preemptive schedule with those machine loads. Hence, if one runs this algorithm with the set of machine loads that solved the mathematical program, one gets an optimal preemptive schedule to the original problem, Theorem 3. In Appendix 4.2 we illustrate the algorithm with an example.

The problem of finding an optimal preemptive schedule is therefore separated into two independent stages. In the first stage we write down the corresponding mathematical program and solve it. In that mathematical program we aim at minimizing the function (2) in a bounded polyhedron in $\mathcal{R}^m$ that reflects a set of linear constraints that manifest our demand for completeness and legality of the schedule. After solving this mathematical program, we face an algorithmic problem: finding a preemptive schedule whose loads equal the solution of the mathematical program. This is achieved by Algorithm 1. This second stage is general in the sense that it is independent of the choice of the target function.

After presenting and studying the general algorithm, we derive explicit results for specific target functions, Section 4. In Section 4.1 we devise a polynomial time algorithm for the solution of the mathematical program when the target function is the ℓ_p-norm, $f(\lambda_1, \ldots, \lambda_m) = \left(\sum_{j=1}^{m} \lambda_j^p\right)^{1/p}$, $1 \le p < \infty$. This target function was studied in the past in the non-preemptive setting [1,6]. We show that by taking the limit $p \uparrow \infty$ the algorithm solves also the makespan minimization problem and that the minimal makespan of optimal preemptive schedules agrees with the value that was derived in [7,12]. In Section 4.2 we continue to explore the threshold cost function, $f(\lambda_1, \ldots, \lambda_m) = \sum_{j=1}^{m} \max(\lambda_j, c)$ where $c > 0$ is some constant threshold, and devise a polynomial time algorithm for its solution. This target function was also studied in the past for non-preemptive scheduling [2,3, 4]. We note that an algorithm due to Hochbaum and Shanthikumar [8] may be applied in order to solve the mathematical program in polynomial time whenever

the target function is separable, i.e., $f(\lambda_1, \ldots, \lambda_m) = \sum_{j=1}^{m} g(\lambda_j)$. It should be noted that even though the ℓ_p-norm target function, with $p < \infty$, and the threshold target function, are separable, the algorithms that we offer for these cases are simpler and more efficient than the general algorithm in [8]. More details are given in the full version.

As a concluding remark we recall that the non-preemptive versions of the above problems are typically strongly NP-hard. Approximation schemes for the makespan problem were given by Hochbaum and Shmoys [9]. The papers [1,6] offer approximations schemes for a wide class of target functions, including the ℓ_p-norms.

Most of the proofs are omitted from the body of the text; they may be found in the full version of this paper.

2 Properties of Optimal Schedules

In this section we derive some qualitative properties of optimal schedules for general symmetric and monotone target functions.

Proposition 1. *There exist optimal schedules in which the loads λ_j are monotonically non-increasing and there are no holes (i.e., no idle times on a machine after which it resumes processing).*

Hereinafter we concentrate only on optimal schedules that comply with Propositions 1. We define the weight on M_j as $\mu_j = s_j \lambda_j$. Namely, the weight on machine M_j under a schedule σ represents the total weight of job parts that are scheduled by σ to be processed on M_j. We also define the following:

$$W_k = \begin{cases} \sum_{j=1}^{k} w_j & 1 \le k \le m-1 \\ \sum_{j=1}^{n} w_j & k = m \end{cases} \tag{3}$$

With these definitions, we state the following key proposition.

Proposition 2. *In optimal schedules that comply with Proposition 1*

$$\sum_{j=1}^{k} \mu_j \ge W_k \quad , \quad 1 \le k \le m-1 \quad \textit{while} \quad \sum_{j=1}^{m} \mu_j = W_m \ . \tag{4}$$

Proof. As the inequality in (4) is just the completeness requirement, we focus on proving the inequality in (4) for an arbitrary value of $1 \le k \le m-1$. Let R_ℓ be defined as the union of all time intervals in which *exactly* ℓ of the k largest jobs, $\mathcal{J}_k := \{J_i\}_{1 \le i \le k}$, are running. Namely, if $t \in R_\ell$, there exists a subset of ℓ jobs $\{J_{i_1}, \ldots, J_{i_\ell}\} \subset \mathcal{J}_k$ that are scheduled to run on ℓ machines at time t, while the remaining $k - \ell$ jobs in $\mathcal{J}_k$ are not being processed at that time. In view of Proposition 1, the entire schedule is embedded in the time interval $[0, \lambda_1)$.

We break up this interval into a disjoint union $[0, \lambda_1) = \bigcup_{\ell=0}^{k} R_\ell$. Proposition 1 implies that

$$\bigcup_{\ell=j}^{k} R_\ell \subset [0, \lambda_j) \quad , \quad 1 \le j \le k \, . \tag{5}$$

Let r_ℓ denote the amount of work that was done on the k largest jobs during R_ℓ. Then, as the schedule is complete, $\sum_{\ell=1}^{k} r_\ell = W_k$. On the other hand, since the schedule is legal, r_ℓ may not exceed the duration of R_ℓ times the sum of speeds of the ℓ fastest machines, i.e., $r_\ell \le S_\ell \cdot |R_\ell|$, where hereinafter $S_\ell = \sum_{j=1}^{\ell} s_j$. Hence, by (5), $W_k \le \sum_{\ell=1}^{k} \left(\sum_{j=1}^{\ell} s_j \right) \cdot |R_\ell| = \sum_{j=1}^{k} s_j \cdot \left(\sum_{\ell=j}^{k} |R_\ell| \right) \le \sum_{j=1}^{k} s_j \lambda_j = \sum_{j=1}^{k} \mu_j$. □

Finally, we state our main result.

Theorem 1.

$$f_{opt} = \min_{\Omega} f\left(\frac{\mu_1}{s_1}, \ldots, \frac{\mu_m}{s_m} \right) \tag{6}$$

where $\Omega \subset (\mathcal{R}^+)^m$ is the bounded polyhedron of all nonnegative weights μ_j that satisfy the legality and completeness constraints (4).

Proof. Let f_{min} denote the minimum of the optimization problem (6) under the constraints in (4) (This optimization problem is a mathematical program to which we refer henceforth as MP). f_{min} is well defined since f is convex and Ω, the domain in which the minimum is sought, is closed and convex. Proposition 2 imply that $f_{opt} \ge f_{min}$. Since Algorithm 1 in the next section produces a complete and legal preemptive schedule with weights $\{\mu_j\}_{1 \le j \le m}$ for *any* $(\mu_1, \ldots, \mu_m) \in \Omega$, we infer that $f_{opt} = f_{min}$. □

3 An Optimal Scheduling Algorithm for a General Target Function

In this section we present and analyze an algorithm that, given a legal and complete allocation of weights to machines, $\{\mu_j\}_{1 \le j \le m} \in \Omega$, finds a preemptive schedule of the jobs to the machines that agrees with those weights. Hence, if we call that algorithm with a solution of the mathematical program MP, we get in return an optimal preemptive schedule.

3.1 The Algorithm

Let $\{\mu_j\}_{1 \le j \le m} \in \Omega$ be a set of nonnegative weights that satisfy the conditions in (4). Let

$$\lambda_j = \frac{\mu_j}{s_j} \quad , \quad \Lambda_j = \sum_{k=1}^{j} \lambda_k \quad , \quad 1 \le j \le m \quad \text{and} \quad \Lambda_0 = 0 \, . \tag{7}$$

Next, we define the following state functions on $[0, \Lambda_m)$: a potential function

$$\Psi(x) = \sum_{j=1}^{m} s_j \cdot \chi_{[\Lambda_{j-1}, \Lambda_j)}(x) \quad \text{where} \quad \chi_I(x) = \begin{cases} 1 & x \in I \\ 0 & x \notin I \end{cases}, \tag{8}$$

a timing function

$$\Theta(x) = \begin{cases} x - \Lambda_{j-1} \text{ if } x \in [\Lambda_{j-1}, \Lambda_j) \text{ for some } 1 \leq j \leq m \\ 0 \quad \text{otherwise} \end{cases}, \tag{9}$$

and an indicator function

$$\Gamma(x) = \begin{cases} j \text{ if } x \in [\Lambda_{j-1}, \Lambda_j) \text{ for some } 1 \leq j \leq m \\ 0 \text{ otherwise} \end{cases} \tag{10}$$

(see Figures 1-3).

Those three functions will always be examined as a triplet, along some interval $[a, b)$: the value of the indicator function $\Gamma(x)$ will indicate the machines under consideration; the range of values of $\Theta(x)$ along that interval will indicate the time segments on which we focus; and the corresponding integrals of the potential function Ψ represents the work that can be done by the corresponding machines in the corresponding time segments. For example, let us consider the state functions in Figures 1-3. They represent a problem with 4 machines, of speeds 1,0.8,0.6 and 0.3, where the allocated loads are 10,8,6 and 3 (e.g., the length of the schedule on M_3 is 6 time units). Consider the interval [9,12). Then, looking at Γ^1 and Θ^1, this interval corresponds to time segment [9,10) on M_1 and time segment [0,2) on M_2. The corresponding integral $\int_9^{10} \Psi^1(x)dx$ gives the amount of work that can be accomplished by M_1 in time segment [9,10), and $\int_{10}^{12} \Psi^1(x)dx$ gives the amount of work that can be accomplished by M_2 in time segment [0,2).

Algorithm 1 produces a preemptive schedule σ of $\mathcal{J}$ on $\mathcal{M}$ such that the weight on machine M_j equals μ_j. The jobs are scheduled one by one, from the largest to the smallest. We schedule a job by associating with it an interval $[a, b) \subset [0, \Lambda_m)$ such that $\int_a^b \Psi(x)dx$ equals the weight of that job, and $\Theta(x)$ is injective along that interval (in order to avoid parallelization). The main idea is to always schedule the next job to the slowest possible machine in the latest possible time; namely, we aim at finding an interval $[a, b) \subset [0, \Lambda_m)$ with the maximal a that can still accommodate that job. Once we found such an interval, we record the resulting schedule of the job by looking at the values of Γ and Θ along $[a, b)$. Then, we have to mark the interval as being occupied. Instead of doing that, we simply remove that interval from the graphs of the three state functions and left shift the "tail" of those graphs, $[b, \infty)$, by $b - a$, in order to close the gap that was created by this removal.

In carrying out the above procedure, we define for each point a the point $b = End(a)$ that stands for the maximal point b such that $\Theta(x)$ is injective on $[a, b]$. As $\Theta(x)$ has a very simple piecewise linear form and it retains that form after each "cut-and-shift" operation that corresponds to scheduling a job, it is very simple to compute the function $b = End(a)$.

Algorithm 1 *1. Initialize Ψ, Θ and Γ according to (7)–(10).*
2. $i = 1$ (current job number).
3. Define End(a) for all $a \in [0, \Lambda_m)$ in the following manner:

$$End(a) = \min(a + \Lambda_j - \Lambda_{j-1}, \Lambda_{j+1}) \quad \forall a \in [\Lambda_{j-1}, \Lambda_j) \quad , \quad 1 \le j \le m \ , \tag{11}$$

where, for the sake of the last interval, we take $\Lambda_{m+1} = \Lambda_m$.
4. Find the maximal value of a for which

$$\int_a^{b=End(a)} \Psi(x)dx = w_i \ . \tag{12}$$

5. Decompose the interval $[a, b)$ into a disjoint union of intervals,

$$[a, b) = \bigcup_{r=1}^{\ell} [a_{r-1}, a_r) \tag{13}$$

where $a_0 = a$, $a_\ell = b$, Γ is constant along $[a_{r-1}, a_r)$, say $\Gamma|_{[a_{r-1}, a_r)} = j_r$, and $j_1 < j_2 < \cdots < j_\ell$.
6. Compute

$$w_{i,r} = \int_{a_{r-1}}^{a_r} \Psi(x)dx \quad , \quad 1 \le r \le \ell \ . \tag{14}$$

7. Break up J_i into ℓ parts, $\{J_{i,r}\}_{1 \le r \le \ell}$, where the weight of $J_{i,r}$ is $w_{i,r}$, $1 \le r \le \ell$.
8. Schedule $J_{i,r}$ to run on M_{j_r} in time interval $[\Theta(a_{r-1}), \Theta(a_r))$, $1 \le r \le \ell$.
9. Remove the interval $[a, b)$ from Ψ, Θ and Γ. More specifically, apply on all three functions the following operator:

$$U_{[a,b)}\Phi := \Phi \cdot \chi_{[0,a)} + L_{b-a}\left\{\Phi \cdot \chi_{[b,\infty)}\right\} \ , \tag{15}$$

where L_d is the d-left shift operator, i.e., $L_d\Phi(x) = \Phi(x + d)$.
10. Update m to indicate the number of discontinuities in the modified timing function Θ and set Λ_j, $1 \le j \le m$, to be the corresponding jth discontinuity.
11. $i = i + 1$.
12. If $i > n$ stop. Else go to Step 3.

The reader is referred to Appendix 4.2 where the algorithm is exemplified.

3.2 Analysis

In this section we prove the validity of the algorithm. Hereinafter, whenever necessary to distinguish between two subsequent rounds, we use the superscript i to denote the values of the algorithm variables during the ith round in the algorithm, $1 \le i \le n$. Namely, $\Psi^i(x)$, $\Theta^i(x)$ and $\Gamma^i(x)$ are the three state functions during the ith round (before they are being updated in Step 9), m^i is the number of discontinuities of $\Theta^i(x)$ while $\{\Lambda_j^i\}_{1 \le j \le m^i}$ are those discontinuities

(with $\Lambda_0^i = 0$), and $End^i(a)$ is the function that is defined in (11) at the ith round. We also define $\Omega^i = [0, \Lambda_{m^i}^i)$ to be the support of the state functions in round i, $\Omega_j^i = [\Lambda_{j-1}^i, \Lambda_j^i)$, $1 \le j \le m^i$, be the decomposition of Ω^i into intervals of continuity of $\Theta^i(x)$, and $\lambda_j^i = |\Omega_j^i|$.

The timing function and its continuity intervals play a significant role in the analysis of Algorithm 1. The next two lemmas provide important information about $\Theta^i(x)$.

Lemma 1. *(i) The timing function is linear on each continuity interval, i.e.,*

$$\Theta^i(x) = \begin{cases} x - \Lambda_{j-1}^i & \text{if } x \in \Omega_j^i \text{ for some } 1 \le j \le m^i \\ 0 & \text{otherwise} \end{cases} . \tag{16}$$

(ii) λ_j^i, $1 \le j \le m^i$, form a non-increasing sequence for all i.

Assume that round i^* was the first round in which we selected in Step 4 a value of a with $End^{i^*}(a) = \Lambda_{m^{i^*}}^{i^*}$ (namely, this is the first round in which the sliding window went all the way to the right end point of the current support, $[0, \Lambda_{m^{i^*}}^{i^*})$, of the three state functions). It is not hard to see that, as the jobs are ordered in a non-increasing order according to their weight, the same will happen in all subsequent rounds, i.e., $End^i(a) = \Lambda_{m^i}^i$ for all $i \ge i^*$. We refer to the first $i^* - 1$ rounds in the execution of the algorithm as *Phase 1,* while rounds i^* through n constitute *Phase 2.* With this terminology, we proceed as follows.

Lemma 2. *During Phase 1 the number of Θ^i-continuity intervals always decreases by one. Namely, for all i, $1 \le i \le i^*$, $m^i = m - i + 1$. Consequently, Phase 1 lasts no more than $m - 1$ rounds.*

The next fundamental proposition is crucial for the justification of Algorithm 1.

Proposition 3. *In all rounds, the set of values of a that satisfy requirement (12) in Step 4 of the algorithm is nonempty and it has a maximum.*

Theorem 2. *Algorithm 1 generates a complete and legal schedule.*

Proof. The algorithm is well defined in view of Lemma 1 and Proposition 3. This implies the completeness of the resulting schedule since each job is assigned time shares on the machines that enable its completion. The schedule is legal since *End(a)* is defined so that the timing function $\Theta^i(x)$ is one-to-one along the interval $[a, End(a))$. □

In view of all of the above, we reach our final statement regarding Algorithm 1.

Theorem 3. *Algorithm 1 outputs an optimal preemptive schedule when the input $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$ is a solution of the corresponding mathematical program* MP, *namely, when it minimizes (6) under (4).*

It may be shown that the number of preemptions that are enforced by the algorithm is bounded by $2(m - 1)$. This was shown to be minimal for some inputs for the makespan minimization problem [7].

4 Examples of Target Functions

4.1 The ℓ_p-Norm

Here we present a polynomial time algorithm that constructs an optimal solution $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$ to MP where f is the ℓ_p-norm. The run time of the algorithm is $O(m^2)$. After presenting the algorithm, we prove that its output, $\boldsymbol{\mu}$, is in Ω and that it is a minimal point in Ω. Even though this section concentrates on $1 < p < \infty$, the results presented herein apply equally to $p = 1$ and $p = \infty$ by taking the corresponding limit. Moreover, when $p = \infty$, we recover an explicit expression for the optimal makespan that was derived in [7].

Algorithm 2 *1. Set $t = 0$ and $k_t = 0$ (at each stage k_t equals the number of values μ_j that were already determined).*

2. For every $k_t + 1 \leq k \leq m$, compute

$$q_k = (W_k - W_{k_t})/S_p[k_t + 1 : k] \quad \textit{where} \quad S_p[a : b] = \sum_{j=a}^{b} s_j^{p/(p-1)} \ . \tag{17}$$

and set k_{t+1} to be the (minimal) value of k for which q_k is maximal.

3. For all $k_t + 1 \leq j \leq k_{t+1}$, set

$$\mu_j = s_j^{p/(p-1)} \cdot (W_{k_{t+1}} - W_{k_t})/S_p[k_t + 1 : k_{t+1}] \ . \tag{18}$$

4. If $k_{t+1} < m$ set $t = t + 1$ and go to Step 2.

We note that the algorithm solves also the extremal cases $p = 1$ and $p = \infty$. When $p = \infty$, the powers $p/(p-1)$ need to be understood as 1. As for $p = 1$, let b denote the number of machines of maximal speed, i.e., $s_j = 1$ for $1 \leq j \leq b$ and $s_j < 1$ for $b < j \leq m$. When $p \downarrow 1$, the powers $p/(p-1) \uparrow \infty$. Hence, $s_j^{p/(p-1)} = 1$ for $1 \leq j \leq b$ and zero for $b < j \leq m$. As a consequence, by (18), the machines which are not among the fastest, M_j, $b < j \leq m$, will be assigned nothing, $\mu_j = 0$, and the entire weight will be spread among the b fastest machines. The manner in which the total weight will be spread among those machines depends on the data but is insignificant because the ℓ_1-norm does not distinguish between such assignments. Such schedules are of-course optimal.

Lemma 3. *Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$ be the solution that Algorithm 2 returned and let $\{k_i\}_{0 \leq i \leq t+1}$ be the corresponding sequence of indices that were identified during the execution of the algorithm. Then $\boldsymbol{\mu} \in \Omega$. Namely, it satisfies (4). Moreover, the set of indices for which (4) holds with equality is exactly $\{k_i\}_{1 \leq i \leq t}$.*

Next, we claim that $\boldsymbol{\mu}$ is optimal for $1 < p < \infty$.

Theorem 4. *Let $\boldsymbol{\mu}' = \{\mu'_j\}_{1 \leq j \leq m}$ be an optimal solution of MP where $1 < p < \infty$. Then $\boldsymbol{\mu}' = \boldsymbol{\mu}$.*

The optimality of $\boldsymbol{\mu}$ for $p = \infty$ is a direct consequence of Lemma 4 as the makespan is the limit of the ℓ_p-norms when $p \uparrow \infty$. In this case, we may even derive an explicit expression for the optimal makespan. The solution $\boldsymbol{\mu}$ that Algorithm 2 outputs satisfies $\max_{1\le k\le m} \frac{W_k}{S_p[1:k]} = \frac{\mu_1}{s_1^{1+\frac{1}{p-1}}} \ge \frac{\mu_2}{s_2^{1+\frac{1}{p-1}}} \ge \cdots \ge \frac{\mu_m}{s_m^{1+\frac{1}{p-1}}}$. When $p = \infty$, this translates into $\max_{1\le k\le m} \frac{W_k}{S_\infty[1:k]} = \max_{1\le k\le m} \frac{W_k}{S_k} = \frac{\mu_1}{s_1} \ge \frac{\mu_2}{s_2} \ge \cdots \ge \frac{\mu_m}{s_m}$; here, as before, $S_k = \sum_{j=1}^{k} s_j$. Therefore, the optimal makespan is $\max\left(\frac{\mu_1}{s_1}, \ldots, \frac{\mu_m}{s_m}\right) = \max_{1\le k\le m} \frac{W_k}{S_k}$, as the result of [7].

4.2 Threshold Cost Functions

Here we study the target function $f(\mu_1, \ldots, \mu_m) = \sum_{j=1}^{m} \max\left(\frac{\mu_j}{s_j}, c\right)$. This case, also known as *extensible bin packing* [2,3,4], describes a scenario in which a fixed payment is due up-front for c time units in each machine, whether they have been used or not, and, in addition, to any excessive time that was used beyond the fixed threshold in any of the machines.

Algorithm 3 computes an optimal solution $\boldsymbol{\mu} \in \Omega$ to MP when the target function f is as above. Here, as before, $S_k = \sum_{j=1}^{k} s_j$ and W_k is as in (3).

Algorithm 3 *1. Set* $\mu_k = 0$ *for all* $1 \le k \le m$ *and* $W = W_m = \sum_{j=1}^{n} w_j$.
2. If $W \le c \cdot s_1$ *set* $\mu_1 = W$ *and stop.*
3. Set $\mu_1 = \max\{c \cdot s_1, \max_{1\le k\le m}(W_k - c \cdot S_k + c \cdot s_1)\}, W = W - \mu_1$.
4. For $k = 2$ *to* $k = m$ *do:*
a) If $W > c \cdot s_k$ *then* $\mu_k = c \cdot s_k$ *and* $W = W - c \cdot s_k$.
b) Else $\mu_k = W$ *and* $W = 0$.

Theorem 5. *The solution* $\boldsymbol{\mu}$ *that Algorithm 3 produces gives a minimum to the threshold cost function in* Ω.

References

1. N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling,* 1:1:55–66, 1998.
2. E. G. Coffman, Jr. and George S. Lueker. Approximation algorithms for extensible bin packing. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01),* pages 586–588, 2001.
3. P. Dell'Olmo, H. Kellerer, M. G. Speranza, and Zs. Tuza. A 13/12 approximation algorithm for bin packing with extendable bins. *Information Processing Letters,* 65(5):229–233, 1998.
4. P. Dell'Olmo and M. G. Speranza. Approximation algorithms for partitioning small items in unequal bins to minimize the total size. *Discrete Applied Mathematics,* 94:181–191, 1999.
5. T. Ebenlendr and J. Sgall. Optimal and online preemptive scheduling on uniformly related machines. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computing Science (STACS-04),* LNCS 2996:199-210, 2004.

6. L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica,* 39(1):151–162, 2004.
7. T. Gonzalez and S. Sahni. Preemptive scheduling of uniform processor systems. *Journal of the ACM,* 25(1):92–101, 1978.
8. D. S. Hochbaum and J. G. Shanthikumar. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM,* 37(4):843–862, 1990.
9. D. S. Hochbaum and D. B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing,* 17(3):539–551, 1988.
10. E. C. Horvath, S. Lam, and R. Sethi. A level algorithm for preemptive scheduling. *Journal of the ACM,* 24(1):32–43, 1977.
11. J. W. S. Liu and A. T. Yang. Optimal scheduling of independent tasks on heterogeneous computing systems. In *Proceedings ACM National Conference,* volume 1, pages 38–45. ACM, 1974.
12. H. Shachnai, T. Tamir, and G. J. Woeginger. Minimizing makespan and preemption costs on a system of uniform machines. In *Proc. of the 10th Annual European Symposium on Algorithms (ESA2002),* pages 859–871, 2002.

An Example

Consider a scenario with $m = 4$ machines, the speeds of which are $(s_1, s_2, s_3, s_4) = (1, .8, .6, .3)$. Assume that the set of job weights dictates machine loads
$(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (10, 8, 6, 3)$ (when $p = 2$ and the global minimum of the ℓ_2-norm in Ω coincides with the global minimum in $\mathcal{R}^m$, it may be shown that the machine loads indeed relate to each other like the machine speeds). Then the three state functions will be initially as described in Figures 1-3. There are $m^1 = 4$ jump discontinuities in the timing function, $\Theta^1(x)$, at $(\Lambda_1^1, \Lambda_2^1, \Lambda_3^1, \Lambda_4^1) = (10, 18, 24, 27)$.

We proceed to describe the scheduling of the first job. Assume that $w_1 = 9$. It is not hard to see that the window in which it fits, Step 4, is [5,15) (i.e., $a = 5$). The values of the indicator and timing functions, Γ^1 and Θ^1, along this window imply that J_1 will be scheduled to run on M_2 in time interval [0,5) and on M_2 in [5,10). After scheduling J_1 we remove the occupied time slots by applying the cut-and-shift operator $U_{[5,15)}$. The function $\Theta^2(x)$ has $m^2 = 3$ jump discontinuities at $(\Lambda_1^2, \Lambda_2^2, \Lambda_3^2) = (8, 14, 17)$.

Next, assume that the second job is of size $w_2 = 7$. Here, the value of a in Step 4 is $a = 1$ and the corresponding window is [1,9). Therefore, the values of Γ^2 and Θ^2 along this interval imply that J_2 will be scheduled to run on M_3 during [0,1), on M_1 during [1,5) and on M_2 during [5,8). After applying $U_{[1,9)}$, $\Theta^3(x)$ has $m^3 = 2$ jump discontinuities at $(\Lambda_1^3, \Lambda_2^3) = (6, 9)$.

We note that if $w_3 < 0.9$, then J_3 will mark the beginning of *Phase 2* and the corresponding window will be completely within the last interval of continuity of Θ^3; in that case $m^4 = m^3 = 2$. If, on the other hand, $w_3 \geq 0.9$, $m^4 = 1$ and then J_4 will be the first job in *Phase 2*.

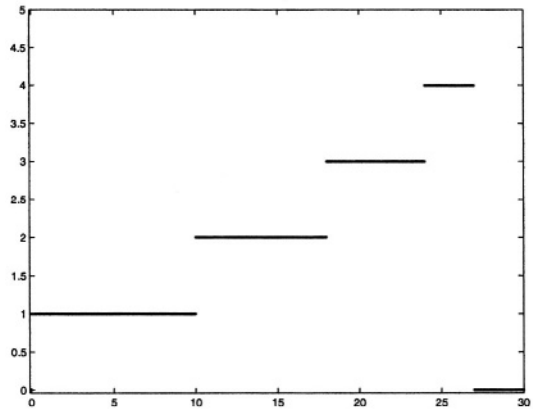


Fig. 1. $\Gamma^1(x)$

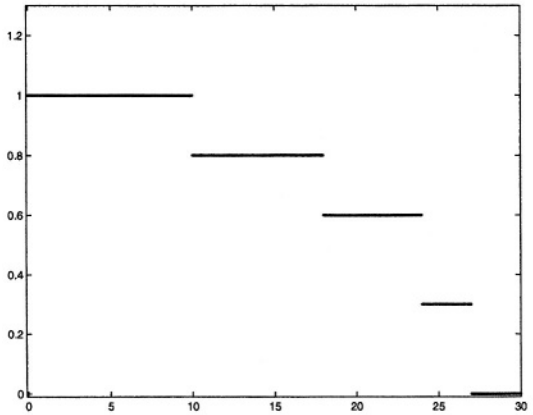


Fig. 2. $\Psi^1(x)$

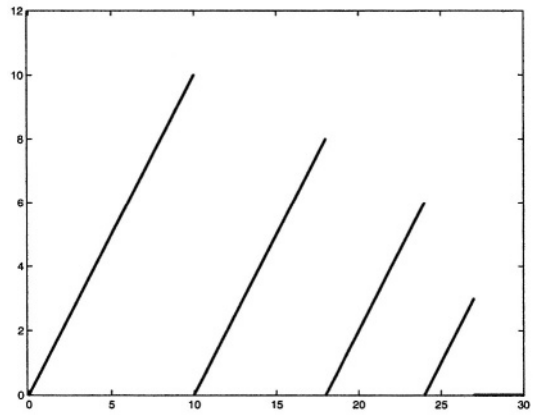


Fig. 3. $\Theta^1(x)$

The Price of Anarchy for Polynomial Social Cost*

Martin Gairing[1], Thomas Lücking[1], Marios Mavronicolas[2], and Burkhard Monien[1]

[1] Faculty of Computer Science, Electrical Engineering and Mathematics, University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany.
{gairing,luck,bm}@uni-paderborn.de
[2] Department of Computer Science, University of Cyprus, P. O. Box 20537, Nicosia CY-1678, Cyprus.
mavronic@ucy.ac.cy

Abstract. In this work, we consider an interesting variant of the well-studied *KP model* [18] for *selfish routing* that reflects some influence from the much older *Wardrop model* [31]. In the new model, user *traffics* are still *unsplittable,* while *social cost* is now the expectation of the sum, over all links, of a certain polynomial evaluated at the total *latency* incurred by all users choosing the link; we call it *polynomial social cost.* The polynomials that we consider have non-negative coefficients. We are interested in evaluating *Nash equilibria* in this model, and we use the *Price of Anarchy* as our evaluation measure. We prove the *Fully Mixed Nash Equilibrium Conjecture* for identical users and two links, and establish an approximate version of the conjecture for arbitrary many links. Moreover, we give upper bounds on the Price of Anarchy.

1 Introduction

Motivation and Framework. The *Price of Anarchy,* also known as *coordination ratio,* has been defined in a seminal work by Koutsoupias and Papadimitriou [18] as a measure of the extent to which competition approximates cooperation. In general, the Price of Anarchy is the worst-case ratio between the value of a social objective function, usually coined as *social cost,* in some equilibrium state of a system, and that of some social optimum. Usually, the equilibrium state has been taken to be that of a *Nash equilibrium* [24] – a state in which no *user* wishes to leave unilaterally its own strategy in order to improve the value of its private objective function, also known as *individual cost.* So, the Price of Anarchy represents a rendezvous of Nash equilibrium, a concept fundamental to Game Theory, with *approximation,* an ubiquitous concept in Theoretical Computer Science today (see, e.g., [30]).

* This work has been partially supported by the DFG-Sonderforschungsbereich 376 Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen, by the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS), by research funds at University of Cyprus, and by the VEGA grant No. 2/3164/23.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 574–585, 2004.
© Springer-Verlag Berlin Heidelberg 2004

The setting considered by Koutsoupias and Papadirnitriou [18], now widely known as the *KP model,* involves a network of m parallel *links* and a collection of n selfish *users* with *traffics* $w_1, \ldots, w_n$ which are *unsplittable.* The *latency* incurred at a link is the total traffic of users choosing it. *Expected latency* makes sense in case users' *strategies* are *mixed,* i.e., when they are probability distributions over the links. The individual cost of a user over a link is the expected latency it would experience had it chosen the link. In a Nash equilibrium, each user is minimizing its individual cost. The social cost is the expectation of maximum (over links) latency; the social optimum is the least possible maximum latency. Recently, there has been a lot of flourishing interest and attention into the KP model; see, e.g., [3,9,11,15,17,22]. The resulting research is overviewed in three authoritative surveys [7,12,16].

A much older theoretical model for selfish routing in transportation networks due to Wardrop [31] dates back to the 1950s. In this model, users' traffics are infinitesimally *splittable,* a feature that rules out mixed strategies from consideration. In addition, social cost is defined here as the sum of the individual costs (those being taken as the sum of latencies, determined by convex functions of the total traffics, along paths). Wardrop's model, and, in particular, the incurred Price of Anarchy, has been reinvestigated in many recent works [19,26,27,28,29].

On a middle ground, some recent research [14,20] has attempted to understand the dependence of the Price of Anarchy on the particular assumptions made in the adopted model for selfish routing. In this direction, [14,20] has each considered a hybridization of the KP model and the Wardrop model. In this paper, we further address this line of research by introducing and studying a new, yet interesting variant of the KP model that reflects some influence from the Wardrop model.

In the proposed model, traffics are unsplittable, which makes it interesting to study mixed strategies, and the network consists of parallel links as in the KP model; however, we define social cost as the expectation of a certain *sum* of link latencies, as opposed to expectation of *maximum* latency adopted in the KP model. The idea of summing (over links) comes from the Wardrop model, although social cost is defined there as the sum of individual costs (over users). An additional influence from the Wardrop model lies in modeling each latency cost function as a certain polynomial of total traffic of users choosing the link. So, social cost is now the expectation of a sum of polynomial functions, and we call it *polynomial social cost.* The polynomials that we consider have non-negative coefficients.

Restricted to *pure* Nash equilibria, our model was already studied for monotone functions in [8], and, restricted to polynomials of degree 2 in [20]. Restricted to identical users, it is a particular instance of what is known as *congestion game* [23,25]. See below for more on related work.

Contribution and Significance. We study a natural conjecture asserting that the fully mixed Nash equilibrium **F,** that is, the Nash equilibrium where each user chooses each link with non-zero probability, is the *worst* Nash equilibrium with respect to social cost:

Conjecture 1 (Fully Mixed Nash Equilibrium Conjecture). For any traffic vector $\mathbf{w}$ and for any Nash equilibrium $\mathbf{P}$, $\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{P}) \leq \mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{F})$.

In the following, we refer to this conjecture as the FMNE Conjecture. Moreover, we investigate the Price of Anarchy. Our results address the following two important special cases of the problem:

1. The case of identical users: To analyze this case, we first show that the polynomial social cost of the *fully mixed Nash equilibrium* is a certain combinatorial sum of *Stirling numbers of the second kind.* Moreover, the polynomial social cost of any (mixed) Nash equilibrium can be bounded from above by a sum of binomial cost functions, which again turns out to be certain combinatorial sum of *Stirling numbers of the second kind.* By considering terms of the same power pairwise, this enables us to prove that the polynomial social cost of any Nash equilibrium is bounded by the polynomial social cost of the fully mixed Nash equilibrium for the case of 2 links. This implies the validity of the *Fully Mixed Nash Equilibrium Conjecture* for this special case, in our model. Equipped now with the validity of the Fully Mixed Nash Equilibrium Conjecture, we prove an *exact* bound of $\frac{1}{2}(2^{d-1}+1)$ on the Price of Anarchy for the special case where the polynomial is just the dth power, for any integer $d \geq 1$. We also obtain an upper bound on the Price of Anarchy for the general case of polynomials with non-negative coefficients.
   We then generalize these results to the case of many links. Using similar techniques as in the case of 2 links, we prove that the polynomial social cost of any arbitrary (mixed) Nash equilibrium is no more than $(1+\frac{1}{n-1})^d$ times the polynomial social cost of the fully mixed Nash equilibrium. This implies the validity of an approximate version of the Fully Mixed Nash Equilibrium Conjecture, in our model. Again, making use of this result, we prove that the Price of Anarchy is bounded from above by $(1+\frac{1}{n-1})^d \cdot B_d$, where B_d is the dth Bell number. We also obtain an upper bound on the Price of Anarchy for the general case of polynomials with non-negative coefficients.
2. The case of pure Nash equilibria: We finally consider the special case of *pure* Nash equilibria. Through elegant combinatorial arguments, we prove a *tight* bound of $\frac{(2^d-1)^d}{(d-1)(2^d-2)^{d-1}}\left(\frac{d-1}{d}\right)^d$ on the Price of Anarchy for the same special case where the polynomial is just the dth power, for any integer $d \geq 2$.

Related Work and Comparison. The KP model has been studied extensively in the recent years; see, e.g., [9,11,15,17,20,22]. Mavronicolas and Spirakis [22] introduced the fully mixed Nash equilibrium. They showed that, in case it exists, the fully mixed Nash equilibrium is unique. Gairing et al. [15] were the first who stated explicitly the Fully Mixed Nash Equilibrium Conjecture. Up to now, the conjecture could be proved only for several particular cases of the KP model [15,21]. A proof of the conjecture will enable the derivation of upper bounds on Price of Anarchy via studying the fully mixed Nash equilibrium.

Lücking et al. [20] considered a hybridization of the KP model with the Wardrop model that adopts *quadratic social cost,* defined as the sum of weighted individual costs. Their model is a special case of our model where the polynomial

is just the second power. In this model, Lücking et al. [20] proved the FMNE Conjecture in case of identical users and identical links.

Gairing et al. [14] introduced yet another model for selfish routing that adopts arbitrary convex latency cost functions and the social cost as the sum of individual costs. This model is identical to our model exactly in the case of identical users and linear cost functions. However, in general the models differ in the definition of individual cost. Hence, the models also differ in the Nash conditions, and thus the sets of mixed Nash equilibria are different. Gairing et al. [14] proved again the FMNE Conjecture in their model.

Tight bounds on the Price of Anarchy for the KP model have been shown to be $\Theta(\frac{\log m}{\log \log m})$ if all links are identical [9,17] and $\Theta(\frac{\log m}{\log \log \log m})$ if links may have varying capacities [9]. The Price of Anarchy has also been studied for a network creation game in [10] and a network design game in [2].

The Wardrop model was already studied in the 1950's [5,31] in the context of road traffic systems. Wardrop [31] introduced the concept of equilibrium to describe user behavior in this kind of traffic networks. For a survey of the early work on this model see [4]. A lot of subsequent work on this model has been motivated by Braess's Paradox [6]. Inspired by the new interest in the Price of Anarchy, Roughgarden and Tardos [27,28,29] re-investigated the Wardrop model. For a survey of results we refer to [12] and references therein.

Road Map. The rest of this paper is organized as follows. Section 2 summarizes some mathematical preliminaries that will be used throughout the paper. Section 3 introduces our formal model and recalls some preliminary facts. In Section 4, we consider the case of identical users and two links. In Section 5, we turn our attention to the more general case of identical users. The case of pure Nash equilibria is treated in Section 6.

2 Mathematical Preliminaries

For all $i \in \mathbb{N}$ we denote $[i] = \{1, \cdots, i\}$.

Falling Factorials, Stirling Numbers and Bell Numbers. For any pair of integers $x \geq 1$ and $i \geq 0$, the ith *falling factorial* $x^{\underline{i}}$ is given by $x^{\underline{i}} = x \cdot (x-1) \cdots (x-i)$. For any pair of integers $d \geq 1$ and $0 \leq i \leq d$, the *Stirling number of the second kind* $S(d,i)$ counts the number of partitions of a set with d elements into exactly i *blocks* (non-empty subsets). In particular, $S(d,0)$ is taken to be 0, while $S(d,1) = S(d,d) = 1$. Also, for all $d \geq 2$, $S(d,2) = 2^{d-1} - 1$. It is known that for all integers $d \geq 1$, it holds that

$$x^d = \sum_{1 \leq i \leq d} S(d,i) \cdot x^{\underline{i-1}}. \tag{1}$$

This implies that the Stirling numbers of the second kind are the *connecting coefficients* between the sequence of powers and the sequence of falling factorials. For any integer $d \geq 1$, the *Bell number* B_d counts the number of partitions of a set with d elements into blocks. Thus, $B_d = \sum_{1 \leq i \leq d} S(d,i)$. For a textbook

introduction to falling factorials, Stirling numbers of the second kind and Bell numbers, see [1, Chapters II & III].

Binomial Cost Functions. We now give the definition and a technical lemma on the *binomial cost function* $H(p,g)$. We will see later that polynomial social cost of a mixed Nash equilibrium can be expressed as sum of binomial cost functions.

Definition 1. *For a set of r probabilities $p = (p_1, \ldots, p_r)$ and a function $g : \mathbb{N} \to \mathbb{R}$ define*

$$H(p,g) = \sum_{A \subseteq [r]} \prod_{k \in A} p_k \prod_{k \notin A} (1 - p_k) \cdot g(|A|).$$

In the same way, we define a function $H(\tilde{p}, r, g)$ by replacing p with a vector of r probabilities all equal to $\tilde{p}$. In case that $g(\delta) = \delta^d$, we write $H_d(p)$ and $H_d(\tilde{p}, r)$, respectively.

Note, that in case of equal probabilities

$$H(\tilde{p}, r, g) = \sum_{1 \leq k \leq r} \binom{r}{k} \tilde{p}^k (1 - \tilde{p})^{r-k} \cdot g(k). \tag{2}$$

Throughout the paper, we will use several times the following technical fact which has been proved in [14].

Lemma 1 (Gairing et al. [14]). *Let g be convex and define $p = (p_1, \ldots, p_r)$ and $\tilde{p} = \frac{\sum_{i \in [r]} p_i}{r}$. Then $H(p,g) \leq H(\tilde{p}, r, g)$.*

The following proposition, proved in the appendix, will be extensively used in the remainder of the paper.

Proposition 1. *For the binomial cost function $H_d(p,r)$ we have*

$$H_d(p,r) = \sum_{1 \leq i \leq d} p^{\underline{i}} \cdot S(d,i) \cdot r^{\underline{i-1}}.$$

3 Model and Preliminaries

General. We consider a *network* consisting of a set of m parallel *links* $1, 2, \ldots, m$ from a *source* node to a *destination* node. Each of n *network users* $1, 2, \ldots, n$, or *users* for short, wishes to route a particular amount of traffic along a (non-fixed) link from source to destination. Denote w_i the *traffic* of user $i \in [n]$. In the model of *identical users,* all user traffics are equal to 1; user traffics may vary arbitrarily in the model of *arbitrary users.* Define the $n \times 1$ *traffic vector* $\mathbf{w}$ in the natural way. Assume throughout that $m \geq 2$ and $n \geq 2$.

A *pure strategy* for user $i \in [n]$ is some specific link. A *mixed strategy* for user $i \in [n]$ is a probability distribution over pure strategies; thus, a mixed strategy is a probability distribution over the set of links. The *support* of the mixed strategy for user $i \in [n]$, denoted $\mathsf{support}(i)$, is the set of those pure strategies (links) to

which i assigns positive probability. The *view* of a link $j \in [m]$, denoted $\mathsf{view}(j)$, is the set of users which assign their traffics to j with non-zero probability. A *pure strategy profile* is represented by an n-tuple $\langle \ell_1, \ell_2, \ldots, \ell_n \rangle \in [m]^n$; a *mixed strategy profile* is represented by an $n \times m$ *probability matrix* $\mathbf{P}$ of nm probabilities p_{ij}, $i \in [n]$ and $j \in [m]$, where p_{ij} is the probability that user i chooses link j.

For a probability matrix $\mathbf{P}$, the support of the mixed strategy for user $i \in [n]$ is the set $\{j \in [m] \mid p_{ij} > 0\}$. A mixed strategy profile $\mathbf{P}$ is *fully mixed* [22, Section 2.2] if for all users $i \in [n]$ and links $j \in [m]$, $p_{ij} > 0$. Throughout, we will cast a pure strategy profile as a special case of a mixed strategy profile.

System and Cost Measures. For a pure strategy profile $\langle \ell_1, \ell_2, \ldots, \ell_n \rangle$, the *latency cost for user* $i \in [n]$ is $\lambda_i = \sum_{k:\ell_k=\ell_i} w_k$; that is, the latency of the link it chooses. For a mixed strategy profile $\mathbf{P}$, denote δ_j the *actual traffic* on link $j \in [m]$; so, δ_j is a random variable. For each link $j \in [m]$, define the *expected latency* Λ_j as the *expected traffic* on link j; thus, $\Lambda_j = \mathcal{E}(\delta_j) = \sum_{i \in [n]} p_{ij} w_i$. For a mixed strategy profile $\mathbf{P}$, the *expected latency cost* for user $i \in [n]$ on link $j \in [m]$, denoted λ_{ij}, is the expectation, over all random choices of the remaining users, of the latency cost for user i had its traffic been assigned to link j; thus, $\lambda_{ij} = w_i + \sum_{k=1, k \neq i} p_{kj} w_k = (1 - p_{ij}) w_i + \Lambda_j$. For each user $i \in [n]$, the *minimum expected latency cost,* denoted λ_i, is the minimum, over all links $j \in [m]$, of the expected latency cost for user i on link j; thus, $\lambda_i = \min_{j \in [m]} \lambda_{ij}$.

Let $\pi^d(\delta) = \sum_{0 \leq t \leq d} a_t \delta^t$ be a polynomial of degree $d \geq 1$ with non-negative coefficients (that is, $a_t \geq 0$ for all $0 \leq t \leq d$). Associated with a traffic vector $\mathbf{w}$, a *polynomial cost function* $\pi^d(\delta)$, and a mixed strategy profile $\mathbf{P}$ is the *polynomial social cost,* or *social cost* for short, denoted $\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w}, \mathbf{P})$, which is the expectation of the sum of the polynomial $\pi^d(\delta)$ evaluated at the incurred link latencies:

$$\begin{aligned} \mathsf{SC}_{\pi^d(\delta)}(\mathbf{w}, \mathbf{P}) &= \mathcal{E}\left(\sum_{j \in [m]} \pi^d\Big(\sum_{k:\ell_k=j} w_k \Big) \right) \\ &= \sum_{j \in [m]} \sum_{A \subseteq [n]} \left(\prod_{i \in A} p_{ij} \right) \left(\prod_{i \notin A} (1 - p_{ij}) \right) \pi^d\Big(\sum_{k:\ell_k=j} w_k \Big). \end{aligned}$$

If we restrict to the polynomial $\pi^d(\delta) = \delta^d$, we write $\mathsf{SC}_{\delta^d}(\mathbf{w}, \mathbf{P})$. Note that

$$\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w}, \mathbf{P}) = \sum_{0 \leq t \leq d} a_t \cdot \mathsf{SC}_{\delta^t}(\mathbf{w}, \mathbf{P}). \tag{3}$$

Moreover, if we restrict to identical users, the formula for social cost reduces to

$$\begin{aligned} \mathsf{SC}_{\pi^d(\delta)}(\mathbf{w}, \mathbf{P}) &= \sum_{j \in [m]} \sum_{A \subseteq [n]} \left(\prod_{i \in A} p_{ij} \right) \left(\prod_{i \notin A} (1 - p_{ij}) \right) \pi^d(|A|) \\ &= \sum_{j \in [m]} H((p_{1j}, \ldots, p_{nj}), \pi^d(\delta_j)). \end{aligned} \tag{4}$$

The *optimum* associated with a traffic vector $\mathbf{w}$, denoted $\mathsf{OPT}_{\pi^d(\delta)}(\mathbf{w})$, is the *least possible* sum of the polynomial $\pi^d(\delta)$ evaluated at the incurred link latencies.

Note that while $\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{P})$ is defined in relation to a mixed strategy profile $\mathbf{P}$, $\mathsf{OPT}_{\pi^d(\delta)}(\mathbf{w})$ refers to the *optimum* pure strategy profile.

Nash Equilibria. We are interested in a special class of mixed strategies called Nash equilibria [24] that we describe below. Formally, the probability matrix $\mathbf{P}$ is a *Nash equilibrium* [18, Section 2] if for all users $i \in [n]$ and links $j \in [m]$, $\lambda_{ij} = \lambda_i$ if $I_{ij} = 1$, and $\lambda_{ij} \geq \lambda_i$ if $I_{ij} = 0$. Thus, each user assigns its traffic with positive probability only on links (possibly more than one of them) for which its expected latency cost is minimized; this implies that there is no incentive for a user to unilaterally deviate from its mixed strategy in order to avoid links on which its expected latency cost is higher than necessary. We will refer to this conditions as *Nash conditions.* Depending on the type of strategy profile, we differ between *pure, mixed* and *fully mixed* Nash equilibria. Recall that in our model, the fully mixed Nash equilibrium $\mathbf{F}$ exists always, and that it is unique (see [22]).

Lemma 2 (Mavronicolas and Spirakis [22]). *Consider the model of arbitrary users. Then, there exists a unique fully mixed Nash equilibrium* $\mathbf{F}$ *with* $f_{ij} = 1/m$, *for any user* $i \in [n]$ *and link* $j \in [m]$.

Price of Anarchy. The *Price of Anarchy,* also called *coordination ratio,* is the maximum value, over all traffic vectors $\mathbf{w}$ and Nash equilibria $\mathbf{P}$ of the ratio $\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{P})/\mathsf{OPT}_{\pi^d(\delta)}(\mathbf{w})$.

4 Identical Users and Two Links

In this section we turn our attention to the case of identical users and two links. We first use Proposition 1 and Lemma 3 to prove the FMNE Conjecture (Theorem 1). Making use of this result, we then give a tight bound on the Price of Anarchy for the special case where the polynomial cost function is just the dth power (Theorem 2). We close with an upper bound on the Price of Anarchy for general polynomial cost functions (Corollary 1).

Lemma 3 (Lücking et al. [20]). *Consider the model of arbitrary users, and let* $j, k \in [m]$. *If* $\mathsf{view}(j) \subsetneq \mathsf{view}(k)$, *then* $\Lambda_j > \Lambda_k$.

Theorem 1. *Consider the model of identical users and two links. Then, for any Nash equilibrium* $\mathbf{P}$, $\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{P}) \leq \mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{F})$.

Proof. Fix any Nash equilibrium $\mathbf{P}$. We can identify three sets of users in $\mathbf{P}$: $\mathcal{U}_1 = \{i : \mathsf{support}(i) = \{1\}\}$, $\mathcal{U}_2 = \{i : \mathsf{support}(i) = \{2\}\}$ and $\mathcal{U}_{12} = \{i : \mathsf{support}(i) = \{1,2\}\}$. Without loss of generality let $|\mathcal{U}_1| \leq |\mathcal{U}_2|$. Denote $u = |\mathcal{U}_1|$, $v = |\mathcal{U}_2| - u$ and $r = |\mathcal{U}_{12}|$.

The proof of the theorem is structured as follows: we first prove that the claim holds if $\mathbf{P}$ is a pure Nash equilibrium. Then we consider the case where $u = 0$. We show that it suffices to consider terms of the same power pairwise. To do so, we proceed by induction on i, $1 \leq i \leq d$, and we show that the claim holds for ith powers. For the case where $u > 0$, we introduce a Nash equilibrium

$\mathbf{Q}$ as a perturbation of $\mathbf{F}$ with a special structure that is similar to the structure of $\mathbf{P}$ such that $\mathsf{SC}_{\pi^{d(\delta)}}(\mathbf{w},\mathbf{Q}) \leq \mathsf{SC}_{\pi^{d(\delta)}}(\mathbf{w},\mathbf{F})$. Due to this special structure, $\mathbf{Q}$ can be compared with $\mathbf{P}$ more easily. To compare the two, we use the fact that the claim holds for $u = 0$. We now continue with the details of the formal proof:

First, let $\mathbf{P}$ be a pure Nash equilibrium, that is, $r = 0$. Due to the Nash conditions, $|\mathcal{U}_1|$ and $|\mathcal{U}_2|$ differ by 1 if n is odd, and $|\mathcal{U}_1| = |\mathcal{U}_2|$ otherwise. Thus, each pure Nash equilibrium $\mathbf{P}$ has unique social cost $\mathsf{SC}_{\pi^{d(\delta)}}(\mathbf{w},\mathbf{P}) = \mathsf{OPT}_{\pi^{d(\delta)}}(\mathbf{w})$, proving the claim for pure Nash equilibria.

Now let $\mathbf{P}$ be an arbitrary (non-pure) Nash equilibrium.

Case 1: $u = 0$: Denote by Λ_1 and Λ_2 the latency on link 1 and 2, respectively. Due to Lemma 3, $\mathsf{view}(1) \subsetneq \mathsf{view}(2)$ implies $\Lambda_1 > \Lambda_2$. First assume $r = 1$. Then, the Nash condition for the mixed user $1 = 1 + v$ implies $v = 0$. This contradicts our assumption that $n \geq 2$. So, assume that $2 \leq r \leq n-1$ (mixed) users are assigned to both links, and that $n - r$ (pure) users are assigned to link 2. Consider any arbitrary mixed user $i \in \mathcal{U}_{12}$. Then, the Nash condition implies

$$\Lambda_1 - p_{i1} + 1 = \Lambda_2 - p_{i2} + 1 = \Lambda_2 + p_{i1} \quad \Rightarrow \quad p_{i1} = \frac{\Lambda_1 - \Lambda_2 + 1}{2}.$$

Since this holds for each mixed user, we write $p(1)$ and $p(2)$ for the probabilities of each mixed user on link 1 and 2, respectively. Since $p(2) = 1 - p(1)$, the Nash condition

$$(r-1)\cdot p(1) + 1 = (r-1)\cdot p(2) + n - r + 1 = (r-1)\cdot(1-p(1)) + n - r + 1$$

for the mixed users implies

$$p(1) = \frac{n-1}{2(r-1)} = \frac{1}{2} + \frac{n-r}{2(r-1)} \qquad \text{and} \qquad p(2) = (1-p(1)) = \frac{1}{2} - \frac{n-r}{2(r-1)}.$$

Thus, we can write

$$\Lambda_1 = r\cdot p(1) = r\cdot\left(\frac{1}{2} + \frac{n-r}{2(r-1)}\right) = \frac{n}{2} + \frac{n-r}{2(r-1)} = \alpha + \beta,$$

where $\alpha = \frac{n}{2}$ and $\beta = \frac{n-r}{2(r-1)}$ with $0 < \beta < \frac{1}{2}$. Moreover, $\Lambda_2 = n - \Lambda_1 = \alpha - \beta$. By Equation (3), we can write $\mathsf{SC}_{\pi^{d(\delta)}}(\mathbf{w},\mathbf{P}) = \sum_{0\leq t\leq d} a_t \cdot \mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{P})$ for any Nash equilibrium $\mathbf{P}$. Since the coefficients a_t are non-negative, it suffices to prove that $\mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{P}) \leq \mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{F})$ for all $1 \leq t \leq d$.

Consider the average probabilities $\tilde{p}(1) = \frac{\Lambda_1}{r} = \frac{\alpha+\beta}{r}$ and $\tilde{p}(2) = \frac{\Lambda_2}{n} = \frac{\alpha-\beta}{n}$ on link 1 and 2, respectively. Since δ^t is convex, we can apply Lemma 1 to derive the upper bound

$$\begin{aligned}
\mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{P}) &\overset{(4)}{=} H_t(\underbrace{(p(1),\ldots,p(1))}_{r}) + H_t(\underbrace{(p(2),\ldots,p(2)}_{r},\underbrace{1,\ldots,1)}_{n-r}) \\
&\overset{Lem.\ 1}{\leq} H_t(\tilde{p}(1),r) + H_t(\tilde{p}(2),n) = H_t(\frac{\alpha+\beta}{r},r) + H_t(\frac{\alpha-\beta}{n},n) \\
&\overset{Prop.\ 1}{=} \sum_{1\leq i\leq t}\left(\frac{\alpha+\beta}{r}\right)^i \cdot S(t,i)\cdot r^{\underline{i-1}} + \sum_{1\leq i\leq t}\left(\frac{\alpha-\beta}{n}\right)^i \cdot S(t,i)\cdot n^{\underline{i-1}} \\
&= \sum_{1\leq i\leq t} S(t,i)\cdot\left[(\alpha+\beta)^i\cdot\frac{r^{\underline{i-1}}}{r^i} + (\alpha-\beta)^i\cdot\frac{n^{\underline{i-1}}}{n^i}\right]
\end{aligned}$$

Moreover, by Lemma 2, we have

$$\mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{F}) \overset{(4)}{=} 2\cdot H_t(\tfrac{1}{2},n) = 2\cdot H_t(\tfrac{\alpha}{n},n) \overset{Prop.\ 1}{=} \sum_{1\le i\le t} S(t,i)\left[2\alpha^i\cdot\frac{n^{\underline{i-1}}}{n^i}\right].$$

In order to prove the claim it suffices to show that

$$\Delta = 2\alpha^i\cdot\frac{n^{\underline{i-1}}}{n^i} - \left[(\alpha+\beta)^i\cdot\frac{r^{\underline{i-1}}}{r^i} + (\alpha-\beta)^i\cdot\frac{n^{\underline{i-1}}}{n^i}\right] \ge 0$$

for all $i \ge 1$. For $i = 1$, we have $\Delta = 2\alpha - [\alpha+\beta+\alpha-\beta] = 0$, proving the basis case. So assume $i \ge 2$. We have

$$\Delta = 2\alpha^i\cdot\frac{n^{\underline{i-1}}}{n^i} - \Bigg[(\alpha+\beta)\cdot\frac{r-(i-1)}{r}\cdot\underbrace{(\alpha+\beta)^{i-1}\cdot\frac{r^{\underline{i-2}}}{r^{i-1}}}_{\ge 0} + (\alpha-\beta)\cdot\frac{n-(i-1)}{n}\cdot\underbrace{(\alpha-\beta)^{i-1}\cdot\frac{n^{\underline{i-2}}}{n^{i-1}}}_{\ge 0}\Bigg].$$

The function

$$f(r) = (\alpha+\beta)\cdot\frac{r-(i-1)}{r} = \left(\frac{(n-1)r}{2(r-1)}\right)\cdot\frac{r-(i-1)}{r} = \frac{n-1}{2}\cdot\frac{r-(i-1)}{r-1}$$

is monotonic increasing in r for all $i \ge 2$, and thus

$$\begin{aligned}\Delta &\ge 2\alpha^i\cdot\frac{n^{\underline{i-1}}}{n^i} - \left[\alpha\cdot\frac{n-(i-1)}{n}\cdot(\alpha+\beta)^{i-1}\cdot\frac{r^{\underline{i-2}}}{r^{i-1}} + \alpha\cdot\frac{n-(i-1)}{n}\cdot(\alpha-\beta)^{i-1}\cdot\frac{n^{\underline{i-2}}}{n^{i-1}}\right]\\ &= 2\alpha^i\cdot\frac{n^{\underline{i-1}}}{n^i} - \alpha\cdot\frac{n-(i-1)}{n}\cdot\left[(\alpha+\beta)^{i-1}\cdot\frac{r^{\underline{i-2}}}{r^{i-1}} + (\alpha-\beta)^{i-1}\cdot\frac{n^{\underline{i-2}}}{n^{i-1}}\right]\\ &\overset{ind.}{\ge} 2\alpha^i\cdot\frac{n^{\underline{i-1}}}{n^i} - \alpha\cdot\frac{n-(i-1)}{n}\cdot 2\alpha^{i-1}\cdot\frac{n^{\underline{i-2}}}{n^{i-1}} = 0,\end{aligned}$$

proving the inductive claim. This finishes the case where $u = 0$.

Case 2: $u > 0$: Consider the following mixed Nash equilibrium $\mathbf{Q}$. On both links, there are u pure users, and the remaining $n-2u$ users have probabilities $q(1) = q(2) = \frac{1}{2}$. Since $\pi^d(\delta)$ is convex and due to Lemma 1, we have $\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{F}) \ge \mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{Q})$. As in case 1, it suffices to show that $\mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{Q}) \ge \mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{P})$ for all $1 \le t \le d$. Denote $\tilde{n} = n-2u$. If $\tilde{n} = 1$, then $\mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{P}) = \mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{Q}) = (u+1)^t + u^t$. Thus, assume $\tilde{n} \ge 2$. We have

$$\mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{Q}) \overset{(4),(2)}{=} 2\cdot\sum_{0\le k\le\tilde{n}}\binom{\tilde{n}}{k}q(1)^k(1-q(1))^{\tilde{n}-k}(k+u)^t = \mathsf{SC}_{(\delta+u)^t}(\tilde{\mathbf{w}},\tilde{\mathbf{Q}}),$$

and by Lemma 1, Equation (4) and Equation (2)

$$\begin{aligned}\mathsf{SC}_{\delta^t}(\mathbf{w},\mathbf{P}) &\le \sum_{0\le k\le r}\binom{r}{k}p(1)^k(1-p(1))^{r-k}(k+u)^t \\ &\quad+\sum_{0\le k\le \tilde{n}}\binom{\tilde{n}}{k}(1-p(1))^k p(1)^{\tilde{n}-k}(k+u)^t \\ &= H(p(1),r,(\delta+u)^t)+H(p(2),\tilde{n},(\delta+u)^t)=\mathsf{SC}_{(\delta+u)^t}(\tilde{\mathbf{w}},\tilde{\mathbf{P}})\end{aligned}$$

where $\tilde{\mathbf{w}}$ is an instance with $\tilde{n}\ge 2$ identical users, $\tilde{\mathbf{Q}}$ is the fully mixed Nash equilibrium of this instance, and $\tilde{\mathbf{P}}$ is a (mixed) Nash equilibrium where with the same structure as in case 1, that is, there are no pure users on link 1. Since $(\delta+u)^t$ is a polynomial of degree t with non-negative coefficients, we can apply case 1, proving the claim. □

Theorem 2. *Consider the model of identical users and two links. Then*

$$\max_{\mathbf{w},\mathbf{P}}\frac{\mathsf{SC}_{\delta^d}(\mathbf{w},\mathbf{P})}{\mathsf{OPT}_{\delta^d}(\mathbf{w})}=\frac{1}{2}\left(2^{d-1}+1\right).$$

Corollary 1. *Consider the model of identical users and two links. Then*

$$\max_{\mathbf{w},\mathbf{P}}\frac{\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{P})}{\mathsf{OPT}_{\pi^d(\delta)}(\mathbf{w})}\le\frac{1}{2}\left(2^d+d-1\right).$$

5 Identical Users

In this section we first prove the validity of an approximate version of the FMNE Conjecture for the model of identical users (Theorem 3). Equipped with this result, we then give a bound on the Price of Anarchy for the special case where the polynomial cost function is just the dth power (Theorem 4), using similar techniques as in [14]. We close with an upper bound on the Price of Anarchy for general polynomial cost functions (Corollary 2).

Theorem 3. *Consider the model of identical users. Then, for any Nash equilibrium* $\mathbf{P}$*,*

$$\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{P})\le\left(1+\frac{1}{n-1}\right)^d\cdot\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{F}).$$

Theorem 4. *Consider the model of identical users. Then*

$$\max_{\mathbf{w},\mathbf{P}}\frac{\mathsf{SC}_{\delta^d}(\mathbf{w},\mathbf{P})}{\mathsf{OPT}_{\delta^d}(\mathbf{w})}\le\left(1+\frac{1}{n-1}\right)^d\cdot B_d.$$

The upper bound on the Price of Anarchy in Theorem 4 consists of two factors. We believe that the factor $(1+\frac{1}{n-1})^d$ is not necessary. Note also, that the second factor B_d exceeds (asymptotically) the tight bound $\frac{1}{2}(2^{d-1}+1)$ on the Price of Anarchy established in Theorem 2 for the case of two links.

Corollary 2. *Consider the model of identical users. Then*

$$\max_{\mathbf{w},\mathbf{P}} \frac{\mathsf{SC}_{\pi^d(\delta)}(\mathbf{w},\mathbf{P})}{\mathsf{OPT}_{\pi^d(\delta)}(\mathbf{w})} \leq \sum_{0\leq t\leq d} \left(1+\frac{1}{n-1}\right)^t \cdot B_t.$$

6 Pure Nash Equilibria

We now prove a tight bound on the Price of Anarchy for pure Nash equilibria for the special case that $\pi^d(\delta) = \delta^d$.

Theorem 5. *Consider the model of arbitrary users, restricted to pure Nash equilibria* **P**, *and let* $d \geq 2$. *Then,*

$$\max_{\mathbf{w},\mathbf{P}} \frac{\mathsf{SC}_{\delta^d}(\mathbf{w},\mathbf{P})}{\mathsf{OPT}_{\delta^d}(\mathbf{w})} = \frac{(2^d-1)^d}{(d-1)(2^d-2)^{d-1}} \left(\frac{d-1}{d}\right)^d.$$

References

1. M. Aigner. *Combinatorial Theory.* Springer-Verlag, 1979.
2. E. Anshelevich, A. Dasgupta, E. Tardos, and T. Wexler. Near-optimal network design with selfish agents. In *Proc. of the 35th Annual ACM Symp. on Theory of Computing,* pp. 511–520, 2003.
3. B. Awerbuch, Y. Azar, Y. Richter, and D. Tsur. Tradeoffs in worst-case equilibria. In *Proc. of the 1st Int. Workshop on Approximation and Online Algorithms,* LNCS 2909, pp. 41–52, 2003.
4. M.J. Beckmann. On the theory of traffic flow in networks. *Traffic Quart,* 21:109–116, 1967.
5. M. Beckmann, C.B. McGuire and, C.B. Winsten. *Studies in the Economics of Transportation.* Yale University Press, 1956.
6. D. Braess. Über ein Paradoxon der Verkehrsplanung. *Unternehmensforschung,* 12:258–268, 1968.
7. A. Czumaj. Selfish routing on the internet. *To appear as a chapter in Handbook of Scheduling: Algorithms, Models, and Performance Analysis, edited by J. Leung, CRC Press,* 2004.
8. A. Czumaj, P. Krysta, and B. Vöcking. Selfish traffic allocation for server farms. In *Proc. of the 34th Annual ACM Symp. on Theory of Computing,* pp. 287–296, 2002.
9. A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms,* pp. 413–420, 2002.
10. A. Fabrikant, A. Luthra, E. Maneva, C.H. Papadimitriou, and S. Shenker. On a network creation game. In *Proc. of the 22nd Annual Symp. on Principles of Distributed Computing,* pp. 347–351, 2003.
11. R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *Proc. of the 30th Int. Colloq. on Automata, Languages, and Programming,* LNCS 2719, pp. 514–526, 2003.
12. R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Selfish routing in non-cooperative networks: A survey. In *Proc. of the 28th Int. Symp. on Mathematical Foundations of Computer Science,* LNCS 2747, pp. 21–45, 2003.

13. D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Proc. of the 29th Int. Colloq. on Automata, Languages, and Programming,* LNCS 2380, pp. 123–134, 2002.
14. M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. Nash equilibria in discrete routing games with convex latency functions. In *Proc. of the 31st Int. Colloq. on Automata, Languages, and Programming,* 2004, to appear.
15. M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and P. Spirakis. Extreme nash equilibria. In *Proc. of the 8th Italian Conf. on Theoretical Computer Science,* LNCS 2841, pp. 1–20, 2003. Accepted to *Theoretical Computer Science.*
16. E. Koutsoupias. Selfish task allocation. *Bulletin of the EATCS,* 81:79–88, 2003.
17. E. Koutsoupias, M. Mavronicolas, and P. Spirakis. Approximate equilibria and ball fusion. *Theory of Computing Systems,* 36(6):683–693, 2003.
18. E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proc. of the 16th Int. Symp. on Theoretical Aspects of Computer Science,* LNCS 1563, pp. 404–413, 1999.
19. V.S.A. Kumar and M.V. Marathe. Improved results for stackelberg scheduling strategies. In *Proc. of the 29th Int. Colloq. on Automata, Languages, and Programming,* LNCS 2380, pp. 776–787, 2002.
20. T. Lücking, M. Mavronicolas, B. Monien, and M. Rode. A new model for selfish routing. In *Proc. of the 21st Int. Symp. on Theoretical Aspects of Computer Science,* LNCS 2996, pp. 547–558, 2004.
21. T. Lücking, M. Mavronicolas, B. Monien, M. Rode, P. Spirakis, and I. Vrto. Which is the Worst-case Nash Equilibrium? In *Proc. of the 28th Int. Symp. on Mathematical Foundations of Computer Science,* LNCS 2747, pp. 551–561, 2003.
22. M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proc. of the 33rd Annual ACM Symp. on Theory of Computing,* pp. 510–519, 2001.
23. I. Milchtaich. Congestion games with player-specific payoff functions. *Games and economic behavior,* 13:111–124, 1996.
24. J. Nash. Non-cooperative games. *Annals of Mathematics,* 54(2):286–295, 1951.
25. R.W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *Int. Journal of Game Theory,* 2:65–67, 1973.
26. T. Roughgarden. Designing networks for selfish users is hard. In *Proc. of the 42nd Annual Symp. on Foundations of Computer Science,* pp. 472–481, 2001.
27. T. Roughgarden. Stackelberg scheduling strategies. In *Proc. of the 33rd Annual ACM Symp. on the Theory of Computing,* pp. 104–113, 2001.
28. T. Roughgarden. The price of anarchy is independent of the network topology. In *Proc. of the 34th Annual ACM Symp. on the Theory of Computing,* pp. 428–437, 2002.
29. T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of the ACM,* 49(2):236–259, 2002.
30. V. Vazirani. *Approximation Algorithms.* Springer-Verlag, 2001.
31. J.G. Wardrop. Some theoretical aspects of road traffic research. In *Proc. of the Institute of Civil Engineers, Pt. II, Vol. 1,* pp. 325–378, 1956.

Agent-Based Information Handling in Large Networks*

Robert Elsässer, Ulf Lorenz, and Thomas Sauerwald

Department of Mathematics and Computer Science, University of Paderborn
Fürstenallee 11, D-33102 Paderborn
{elsa,flulo,sauerwal}@uni-paderborn.de

Abstract. Mobile agents are software abstractions that can migrate across the links of a network. They naturally extend the object oriented program style and nicely correspond to agents as examined in game theory. In this paper, we introduce a simple, robust, and efficient randomized broadcast protocol within this mobile agent programming paradigm. We show that by using this scheme, broadcasting requires in a random graph of certain density $O(\ln n)$ steps, where n denotes the number of nodes in the graph. Then, we consider bounded degree graphs and prove that we are able to distribute an information among all nodes in $O(D)$ steps, where D denotes the diameter of the graph. We also show that, in contrast to traditional randomized broadcasting, graphs exist in which spreading an information requires $\Omega(n^2)$ steps. On the other hand, some graphs which require $\Omega(n \ln n)$ steps to spread the information in the traditional broadcast model, allow very fast agent based broadcasting. It should be noted that the previously mentioned results are guaranteed with probability $1 - o(1/n)$.

1 Introduction

In the agent-based model, as examined in this paper, n agents are randomly distributed in a graph with n nodes. To each node belongs one agent, which possesses certain information of its node. The agents move randomly from one node to another, in rounds, across the edges of the network. At a given time, we inject a piece of information to a node v, which is then called *informed* or *infected.* In the succeeding rounds, informed nodes infect visiting agents and infected agents carry the information to other nodes. The goal is to determine the number of rounds required by the agents to infect all nodes in the network. The agents only know the neighborhood of their hosting nodes; the size of the network, its topology or the 'infected' area are unknown.

The randomized broadcast strategy, as introduced in this paper, is simple, scalable, and robust (i.e., some nodes can be switched off, while their corresponding agents are working in the system). Apart from these characteristics,

* This work was partly supported by the IST Program of the EU under contract numbers IST-2001-33116 (FLAGS) and IST-1999-20751 (UpTV).

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 586–598, 2004.
© Springer-Verlag Berlin Heidelberg 2004

the algorithm also allows to interleave several tasks. Due to this property, agent based randomized broadcasting is well suited for asynchronous systems, in which we cannot expect to control the task flow of all participants. The object oriented programming view greatly interferes with this paradigm [12].

This agent based model is fairly different from the traditional randomized broadcast model (TRB) [8], in which any informed node sends, in each step, some rumor to a randomly chosen neighbor. We will see in Section 4 that there are examples for the agent based algorithm being asymptotically slower than the corresponding algorithm in the TRB, and vice versa. Nevertheless, we show that for several important graph classes broadcast can be performed asymptotically as fast as in the TRB model.

1.1 Specific Background

Personalized television is one of the most interesting trends in the development of a new generation of video recorders. The idea is that each 'Personal Video Recorder' (PVR) collects information about its user, and then filters the sky for contributions which the PVR supposes to be interesting for it. The collected information is called a user profile. One possibility for a PVR to come to reasonable predictions is to compare its profile with the profiles of other users, and then to base predictions on the opinions of users which have a similar profile. Thereby, two questions arise. First, how to define similarity, but reasonable heuristics exist. Second, how can find a PVR the users which are similar to it, i.e., how to find *friends.* The common procedure is to build a database of profiles and compare them centralized [17]. There have also been efforts to work within a distributed database, however, the main focus remains centralistic and it is questionable whether the existing proprietary systems will work in the mass market with more than 10^8 PVRs.

We propose another way to deal with such a large number of PVRs. Assume that every PVR that enters the system, notifies a central instance which places it into a ground network (such as a grid), ensuring that all PVRs are within one connected component. From a practical point of view, this causes no timing problems, since every PVR enters the system only once. Each PVR has the ability to build a specified number of connections to other PVRs, which means that it possesses a couple of half-links at the beginning. A pair of half links becomes a virtual link across the Internet, as soon as the corresponding PVRs have found each other. In order to achieve this, each PVR is allowed to send out one agent which executes the following protocol: When an agent A enters a node P, then A asks P whether it wants to be a friend of $A's$ origin Q. If P has either free half-links, or the distance between the profile of Q and P is smaller than the distance to one of its neighbors, then P agrees and sends a request directly to Q. If Q wants to be $P's$ friend too, then they establish the link between Q and P. Possibly, P or Q have to resolve an existing link to a processor R. In that case R is notified.

Lots of links are resolved and connected in the system, however, after a finite time the system stabilizes. The reason simply is that there is one cheapest edge

in the system. As soon as this edge is detected, it will never be resolved again. Altogether, the procedure leads to a distributed greedy algorithm, and from the edges point of view, the system finishes in a Nash equilibrium. When the system is frozen, the nodes are not interested in changing their connections without further connections being changed simultaneously. The number of steps until the system is frozen depends on the profiles and on the networks which arise during the procedure. Simulations showed that everything stabilizes fast with randomly chosen profiles, however one subproblem remains: When the algorithm comes close to its final state, a few free half-links occur which should be reconnected as fast as possible. This subproblem can be reduced to the problem of fast broadcasting in the agent based model.

The system can be seen as a network of n processors, which communicate via a fixed number of agents, ensuring a fast distribution of information, without flooding the network, and without restricting the reach of a message. Since the number of PVRs which enter or possibly leave the system changes slowly, we assume that the number of nodes in our network is fixed. The agent based view on the given problem fits nicely into known distributed agent protocols, e.g. Distance Vector Routing [13].

1.2 Related Work

Mobile agents (MAs) [16] are software abstractions that can migrate across a network, as for example, the Internet. They are the successors of process migration, and therefore in its core as old as distributed computing itself. Several high level programming tools, such as Java or the script language Tcl/tk, support mobility. Aglets, Concordia, Jumping Beans etc. are examples of industrial programming tools which support the Mobile Agent paradigm. From the software technology's point of view, the MA paradigm serves with benefits like improved locality of reference, the ability to deal with ad hoc ideas as disconnected users, and flexibility. It is worth mentioning that the agent is the basic entity in game theory mostly analyzed within the context of selfish agents [20]. Nevertheless, the MA paradigm stands in concurrency with other techniques like Message Passing. Certainly, many problems addressed by mobility can also be solved even more efficiently and more securely by static clients that exchange messages. We see the difference between MA and Message Passing only in the programmer's point of view. Indeed a mobile agent is a piece of software, and an MA is a data packet which invokes some action on the target computer, from a machine's point of view, MA is very similar to Message Passing. In the same way as a message, the MA must be processed with the help of its recipient. An MA however, is a quite special message, as not every traditional message encodes a piece of a program.

There is also a long history of empirical and theoretical research on *epidemic disease* within cliques and random networks [11,18,1]. Concerning results on the field of *broadcasting,* Frieze and Molloy [9] showed that in a random graph G_p with n vertices, an upper bound of $\Theta(\frac{\ln n}{n})$ is required on the edge density in order to deterministically broadcast information in $\lceil \log_2 n \rceil$ steps (with high

probability). This result has been improved by Chen in [3]. The traditional randomized broadcasting also been examined within geometric networks in [15]. It is shown that new information is spread to nodes at distance t with high probability in $O(\ln^{1+\epsilon} t)$ steps. A similar broadcasting model has been analyzed under the name of rumor-spreading. There, one of n people knows some rumor and any 'knower', in our language an infected person, infects in each round another randomly chosen person of the population. The goal is to determine the number of rounds required for infecting all persons in the system. Pittel [19] proved a nice result, which shows that within $\log_2(n) + \ln(n) + O(1)$ steps they are probably infected. Feige et. al. [8] extended the results to different graph classes. Karp et. al. [14] showed that, in the so-called random phone call model, the number of messages can be bounded by $O(n \ln \ln n)$.

1.3 Our Results

We start our analysis by considering randomized agent based broadcasting in random graphs. We show in Section 2 that within $O(\ln n)$ rounds, every node of a random graph receives the information (with high probability) whenever the generating probability function of the random graph exceeds some certain threshold. In Section 3, we prove that in any bounded degree graph, we can broadcast within $O(D)$ rounds with high probability, where D denotes the diameter of the graph. In Section 4, we consider graph classes on which agent based broadcasting performs very fast or very slow. We show that each graph allows agent based broadcasting in $O(n^2)$ rounds, however, graphs exist on which $\Omega(n^2)$ steps are always required. In contrast to this, on the star we need only $O(\ln n)$ rounds to spread the information (with high probability), while in the TRB model at least $\Omega(n \ln n)$ steps are needed.

2 Information Spreading in Random Graphs

In this section, we consider the problem of agent based information spreading in a randomly connected environment. We assume that the underlying network is modelled by a random graph $G_p = (V, E)$ defined as follows: Given n and p, generate graph G_p with n vertices by letting each pair be an edge with probability p, independently [2]. Here, we assume that $p = \delta \frac{\ln n}{n}$, where $\delta = 2 + \Omega(1)$. This implies that the graph is connected with high probability [1] (e.g. [2]). This choice of p also implies that the constants $\alpha > 0$ and $\beta > \alpha$ exist so that, with high probability, $\alpha pn \leq d_{\min} \leq d_{\max} \leq \beta pn$, where $d_{\min}$ and $d_{\max}$ represent the minimal and maximal vertex degrees in G_p, respectively. The function $t(n) = \ln n/n$ is the so called threshold probability function for the connectivity of G_p, i.e., any p with $p/t(n) \to 0$ implies that almost no G_p is connected and any p with $p/t(n) \to \infty$ implies that almost every G_p is connected. It is worth mentioning that if $\delta < 2$, then the graph is disconnected with probability at least $1/n$.

[1] "With high probability" or "w.h.p." means with probability $1 - o(1/n)$.

Therefore, the number of steps required to spread the information in this case is ∞ with probability $\Omega(1/n)$.

For any $t \in \mathbb{N}$, the agents lying at time t on a node $v \in V$ are able to change in round $t+1$ from v to any of its neighbors with probability $1/(d_v+1)$, where d_v denotes the degree of v. Accordingly, an agent remains on its hosting node with the same probability $1/(d_v+1)$. At some time, a piece of information is placed at one of the nodes, and the goal is to determine the number of rounds needed to distribute this information by n agents in the system. Note that at the beginning, each agent is equally distributed among the nodes of the system.

Let $I(t)$ denote the number of infected nodes at time t. Similarly, $I_a(t)$ denotes the number of infected agents at time t. Let $i(t) = I(t)/n$ and $i_a(t) = I_a(t)/n$ be the fraction of infected nodes and infected agents, respectively, at time t. The healthy nodes and agents are denoted by $H(t) = n - I(t)$ and $H_a(t) = n - I_a(t)$.

We will show that if $p = \delta \frac{\ln n}{n}$ with $\delta = 2 + \Omega(1)$, then every node of G_p is infected after $O(\log n)$ steps, with high probability. First we analyze the distribution of the agents among the nodes of the graph. If a graph is regular, then each agent jumps to some neighbor of the hosting node with the same probability. We can describe this process by a Markov chain with transition matrix P, where $P_{i,j} = 1/(d+1)$ if $\{i,j\} \in E$ or $i = j$ (d denotes the degree of the nodes in the graph), and $P_{i,j} = 0$ otherwise. Since P is double stochastic, the vector $1 = (1,1,\ldots,1)$ is an eigenvector of P with eigenvalue 1 and all other eigenvalues are in the range $(-1,1)$. If we assume that at the beginning each agent is distributed with the same probability among the nodes, then in any step, an arbitrary agent lies on some node of the graph with probability $1/n$. However, if a graph is not regular, then this does not hold. The transition matrix P is then defined by $P_{i,j} = 1/(d_j+1)$, if $\{i,j\} \in E$ or $i = j$[2], where d_j describes the degree of node j, and $P_{i,j} = 0$ otherwise. If the vector $(x_1,\ldots,x_n)$ denotes the stationary distribution of the ergodic reversible Markov chain described by P, then it holds that $x_i = (d_i+1)/(\sum_{j=1}^{n} d_j + n)$ for any $i \in \{1,\ldots,n\}$. In the case of a random graph G_p, if $p = (2+\Omega(1)) \ln n/n$, then $x_{\max}/x_{\min} \leq (1+\beta)/(1+\alpha)$, where α and β are the constants defined in the first paragraph of this section.

Concerning the expansion properties of $G_p = (V,E)$, we can use the results of [2,5] and state the following theorem.

Theorem 1. *Let $G_p = (V,E)$ be a random graph with n nodes, generated by $p = (2+\Omega(1)) \ln n/n$. Then, it holds with high probability that $\min\{|E(X,\overline{X})|/m \mid X \subset V \text{ and } |X| = m\} \geq \gamma pn$ for any $m \leq n/2$, where $E(X,\overline{X})$ represents the set of edges between X and $\overline{X}$, and $\gamma > 0$ is a constant value.*

This theorem implies that, with high probability, the distribution of the agents among the nodes of G_p becomes very close to the stationary distribution within $O(\ln n)$ steps [5,21,7]. As described in the previous paragraph, the stationary

[2] Letting $P_{i,i}$ be nonzero is a simple device to avoid periodicity problems.

distribution differs for some node of G_p only by a constant factor from the uniform distribution.

We assume in the sequel that in each step every agent is distributed with the same probability $1/n$ among the nodes of the graph. The results we prove in this section also hold without this assumption, however, the proofs are more complicated, and due to simplicity reasons we show the main theorem under the assumption mentioned before. First we analyze the case when $I(t) \leq q \ln \ln n$, where q is a properly chosen constant value. We should mention that some of the proofs are omitted here due to space limitations. They will apear in the journal version of the paper.

Remark 1. If $I(t) \leq q \ln \ln n$ at some time t and $I_a(t) \geq 1$, then in step $t+1$ another node will be infected with probability of at least $1 - O(\ln \ln n / \ln n)$, where q is a properly chosen large constant.

Now consider the case when $q \ln \ln n \leq I(t) \leq q \ln n$.

Lemma 1. *Let S be the set of infected nodes in G_p at time t, and assume that each node $i \in S$ has at most $\max\{c_\delta \ln |S|, c_\delta p|S|\}$ neighbors in S, where c_δ is a constant value. If $I(t) \in [q \ln \ln n, q \ln n]$, where $I(t) = |S|$, then a constant c exists so that $I(t+1) \geq I(t)(1 + 1/c)$ with probability $1 - o(1/\ln n)$.*

After $I(t) \geq q \ln n$ is achieved, similar methods to that of [8] can be used to show fast broadcasting. However, in this agent based model we can only guarantee that at least half of the infected nodes are able to forward the information. We have to show that a constant fraction of the agents positioned on these nodes infect some healthy nodes in the next step. We consider now the case when $q \ln n \leq I(t) \leq \sqrt[4]{n}$.

Lemma 2. *Let S be the subset of infected nodes at some time t with $I(t) = |S|$. If $I(t) \in [q \ln n, \sqrt[4]{n}]$, then it holds that $I(t+1) \geq I(t)(1 + 1/c)$ with probability $1 - o(1/n)$, where c is a proper constant value.*

Proof. Let a node be called empty if no agent is lying on it. Assuming that the agents are equally distributed among the nodes of the graph, a node is empty with probability $p_1 \approx 1/e$. It is worth mentioning that another node is empty with the conditional probability p_2 (given that a node exists, which is already empty), where $p_2 < p_1$. Nevertheless, we can show by using the Chernoff bound on the tail of a binomial distribution [4,10] that the probability for having more than $I(t)/2$ empty nodes is $o(1/n^2)$, whenever q is large enough. Due to Theorem 1, we know that at least $\gamma |S| pn$ edges are connecting S and $V \setminus S$. Since all these edges occur with the same probability between two arbitrary nodes $u \in S$ and $v \in V \setminus S$, there are at least $I(t)/4$ nonempty infected nodes having each more than $\gamma pn/2$ edges to noninfected nodes. Then, we can show using the Chernoff bounds as before that a constant c'' exists so that the agents from at least $I(t)/(4c'')$ infected nonempty nodes jump to healthy nodes. The probability that j infected agents share the same node after the $(t+1)$st step is smaller than $\binom{I_a(t)}{j} (1/n)^j = O(1/n^{3j/4})$, where $j \in \mathbb{N}$. Thus, with probability

$1 - o(1/n^2)$, each newly infected node has at most 2 previously infected agents lying on it. Since $I(t)/(4c'')$ infected agents are jumping to healthy nodes in step $t+1$ (with probability $1 - o(1/n^2)$), and each newly infected node has at most 2 of such agents, it follows that at least $I(t)/(8c'')$ nodes become infected in step $t+1$ (with probability $1 - o(1/n^2)$). □

Lemma 3. *Let S be the subset of infected nodes at some time t with $I(t) = |S|$. If $\sqrt[4]{n} \leq I(t) \leq n - \sqrt[4]{n}$, then it holds that $I(t+1) \geq I(t)(1 + H(t)/(cn))$, for some constant c.*

If $H(t) \leq \sqrt[4]{n}$, then we can show the following lemma.

Lemma 4. *Let S be a set of healthy nodes in G_p. If $H(t) \leq \sqrt[4]{n}$ at some time t, where $H(t) = |S|$, then a constant $c > 0$ exists so that an arbitrary node $v \in S$ is infected in step $t+1$ with probability $1/c$. Moreover, after $O(\ln n)$ rounds, all nodes of the graph are infected with high probability.*

We are now ready to compute the number of steps needed for the distribution of an infection in the system.

Theorem 2. *Let $G_p = (V, E)$ be a random graph with n vertices, where $p = (2 + \Omega(1)) \ln n/n$. At the beginning , we equally distribute n agents among the nodes of G_p, and allow each agent lying on some node $v \in V$ to change with probability $1/(d_v + 1)$ to one of the neighbors of v. Assuming that at some time an arbitrary node receives an information, after $O(\log n)$ rounds the information is completely distributed in the whole system with high probability.*

Proof. After a piece of information is injected to some node $v \in V$, then within $O(\ln n)$ rounds one of the agents will visit v, with high probability, and carry the information further. Now, we consider the following algorithm. Until $I(t) \leq q \ln \ln n$, we allow only this one agent to transmit the infection from one node to another. Due to Remark 1, within $O(\ln n)$ steps we achieve $I(t) \geq q \ln \ln n$ (w.h.p.). In the following rounds, we call a step successful, if within this step at least $I(t)H(t)/(cn)$ uninfected nodes become infected, where c is the constant defined in Lemma 1, 2 and 3. Otherwise, the step is called unsuccessful and the agents that have jumped to healthy nodes become healthy (instead of infecting the nodes). Since the assumptions of Lemma 1, 2, and 3 are fulfilled, we need $O(\ln n)$ steps to achieve $H(t) \leq \sqrt[4]{n}$ (w.h.p.). Finally, if $H(t) \leq \sqrt[4]{n}$, we do not speak anymore about successful or unsuccessful rounds and allow each infected agent to infect any healthy node without becoming itself healthy at all. Using Lemma 4, we can show that with high probability, within another $O(\ln n)$ steps all nodes in the graph become infected.

Due to the arguments described above, the algorithm described in this proof spreads the information among all nodes in the graph within $O(\ln n)$ steps. Obviously, the algorithm presented at the beginning of this section performs faster than the one described here, and therefore the theorem holds. □

The results of Theorem 2 can easily be generalized to the other traditional random graph model [2]: Given n and m, let each graph with n vertices and m edges occur with probability $\binom{N}{m}^{-1}$, where $N = \binom{n}{2}$. The random variable $G_{n,m}$ represents a graph generated in this way. If $m = \delta n \ln n$ with $\delta = 1 + \Omega(1)$, then the results described in this section also hold for $G_{n,m}$.

3 Information Spreading in Bounded Degree Graphs

In the previous section, we determined the number of steps required to distribute the information in a system described by a random graph. In this section, we consider the previously mentioned distribution problem in a more restricted environment. We also slightly modify the randomized scheme in the following way. An agent lying at time t on some node $v \in V$ can jump to one of the neighbors of v with probability $1/(2d_v)$, where d_v denotes the degree of node v. Accordingly, the agent remains on v with probability 1/2. We assume throughout this section that d_v is bounded by some constant value c for any node v.

We show that with high probability, in at most $O(D)$ steps the information is distributed among the nodes of the system, where D denotes the diameter of G. First we analyze the probability distribution of the agents among the nodes of the graph. As described in the previous section, each agent is equally distributed at the beginning. Then, the following lemma can be stated.

Lemma 5. *Let $G = (V, E)$ be a graph and let P be a transition matrix defined in the following way: $P_{i,i} = 1/2$ for any $i \in \{1, \ldots, n\}$, $P_{i,j} = 1/(2d_j)$ if $\{i, j\} \in E$, where d_j describes the degree of node j, and $P_{i,j} = 0$ otherwise. We assume that a constant c exists so that $c \geq d_i$ for any $i \in \{1, \ldots, n\}$. Let $x = (x_1, \ldots, x_n)$ be defined by $x^T = P^t \cdot \frac{1}{n}\mathbf{1}^T$, where t is an arbitrary chosen integer. Then, it holds that $x_i \geq 1/(cn)$ for any $i \in \{1, \ldots, n\}$.*

Proof. The matrix P represents the transition matrix of an ergodic reversible Markov chain with stationary distribution π, where $\pi_i = d_i / \sum_{j=1}^{n} d_j$. Vector π is an eigenvector of P with eigenvalue 1. Due to the properties of P, its eigenvalues are real and they lie in the range [0,1]. On the other hand, the matrix P can be viewed as a diffusion matrix of a node-weighted graph with the structure of G and node-weights d_i [7]. Then, $P = I - \frac{1}{2}L\mathcal{D}^{-1}$, where I is the identity matrix, L is the Laplacian of G [6], and $\mathcal{D}$ is the diagonal matrix containing d_i in the ith diagonal entry. The initial load distribution has the form $w^0 = \frac{1}{n}\mathbf{1}$, and it holds that $w^{k+1} = Pw^k$ in every iteration step k, where w^k denotes the load distribution in G after k steps. Then, it also holds that

$$w_i^{k+1} = w_i^k - \sum_{j \in N(i)} \frac{1}{2}\left(\frac{w_i^k}{d_i} - \frac{w_j^k}{d_j}\right),$$

where $N(i)$ defines the set of neighbors of node i. We show that $\min_{i=1}^{n} \frac{w_i^k}{d_i} \leq \min_{i=1}^{n} \frac{w_i^{k+1}}{d_i}$ for any k. Assume first that $\frac{w_l^k}{d_l} = \min_{i=1}^{n} \frac{w_i^k}{d_i}$ for some $l \in \{1, \ldots, n\}$. For any node $i \in V$ it holds that

$$w_i^{k+1} = w_i^k - \sum_{j \in N(i)} \frac{1}{2} \left(\frac{w_i^k}{d_i} - \frac{w_j^k}{d_j} \right) = \frac{w_i^k}{2} + \frac{1}{2} \sum_{j \in N(i)} \frac{w_j^k}{d_j} \geq \frac{w_i^k}{2} + \frac{d_i}{2} \cdot \frac{w_l^k}{d_l}.$$

Thus,

$$\frac{w_i^{k+1}}{d_i} \geq \frac{1}{2} \left(\frac{w_i^k}{d_i} + \frac{w_l^k}{d_l} \right) \geq \frac{w_l^k}{d_l}.$$

Since $\min_{i=1}^{n} \frac{w_i^0}{d_i} \geq \frac{1}{cn}$, the lemma holds. □

We are now ready to determine the number of steps needed to distribute the information in a graph with bounded vertex degree.

Theorem 3. *Let $G = (V, E)$ be a graph, where $d_i \leq c$, c being constant, for any $i \in V$. We assume that at the beginning, n agents are equally distributed among the nodes of G, and at some time an information is placed on a node of G. If the movements of the agents satisfy the transition probabilities described by the transition matrix P (as defined in Lemma 5), then within $O(D)$ steps every node will get the information by some agent with probability $1 - o(1/n)$.*

Proof. Due to Lemma 5, any agent lies in each step on some arbitrary node i with a probability of at least $1/(cn)$. Now we can use the techniques described in [8] to show the theorem. It is obvious that the diameter of a graph with bounded vertex degree is $\Omega(\log n)$. Given a shortest path between two vertices in G, the information fails to traverse this path in $6c^2(D + 3 \ln n)$ steps with probability $o(1/n^2)$. Hence, the information reaches all vertices of the graph within $O(D)$ steps with probability $1 - o(1/n)$. □

4 Best-Case and Worst-Case Graphs

In this section, we consider graph classes on which we can broadcast in the agent based model very fast or very slowly (with high probability). The agents lying on the nodes of some graph $G = (V, E)$ are able to change from a node $v \in V$ to any of its neighbors with probability $1/(d_v + 1)$, where d_v denotes the degree of i. Accordingly, an agent remains on its hosting node with the same probability $1/(d_v + 1)$. Here, we assume that the agents are lying on the nodes according to the stationary distribution of the Markov chain determined by the transition matrix P as defined in Section 2.

First, we determine the upper bound on the time needed to spread the information in an arbitrary graph G.

Theorem 4. *In a graph $G = (V,E)$, we need at most $O(n^2)$ rounds to spread an information among the nodes of G. Moreover, graphs $G = (V, E)$ exist in which $\Omega(n^2)$ rounds are necessary for broadcasting in the agent based model.*

Proof. We assume that the agents are distributed among the nodes according to the stationary distribution, i.e., an agent is lying on node $i \in V$ with probability $(d_i + 1)/(\sum_{j=1}^{n} d_j + n) \geq (d_i + 1)/n^2$. Let $(u_1, u_2, \ldots, u_k)$ be a shortest path connecting u_1 and u_k. The expected number of iterations between the time u_i receives the information and until the time u_i transmits the information to u_{i+1}, is bounded by $(d_i + 1)en/((e-1)(d_i + 1))$. The expected number of steps until u_k receives the information is therefore less than $enk/(e-1)$. Using the Chernoff bound [4,10], we can show that u_k receives the information within $O(n(k+\ln n))$ steps with probability $1 - o(1/n)$. Since $k = O(n)$, the first statement of the theorem follows.

In order to show the second statement, we consider the graph $G = (V, E)$ with n vertices constructed as follows: The first $n/2$ vertices, labelled $1, \ldots, n/2$, are connected with each other mutually, forming a complete graph with $n/2$ vertices. Vertex $n/2$ is additionally connected to vertex $n/2 + 1$. Then, we connect the vertices $n/2+j$ and $n/2+j+1$ for any $j \in \{1, \ldots, n/2-1\}$, by letting the last $n/2$ nodes form a path of length $n/2$. An agent is lying on node $i \in \{1, \ldots, n/2 - 1\}$ with probability $n/(n^2/2 + n + 3)$. Node $n/2$ has an agent with probability $(n+2)/(n^2/2+n+3)$. All other nodes excepting n have an agent with probability $3/(n^2/2 + n + 3)$, and finally on n a certain agent is lying with probability $2/(n^2/2 + n + 3)$.

With some constant probability, each agent lies on some node $i \in \{1, \ldots, n/2\}$. Letting the information be placed somewhere, node n receives the information only if one of the agents traverses the whole path from node $n/2+1$ to node n. Using simple probability theory, an agent traverses the path (without dropping first into the complete part) with probability $1/n$. Noting that an agent jumps from node $n/2$ to node $n/2 + 1$ with probability $O(1/n)$, we obtain the second statement of the theorem. □

Now we study a graph on which an exponential gap between the speed of information spreading in the traditional broadcast model and in the agent based model occurs. Let $G_u = (V, E)$ be a graph, which consists of $2k$ levels, where each level $i \in \{0, \ldots, 2k - 1\}$ contains $2^{(i \bmod k)}$ vertices. We connect the vertices of two consecutive levels i and $i + 1$ mutually, obtaining for any $(i \bmod k) \in \{0, \ldots, k-2\}$ a complete bipartite graph between levels i and $i+1$. Additionally, we connect the nodes between levels $k-1$ and $2k-1$ in the same way. Using the techniques described in the proof of Theorem 4, it can be shown that we need $\Omega(k2^k)$ steps to propagate the information through the network. In the traditional randomized broadcast model, only $O(k)$ steps are required. Thus, there is an exponential gap between the speed of broadcasting in these two models.

In contrast to the previous paragraph, the star is a good-natured graph for broadcasting in our agent based model. Despite the fact that the star has only $\Theta(n)$ edges, whereas the complete graph has $\Theta(n^2)$, the time of a broadcast is

$O(\ln n)$ with high probability for both. Surprisingly, the infection among the agents is even faster than on the complete graph. The reason for this is the outstanding position of the central node, which we call z, and we denote the number of agents visiting z at time t by $z(t)$.

Theorem 5. *Let $\mathcal{S}_n = (V, E)$ be the star with n vertices and let z be its central node. In $\mathcal{S}_n$ we equally distribute n agents among the nodes and allow each agent lying on some node $v \in V \setminus \{z\}$ to change with probability 1/2 to z. Any agent lying on z is allowed to change with probability $p_v = 1/n$ to some noncentral node v. Assuming that at some time an arbitrary node receives an information, after $O(\ln n)$ rounds the information is completely distributed in the whole system with probability $1 - o(1/n)$.*

Proof. First, we show that the node z is infected in constant expected time. If $i_a(t) > 0$, then we may suppose that the infected agent is on a noncentral node $v \in V \setminus \{z\}$. Since the agent jumps to z with probability 1/2, the agent infects z after 2 expected rounds. If $i_a(t) = 0$ and $i(t) > 0$, then let $v \in V \setminus \{z\}$ be an infected noncentral node. After an agent reaches node v, we are able to apply the previous case. The probability that a certain uninfected agent A reaches node v within two rounds is at least $1/(2n)$. To see this, first consider the case when A is on another noncentral node v'. The agent leaves v' with probability 1/2. However, if A is on z, it jumps to v with probability $1/n$. Thus, the probability that at least one agent reaches v within two rounds is: $1-(1-1/(2n))^n \geq 1-e^{-\frac{1}{2}}$. Therefore, the expected time until an agent jumps to v is $2/(1-e^{-\frac{1}{2}}) \leq 2 \cdot 3 = 6$. Using the previous results, node z is infected within $O(\ln n)$ steps with probability $1 - o(1/n)$.

Now we will concentrate on the infection among the agents. First we show that $\max\{z(t), z(t+1)\} \geq n/5$ for any $t \in \mathbb{N}$ with probability $1 - o(1/n^2)$. To prove this, we may assume $z(t) < n/5$. Then, at least $4n/5$ agents are on $V \setminus \{z\}$ at time t. Since each of these agents jumps to z with probability 1/2, we can apply the Chernoff bound [4,10] to show that $z(t+1) \geq n/5$ with probability $1 - o(1/n^2)$. As a consequence after two rounds, at least $n/5$ agents become infected with a probability of $1 - o(1/n)$.

Now we turn our attention to the infection of nodes. The probability that an uninfected node remains uninfected after t rounds is $(((n-1)/n)^{\frac{n}{5}})^t \leq (1/e)^{\frac{t}{5}}$. Hence, for some $t = \Theta(\ln n)$, we obtain the theorem. □

We have seen that agent based broadcasting can be performed very fast on a star. Let us now consider the graph G_s which is very similar to the star and is defined in the following way: The first $n/2$ nodes of G_s, labelled $0, 1, \ldots, n/2-1$, of G_s are connected mutually with each other, forming a complete graph with $n/2$ nodes, while a node $n/2+j-1$ is only connected to $j-1$. Using the techniques of Theorem 5, it can be shown that agent based broadcasting requires $\Omega(n \ln n)$ rounds. This example shows that, although the diameter of G_s is $O(1)$ and the graph is very similar to the star, broadcasting performs slowly.

5 Conclusion

In this paper, we analyzed the performance of randomized broadcasting within the agent based model. We have shown that with high probability, we can distribute within $O(\ln n)$ steps an information among the nodes of a random graph of certain density. We proved that broadcasting in a bounded degree graph can be performed with high probability in $O(D)$ steps, where D represents the diameter of the graph. We considered examples of graphs, in which agent based broadcasting is very fast or very slow. We also pointed out some differences between broadcasting in the agent based model and the traditional randomized broadcast model.

References

1. F. Ball. A threshold theorem for the reed-frost chain-binomial epidemic. *Journal of Applied Probability 20,* pages 153–157, 1983.
2. B. Bollobas. *Random Graphs.* Academic Press, 1985.
3. Hui Chen. Threshold of broadcast in random graphs. *DIMACS Tech. Rep. 97-12,* 1997.
4. H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Stat.,* 23:493–507, 1952.
5. F.R.K. Chung. *Spectral Graph Theory,* volume 92 of *CBMS Regional conference series in mathematics.* American Mathematical Society, 1997.
6. D.M. Cvetkovic, M. Doob, and H. Sachs. *Spectra of Graphs.* Johann Ambrosius Earth, 3rd edition, 1995.
7. R. Elsässer, B. Monien, and R. Preis. Diffusion schemes for load balancing on heterogeneous networks. *Theory of Computing Systems,* 35:305–320, 2002.
8. U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. *Random Structures and Algorithm,* 1(4):447–460, 1990.
9. A.M. Frieze and M. Molloy. Broadcasting in random graphs. *Discrete Applied Mathematics 54,* pages 77–79, 1994.
10. T. Hagerup and C. Rüb. A guided tour of chernoff bounds. *Information Processing Letters,* 36(6):305–308, 1990.
11. H.W. Hethcore. Mathematics of infectious diseases. *SIAM Review 42,* pages 599–653, 2000.
12. J. Hylton, K. Manheimer, F. L. Drake, B. Warsaw, R. Masse, and G. van Rossum. Knowbot programming: system support for mobile agents. *In Proc. of IWOOOS'96,* pages 8–13, 1996.
13. A. A. Kaizar, J. T. Mayes, and A. R. Mikler. Agent-based distance vector routing. *Proc. Third Intl. Workshop MATA,* pages 41–50, 2001.
14. R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. *Proc. of FOCS'00,* pages 565–574, 2000.
15. D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. *Proc. of STOC'01,* pages 163–172, 2001.
16. D. Kotz and F. Mattern. Agent systems, mobile agents, and applications. *Second International Symposium on Agent systems and Applications and Fourth International Symposium on Mobile Agents, ASA/MA, LNCS 1882,* 2000.

17. Atsuyoshi Nakamura, Naoki Abe, Hiroshi Matoba, and Katsuhiro Ochiai. Automatic recording agent for digital video server. *Proceedings of the eighth ACM international conference on Multimedia, Marina del Rey, USA,* pages 57–66, 2000.
18. M. E. J. Newman. The spread of epidemic disease on networks. *Phys. Rev. E 66, 016128,* 2002.
19. B. Pittel. On spreading rumor. *SIAM Journal on Applied Mathematics,* 47(1) :213–223, 1987.
20. T. Roughgarden and E. Tardos. How bad is selfish routing? *Proc. of FOCS'00,* pages 93–102, 2000.
21. A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Inform. and Comput.,* 82:93–133, 1989.

Approximating Earliest Arrival Flows with Flow-Dependent Transit Times*

Nadine Baumann[1] and Ekkehard Köhler[2]

[1] Max-Planck-Institut für Informatik, 66123 Saarbrücken
nadine@mpi-sb.mpg.de
[2] Technische Universität Berlin, Institut für Mathematik, 10623 Berlin
ekoehler@math.tu-berlin.de

Abstract. For the earliest arrival flow problem one is given a network $G = (V, A)$ with capacities $u(a)$ and transit times $\tau(a)$ on its arcs $a \in A$, together with a source and a sink vertex $s, t \in V$. The objective is to send flow from s to t that moves through the network over time, such that for each point in time $\theta \in [0, T)$ the maximum possible amount of flow reaches t. If, for each $\theta \in [0, T)$ this flow is a maximum flow for time horizon θ, then it is called *earliest arrival flow*. In practical applications a higher congestion of an arc in the network often implies a considerable increase in transit time. Therefore, in this paper we study the earliest arrival problem for the case that the transit time of each arc in the network at each time θ depends on the flow on this particular arc at that time θ.

For constant transit times it has been shown by Gale that earliest arrival flows exist for any network. We give examples, showing that this is no longer true for flow-dependent transit times. For that reason we define an optimization version of this problem where the objective is to find flows that are almost earliest arrival flows. In particular, we are interested in flows that, for each $\theta \in [0, T)$, need only α-times longer to send the maximum flow to the sink. We give both constant lower and upper bounds on α; furthermore, we present a constant factor approximation algorithm for this problem. Finally, we give some computational results to show the practicability of the designed approximation algorithm.

1 Introduction

Suppose you are confronted with the problem of evacuating some huge building, a stadium, or a similarly large site in a case of emergency. So you are given a system of hallways, staircases, and channels, each having a certain capacity, i.e., a maximum number of people that can enter the particular entity per time unit, together with a corresponding transit time. The objective is to send at each point in time the maximum number of people out of the building. Modeled as a graph

* The authors were supported by DFG Focus Program 1126, "Algorithmic Aspects of Large and Complex Networks".

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 599–610, 2004.
© Springer-Verlag Berlin Heidelberg 2004

problem, this reduces to a special kind of maximum dynamic flow problem (see Section 2 for the definition of dynamic flows).

Dynamic flows, also called flows over time, were first studied by Ford and Fulkerson in their seminal work on flows in networks [6,5]. While they only considered the problem of sending the maximal amount of flow from a source s to a sink t for a fixed time horizon T, Gale [7] studied a more general problem. He was interested in the question whether there exists a flow that sends not only the maximum possible amount of flow from s to t for the time horizon T but also for each $\theta \in [0, T)$. Flows satisfying this requirement are called *earliest arrival flows.*

Having shown the existence of earliest arrival flows, the next question is of course whether one can find them efficiently. Minieka [15] and Wilkinson [19] were the first who tackled this problem and designed algorithms for finding earliest arrival flows. Both algorithms rely on the successive shortest path algorithm by successively searching for shortest paths from s to t in the residual network. As shown by Zadeh [20] there are families of graphs that require the successive shortest path algorithm to have a pseudo-polynomial number of iterations. In fact, the networks given by Zadeh can be interpreted as instances of the earliest arrival flow problem, requiring the algorithm to need $\Omega(T)$ iterations and it is still an open question whether the earliest arrival flow problem can be solved in polynomial time. Hoppe and Tardos [11] were the first to present an approximation algorithm for finding earliest arrival flows where their objective is to find for a given $\varepsilon > 0$ a flow f such that for each $\theta \in [0, T)$ the amount of flow reaching t is at most an ε-fraction less than the optimum. They achieve this by using a capacity scaling approach and a so-called generalized flow decomposition. Very recently, a different approximation scheme for this problem was presented by Fleischer and Skutella [4]. They use a condensed time-expanded network for their algorithm.

All the above approaches assume transit times on the arcs to be constant. Whoever has tried to leave a stadium after a soccer game knows, however, that this assumption is far from reality. A much more realistic setting is to view transit time as a value that depends on the flow-rate, the congestion, or the amount of flow in an arc of the network. In particular, this means that the more flow units are present in an arc the higher is the transit time of this arc. This is the approach that we consider in this paper. Dynamic flows with flow-dependent transit times have been studied before. Merchant and Nemhauser [14] suggest a model where for every arc there is both a flow-dependent cost function and a so called exit function that determines the amount of traffic that can leave the arc in dependence of the amount of flow on that particular arc. While their model was both nonlinear and nonconvex and thus difficult to handle for efficient algorithms, it was influential for many other results in this area. For constant transit times a very helpful tool is the time-expanded graph, i.e., a graph that contains a copy of the original graph for each discrete point in the time horizon $[0, T)$. Carey and Subramanian [2] introduced a model that tries to use a similar approach for the case of flow-dependent transit times. Building up on this approach Köhler, Langkau, and Skutella [12] introduce a model that is both time-expanded and expands the transit time function for the case of transit

times that depend on the amount of flow entering an arc. While this expanded model only allows pseudo-polynomial algorithms, they also give a reduced version of this network. Using this, they design an efficient 2-approximation algorithm for the quickest s,t-flow problem (minimize the time horizon for sending a fixed amount of flow from a source vertex s to a sink vertex t). Hall, Langkau, and Skutella [8] extended this to an approximation scheme for this problem even for the multi-commodity case. A more precise description of transit times that depend on the flow in real application is to assume that these transit times depend not only on the amount of flow entering an arc but also on the amount of flow being currently in the whole arc. Köhler and Skutella [13] gave a model for this setting and both give a 2-approximation algorithm for the quickest s,t-flow problem and show that there cannot be a polynomial-time approximation scheme unless P=NP.

None of the above papers on flow-dependent models tackles, however, the earliest arrival problem. Most of them either study standard traffic assignment problems, the quickest flow problem or maximum flow problems. In fact, to the best of our knowledge, there has been no previous work on earliest arrival flows for the case of flow-dependent transit times in the literature. A closely related problem is the earliest arrival problem for time-dependent transit times (i.e., every arc can have for different points in time different (constant) transit times) or time-dependent capacities. For these cases the existence of earliest arrival flows was shown by Gale already in his first paper on earliest arrival flows. Ogier [16] studies the case of zero transit times where arc capacities and node storage capacities are piecewise-constant functions of time. For this special case he gives a polynomial algorithm. Recently, Fleischer [3] improved the running time of this algorithm. Tjandra [18,10] considered the earliest arrival flow problem for the case of time dependent transit times and capacities, and node capacities, correspondingly.

2 Basic Definitions

2.1 Flows over Time with Constant Transit Times

A dynamic network (G, u, τ, s, t) consists of a directed graph G together with two special nodes: the source s and the sink t; a capacity function $u : A \to \mathbb{R}^+$, and a transit time function $\tau : A \to \mathbb{R}^+$, representing the time that is needed to traverse the particular arc.

In various well known static flow problems one seeks a function $x : A \to \mathbb{R}^+$ that assigns to an arc a a flow value $x(a)$. In contrast to that, a dynamic s,t-flow f, also called a s,t-flow over time, is a function on $A \times \mathbb{R}^+$ where the second parameter denotes a *time* component—$f(a, \theta)$ describes the flow on arc a at time θ. This flow can be interpreted as flow rate, i.e., the amount of flow entering the particular arc per time unit. The flow rate entering an arc is bounded by the capacity of the arc, i.e., $f(a, \theta) \leq u(a)$. The flow function f has to satisfy flow conservation constraints not only in every node $v \in V \setminus \{s, t\}$, but also at every time $\theta \in \mathbb{R}$:

$$\sum_{a\in\delta^+(v)} \int_{-\infty}^{\theta} f(a,x)dx \leq \sum_{a\in\delta^-(v)} \int_{-\infty}^{\theta} f(a,x-\tau(a))dx. \tag{1}$$

Note that equality is not required in order to allow flow to be stored in nodes of the network. A dynamic flow f, satisfying the flow conservation constraints, is said to be *feasible* if for all arcs a and every time θ: $f(a,\theta) \leq u(a)$. For a given *time horizon* T we further assume that there is only flow in the network during the time interval $[0,T)$, i.e., $f(a,\theta) = 0$ for all $\theta \notin [0,T)$. The flow conservation constraints are now needed only within this interval. The *value* of the dynamic flow f with time horizon T is determined by the value over all time-steps $\theta \in [0,T)$ as

$$\text{value}(f) = \sum_{\delta^+(s)} \int_0^T f(a,\theta)d\theta - \sum_{\delta^-(s)} \int_0^T f(a,\theta)d\theta. \tag{2}$$

We write $\text{value}(f,\theta)$ when looking at the flow f at a specified time θ where θ is not necessarily the time horizon.

Various results for this dynamic flow model are known. One of the most remarkable ones is a theorem by Ford and Fulkerson on maximum dynamic s,t-flows for time horizon T. They defined a *temporally repeated flow* to be a static flow x with a path decomposition $\mathcal{P}$ that repeatedly sends flow over the paths of $\mathcal{P}$ as long as this flow can reach the sink before T. They show that there is always a temporally repeated flow that is a maximum dynamic s,t-flow.

2.2 Flows over Time with Flow-Dependent Transit Times

While flows over time are much more appropriate to model real-world situations than static flows, often the constant transit times do not model reality sufficiently precise. For example, it can be easily observed that there is a high correlation between the congestion of a hallway, staircases, or a road and the time needed to traverse it. A more precise method for describing this correlation is provided by the use of *flow-dependent transit times*. For this purpose we will assume in the following that a transit time function $\tau : A \times \mathbb{R}^+ \to \mathbb{R}^+$ is given as a left-continuous, non-negative, and non-decreasing function of the flow on arcs. Here, in contrast to the model with constant transit times, we are given a second parameter that refers to the flow.

While for static flows there is a canonical mapping from flows to transit times, for flows over times there are different options for such a mapping. Therefore, we will shortly mention two known models that we will make use of in the sequel.

On the one hand there are *inflow-dependent transit times* as described in Köhler, Langkau, Skutella [12]. There the transit time is a function of the amount of flow entering an arc a at time θ. In particular, flow entering arc a at the flow rate $f(a,\theta)$ needs $\tau(a,f(a,\theta))$ time to traverse the arc. Hence, flow units entering an arc at the same time have the same transit time and, while traversing the arc, their transit time keeps constant. This model allows a comparably easy

description of the dependency between flow and transit times. However, it has the disadvantage that there can be situations where flow entering at a small flow rate can pass flow units that entered the same arc at a higher flow rate before—the so called *First-In-First-Out (FIFO)* property is not fulfilled.

On the other hand there are *load-dependent transit times* as described in [13]. In this case the total amount of flow on an arc a at a time θ is used as the input to the transit time function τ; this amount of flow is called the *load* of the arc. Since the flow on an arc changes continuously, also the transit time of each unit of flow changes. Note, however, that at each point in time on an arc every unit of flow has the same transit time.

Although both of the above models cannot describe traffic flows in their whole complexity, they are however capable of modeling at least some aspects of traffic behavior.

3 Existence of Earliest Arrival Flows

Earliest arrival flows are those maximum dynamic s,t-flows that send for each point in time $\theta \in [0,T)$ the maximum amount of flow from s to t. While the existence of a maximum flow for a fixed time horizon T is obvious, it is not that clear that there are always flows over time that are not only optimal for T but also for each $\theta \in [0,T)$. Gale showed the existence of earliest arrival flows for general networks with constant transit times on the arcs and, more generally, for networks with time-dependent (but not flow-dependent) transit times and capacities on the arcs. He made use of the fact that one can model flows over time with constant or time-dependent transit times using static flows in the time-expanded network. There the standard max-flow min-cut theorem yields the result.

For the case of flow-dependent transit times it is again obvious that for any fixed time horizon T there exists a dynamic s,t-flow that sends the maximum amount of flow from s to t. It is quite natural to ask whether there is again such a maximum s,t-flow that is maximal also for each $\theta \in [0,T)$. Unfortunately, there is no simple time-expanded model for dynamic flows with flow dependent transit times that allows to reduce the time-dependent flow problem to a static one. In fact, as will be shown by the following simple counterexample, there cannot be a similarly nice existence result for earliest arrival flows for the case of flow-dependent transit times as the one by Gale.

Consider the one-arc network, shown in Figure 1, together with the simple linear transit time function given next to it and a capacity of 2 units of flow per time unit. We consider a flow model with inflow-dependent transit times. Let $T = 3$ be the considered time horizon. When sending flow from s to t at a flow rate of 2 in time interval [0,1) and at flow rate linearly decreasing from 2 to 0 in time interval [1,3), then by time 3 a flow of 4 units has reached the sink t. In fact, this is the maximum amount of flow that can be sent from s to t in this time horizon.

To construct an earliest arrival flow, we have to make sure that the maximum possible amount of flow has reached the sink for any $\theta \in [0,T)$. To show that this is not possible for this example we examine just two values of θ. Sending

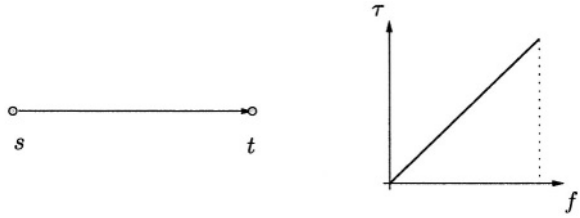


Fig. 1. Dynamic network with inflow-dependent transit times.

flow at a flow rate linearly decreasing from 2 to 0 in time interval [0,2) shows that an earliest arrival flow must send at least 2 units of flow to t up to time $\theta = 2$. In fact, sending any of the flow in this time interval at a higher flow rate would result in a decrease of the flow value reaching t up to time $\theta = 2$. It follows easily that any flow sending the maximum amount of flow up to $\theta = 2$ into t cannot send more than 2.5 units of flow into t up to $\theta = 3$. Since, however, the maximum flow for time horizon 3 is 4, this implies the following theorem.

Theorem 1. *In the setting of inflow-dependent transit times, earliest arrival flows do not exist in general.*

Similarly as for inflow-dependent transit times, there are also examples for the case of load-dependent transit times.

4 α-Earliest Arrival Flows

As shown above, in the case of flow-dependent transit times there are instances where there is no earliest arrival flow. Because of that, we are interested in related optimization problems that determine "almost earliest arrival flows". For this we introduce the problem of finding an α-earliest arrival flow to minimize the lateness of flows and show that there is a constant approximation of this new problem. We also give lower bounds.

In the sequel we will consider the following optimization problem: Find the minimum α such that there is a flow over time f that sends for each $\theta \in [0, T)$ at least as much flow into the sink t as can be sent into t up to time $\frac{\theta}{\alpha}$ by a maximum flow over time $f_{\max}(\frac{\theta}{\alpha})$ within this time horizon, i.e., $\text{value}(f, \theta) \geq \text{value}(f_{\max}(\frac{\theta}{\alpha}), \frac{\theta}{\alpha})$. Such a flow will be called an *α-earliest arrival flow.*

4.1 Upper Bound

In this section we show that there is always a 4-earliest arrival flow. More precisely, we prove that there is always a flow that sends up to every time $\theta \in [0, T)$ at least as much flow into the sink as a maximum dynamic flow sends within a time horizon of $\frac{\theta}{4}$. For showing this result we make use of a construction called *interval stacking.*

Consider a time horizon $[0,T)$. A 2-*interval stacking* of this time horizon is given by a logarithmic subdivision of this interval into a sequence of subintervals.

The first $\lfloor \log(T+1) \rfloor - 1$ intervals axe defined such that the i^{th} interval I_i is given by $I_i = [(2^i - 1), (2^{i+1} - 1))$ for $i \in \{0, 1, \ldots, \lfloor \log(T+1) \rfloor - 1\}$. Since for the remaining part of the time horizon this scheme cannot be continued, it is filled up with smaller intervals of size 2^i, for $i \in \{0, 1, \ldots, \lfloor \log(T+1) \rfloor - 1\}$ where subsequently each of these intervals is chosen as large as possible. Using this 2-interval stacking, we now construct a flow f_{IS} as follows. For every time horizon 2^i $(i = 0, 1, \ldots, \lfloor \log(T+1) \rfloor - 1)$ a maximal dynamic flow f_i is determined. Then the flow f_i is sent within the corresponding time interval of length 2^i in the interval stacking. The so constructed flow f_{IS} in the interval stacking will now be shown to give a 4-approximation for our problem.

We assume that a shortest s,t-path in our network with respect to the zero flow transit times of the arcs has length at least 1. For the case of shorter transit times we refer the reader to the extended version of the paper [1].

Define for $\theta \in [0, T)]$ the value α_θ by $\alpha_\theta = \frac{\theta}{\text{length}(I)}$ where I is the largest interval of the interval stacking lying in front of θ. For example, the value α_θ for $\theta = 10$ would be $\frac{10}{4}$. Using this definition we can conclude the following simple lemma.

Lemma 1. *Up to time θ at least* $\text{value}(f_{\max}(\frac{\theta}{\alpha_\theta}), \frac{\theta}{\alpha_\theta})$ *can reach the sink using flow f_{IS}.*

Consequently, α_θ is an upper bound on the value α that we get for the flow f_{IS} at the point θ. Thus, if we can bound α_θ for all $\theta \in [0, T)$ by a constant, we have a bound on α. However, within a particular interval I_i, $i \in \{1, \ldots, \lfloor \log(T+1) \rfloor - 1\}$ the ratio $\alpha_\theta = \frac{\theta}{\text{length}(I_{i-1})}$ is maximal for $\theta = (2^{i+1} - 1) - \varepsilon$ and $\varepsilon > 0$ small. Thus we have:

$$\alpha_\theta = \frac{(2^{i+1} - 1) - \varepsilon}{2^{i-1}} = 4 - \frac{1 + \varepsilon}{2^{i-1}},$$

implying that α_θ is bounded by 4 for this case. To prove the same ratio for θ within the remaining part of the interval stacking (i.e., for $\theta \in [2^{\lfloor \log(T+1) \rfloor} - 1, T))$ we simply prolong this rest to an interval of length $2^{\lfloor \log(T+1) \rfloor}$ and use the same calculation as above.

This proves our intended bound of 4 for the case of s,t-paths having a length with respect to the zero flow transit times of at least 1. This, together with the case of shorter transit times (see [1]), leads to the following theorem and corollary.

Theorem 2. *For any $\varepsilon > 0$ there is a dynamic s,t-flow f_ε that sends for any $\theta \in [0, T)$ at least as much flow into the sink t up to time $\theta + \varepsilon$ as is maximal for $\frac{\theta}{4}$.*

Corollary 1. *For dynamic s,t-flows with flow-dependent transit times there is always a 4-earliest arrival flow.*

It can be shown that the choice of 2 as the basis of the logarithm in the interval stacking is the best possible. A very similar scheme has been used for an online scheduling problem by Hall, Schulz, Schmoys, and Wine [9].

4.2 Lower Bound

As shown above there is a constant approximation bound for the earliest arrival flow problem with flow-dependent transit times. In this section we concentrate on lower bounds for α. For the method of interval stacking there are easy examples showing that this approach cannot achieve a better α-value than 4. In order to find lower bounds for the general problem we analyzed the inflow-dependent and the load-dependent transit time model independently.

Inflow-Dependent Transit Times

Theorem 3. *In the case of inflow-dependent transit times, for all $\varepsilon > 0$, there are instances having no α-earliest arrival flow for $\alpha \leq \frac{3}{2} - \varepsilon$.*

Proof. For the given $\varepsilon > 0$ choose an integer $k > 1$ such that $\frac{3k-1}{2k} > \frac{3}{2} - \varepsilon$. Now consider a single arc network with capacity $2k$ and the transit time function $\tau(f)$, with $\tau(f) = 1$ for $f \leq 1$ and $\tau(f) = 3k - 1$ for $f > 1$. For this network the maximum flow reaching the sink t up to time $\theta = 2k$ has value $2k - 1$ and up to time $\theta = 4k$ has value $(k+1)2k + (3k-2) = 2k^2 + 5k - 2$.

Suppose there is an α-earliest arrival flow f^* with $\alpha = \frac{3k-1}{2k}$. Thus, at time $\theta_1 = 3k - 1$, f^* has to send at least as much flow into the sink as is maximal for $\frac{\theta_1}{\alpha} = 2k$ and, similarly, at time $\theta_2 = 6k - 2$, f^* has to send at least as much flow into t as is maximal for $\frac{\theta_2}{\alpha} = 4k$.

Since f^* has to satisfy $\text{value}(f^*, 3k-1) \geq \text{value}(f_{\max}(2k), 2k) = 2k - 1$, it has to send at least $2k - 1$ units of flow at the lower rate of 1 from s to t during the interval $[0, 3k-1)$. Thus, up to time $3k - 1$ the flow f^* can send no more than $(2k-1) \cdot 1 + ((3k-1) - (2k-1)) \cdot 2k = 2k - 1 + 2k^2$ units of flow out of the source. On the other hand, f^* cannot send more than $3k - 2$ units from s to t during the interval $[3k-1, 6k-2)$. This implies that $\text{value}(f^*, 6k-2) \leq 3k - 2 + 2k - 1 + 2k^2 = 2k^2 + 5k - 3$. This is a contradiction to f^* being an α-earliest arrival flow, since $\text{value}(f_{\max}(4k), 4k) = 2k^2 + 5k - 2 > 2k^2 + 5k - 3$. □

Remark. Note that the transit time function in the example can easily be transformed into a continuous function, without changing the bound given in Theorem 3.

Load-Dependent Transit Times. A similar example as for the inflow-dependent case can also be constructed for the load-dependent model. For this result and the corresponding proof we refer the reader to the extended version of the paper [1].

5 An Approximation Algorithm

Using the interval stacking approach one cannot only show upper bounds for the α-earliest arrival flow but can also create algorithms for determining such a

flow. In contrast to the case of flows over time with constant transit times, the maximum flow problem for flow-dependent transit times is an NP-hard problem (see [8,13]). Although no approximation algorithms for this problem are known, there are approximation algorithms for the closely related quickest flow problems, i.e., for the problem to send a fixed amount of flow as quickly as possible from s to t. A c-approximation algorithm for this problem determines a flow that sends the given amount of flow in no more than c times the optimal time from s to t. As we will show in the following, these approximation algorithms for the quickest flow problem can be used together with the interval stacking to get approximation results also for the earliest arrival flow problem.

Since we will use the interval stacking method from Section 4, for simplicity, we from now on assume that the shortest path distance is at least 1.

Lemma 2. *Suppose there is a dynamic flow algorithm A that computes for any given time horizon $c \cdot T$ ($c \geq 1$) a flow of value at least as large as the value for a maximum flow for time horizon T. Then there is a $4c$-approximation algorithm for the α-earliest arrival flow problem.*

Proof. In a first step algorithm A is used to compute for each $T = 2^i$ with $i \in \{0, \ldots, \lfloor \log(T+1) \rfloor - 1\}$ a flow for time horizon cT. By the assumption of the lemma, each of those flows has a value at least as large as a maximal flow for time horizon T.

In a second step the so computed flows are put together in the same way as the flow f_{IS} in Section 4.1, with the difference that the length of the intervals of the stacking is not 2^i but $c2^i$ for $i \in \{0, \ldots, \lfloor \log(T+1) \rfloor - 1\}$. By the same argument as used for proving Theorem 2, the so constructed flow can be shown to be a $4c$-earliest arrival flow. □

Although the above lemma seems to be rather restrictive, it is applicable both for the inflow-dependent and the load-dependent model. In both cases the known approximation algorithms for the quickest flow problem satisfy the conditions of Lemma 2. The following theorems from [12] and [13] summarize the corresponding results for the two models.

Theorem 4 (Theorem 1 in [12]). *Consider an instance of the quickest flow problem with inflow-dependent transit times where all transit time functions are non-decreasing step functions. If there is a flow over time with inflow-dependent transit times sending D units of flow from s to t within time T, then there exists a temporally repeated flow with inflow-dependent transit times satisfying demand D within time horizon at most 2T. Moreover, such a flow can be computed in strongly polynomial time.*

Theorem 5 (Theorem 3.1 in [13]). *If there is a flow over time which sends D units of flow from s to t within time T, then there exists a temporally repeated flow satisfying demand D within time horizon at most 2T. Moreover, for every $\varepsilon > 0$, one can compute a temporally repeated flow in polynomial time which satisfies demand D within time horizon at most $(2+\varepsilon)T$.*

Also the recent approximation scheme for the quickest flow problem in the inflow-dependent model (see [8]) can be used for approximating an α-earliest arrival flow, following the approach of Lemma 2. The idea of the approximation scheme in [8] is to first compute a relaxed flow for a time horizon T. This relaxed flow is not necessarily feasible for the inflow-dependent setting, however, the construction can be used to show that this relaxed flow gives an upper bound on the amount of flow that a maximal dynamic flow can send during that time horizon in the inflow-dependent model. In a second step the relaxed flow is augmented to be a feasible flow in the inflow-dependent model where this augmentation does not change the value of the flow and enlarges the necessary time horizon by no more than a factor of $1+\varepsilon$ (we omit further details of this algorithm and refer to [8]).

Summing up the observations on the quickest flow algorithms from [8,12,13], we can draw the following corollary.

Corollary 2. *There are* $4c$*-approximation algorithms for the* α*-earliest arrival flow problem, where* $c = 2$ *for the inflow- and load-dependent dynamic flow model and* $c = 1+\varepsilon$ *for every* $\varepsilon > 0$ *for the inflow-dependent dynamic flow model.*

6 Computational Results

In the last section we provided the method of using an interval stacking to find a constant upper bound for the α-earliest arrival flow problem in the case of flow-dependent transit times. Since the given upper bound of 4 only roughly estimates the quality of the approximation, it is interesting to ask which values of α are reachable in practice with this approach. Due to the fact that there are no known exact algorithms to compute maximum flows for flow-dependent transit times we implemented an 8-approximation algorithm for the inflow-dependent model using the 2-interval stacking together with the quickest flow approximation algorithm described in [12] (see Corollary 2) and conducted some computational tests. In this section we present a few results for a very small real world network that represents a large university building. Of course such a small example does not allow to draw conclusions about the practical quality of the approximation in general, however, it proves at least that the approach is not only of theoretical interest and gives an idea about the gap between the theoretical and practical ratio.

The network consists of 22 nodes and 66 arcs. As transit time function we used the B.P.R. function which is a well accepted transit time function for traffic networks (see [17] for further details on the B.P.R. function). This function is given by $\tau(x) = t_a^0 \left(1 + \gamma \left(\frac{x}{c'_a}\right)^\beta\right)$, where t_a^0 is the transit time of arc a when a is empty, c'_a is the "practical capacity" of a; the parameter γ is set to $\gamma = 0.15$ and for β the values $\beta \in \{1, 2, 4, 6\}$ are chosen. As zero flow transit time t_a^0 we used three different values to simulate the situation for three different groups of people: a group of senior citizens, a group of 'normal' people and a group of rather sportive people. The aim is to send as many people as possible as early

as possible from a central hall of the building to the area in front of the building within a time horizon of 5000 seconds.

Theoretically we know that at least as much flow reaches the sink at a time θ as is maximal for a time $\frac{\theta}{8}$. However, the actual performance of the algorithm for our small example turns out to be much better; for big θ even near-optimal. The fastest people in the group of sportsmen need at least 96 seconds and start with an α-value of 4.8; senior citizens need nearly 400 seconds but have an α-value of no more than 4.0. For later points in time the α-value decreases to a nearly constant value of about 1.3 for sportsmen (after 1500 seconds) and for the slow people to a value between 2.0 and 1.5 (after 2800 seconds).

Another value of interest is the relation between the maximum flow compared to the amount of flow sent into the sink by the approximation algorithm for each θ $(0 \leq \theta < T)$. Note that here flow values are compared while for determining the α-value one has to compare time steps. The observation here is that the fraction of the flow sent by the approximation compared to the maximal flow[1] is small for small values of θ but rather high for large θ. Starting at a ratio of zero at the already stated points in time (96 seconds for sportsmen and 400 seconds for senior citizens) the sportsmen's curve reaches a ratio of one half already at time 450 and a ratio over 80 percent at time 2800. The flow that models senior citizens reaches a value of 50 percent of the maximal flow at time 3200 and increases this value to a maximum of slightly more than 60 percent at time 4000 (we considered a time horizon of 5000 seconds).

Besides these comparisons there are other interesting aspects to examine. The amount of arcs and nodes of the expanded graph depends on the chosen parameters for β and the accuracy of the step function approximating the transit time function (for details see [12]). As accuracy we choose parameters $\eta = 0.5, 0.25, 0.125$. The left of the following tables shows this dependency by giving the number of arcs of the expanded graph for different values of η and β. The right table, on the other hand, gives some running times in seconds for a high precision of the approximated step function (the tests were conducted on a 1.7 GHz AMD Athlon™ XP2100+ with 512 MByte memory).

$\eta \setminus \beta$	1	2	4	6
0.125	1848	5280	12672	20064
0.25	1056	2772	6732	10560
0.5	528	1584	3696	5808

$\eta \setminus \beta$	1	2	4	6
0.0125	17	79	418	750
0.025	4	22	145	286
0.05	2	5	43	97

Summing up, the comparably good quality of the practical approximation ratio, the good ratio when comparing the maximal flow to the flow computed by the approximation algorithm, and the running times suggest that the developed approximation method gives a practically efficient tool for computing good approximate solutions for the earliest arrival flow problem also for larger instances than just this small example.

[1] Since there is no known polynomial time algorithm that can compute maximum flows in networks with inflow-dependent transit times, we used a relaxation from [12] for getting an upper bound on the value of a maximum flow for this comparison.

References

1. N. Baumann and E. Köhler. Approximating earliest arrival flows with flow-dependent transit times, 2004. *http://www.math.tu-berlin.de/~ekoehler.*
2. M. Carey and E. Subrahmanian. An approach to modeling time-varying flows on congested networks. *Transportation Research Part B,* 34:157–183, 2000.
3. L. Fleischer. Universally maximum flow with piecewise-constant capacities. In Cornuejols et al., editor, *IPCO'99,* LNCS 1610, pages 151–165, 1999.
4. L. Fleischer and M. Skutella. Minimum cost flows over time without intermediate storage. In *Proceedings of SODA'03,* pages 66–75, 2003.
5. L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.
6. L.R. Ford and D.R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research,* 6:419–433, 1958.
7. D. Gale. Transient flows in networks. *Michigan Math. Journal,* pages 59–63, 1959.
8. A. Hall, K. Langkau, and M. Skutella. An FPTAS for quickest multicommodity flows with inflow-dependent transit times. In Arora et al., editor, *Proc. of APPROX'03,* LNCS 2764, pages 71–82, 2003.
9. L.A. Hall, A. Schulz, D.B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. In *Proc. of SODA'96,* pages 142–151, 1996.
10. H. W. Hamacher and S. A. Tjandra. Earliest arrival flow model with time dependent capacity for solving evacuation problems. In *Pedestrian and Evacuation Dynamics,* pages 267–276, 2002.
11. B. Hoppe and É. Tardos. Polynomial time algorithms for some evacuation problems. In *Proc. of SODA'94,* pages 433–441, 1994.
12. E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *Proc. of ESA 02,* pages 599–611, 2002.
13. E. Köhler and M. Skutella. Flows over time with load-dependent transit times. In *Proc. of SODA'02,* pages 174–183, San Francisco, CA, 2002.
14. D. K. Merchant and G. L. Nemhauser. A model and an algorithm for the dynamic traffic assignment problems. *Transportation Science,* 12(3):183–199, 1978.
15. E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research,* 21:517–527, 1973.
16. R. G. Ogier. Minimum-delay routing in continuous time dynamic networks with piecewise-constant capacities. *Networks,* 18:303–318, 1988.
17. Y. Sheffi. *Urban Transportation Networks.* Prentice-Hall, New Jersey, 1985.
18. S.A. Tjandra. *Dynamic network optimization with application to the evacuation problem.* PhD thesis, Fachbereich Mathematik der Universität Kaiserslautern, 2003.
19. W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research,* 19:1602–1612, 1971.
20. N. Zadeh. A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming,* 5:255–266, 1973.

A Hierarchy of Irreducible Sofic Shifts

Marie-Pierre Béal, Francesca Fiorenzi, and Dominique Perrin

Institut Gaspard-Monge, Université de Marne-la-Vallée
77454 Marne-la-Vallée Cedex 2, France
{beal,perrin,fiorenzi}@univ-mlv.fr

Abstract. We define new subclasses of the class of irreducible sofic shifts. These classes form an infinite hierarchy where the lowest class is the class of almost finite type shifts introduced by B. Marcus. We give effective characterizations of these classes with the syntactic semigroups of the shifts.

Keywords: Automata and formal languages, symbolic dynamics.

1 Introduction

Sofic shifts [18] are sets of bi-infinite labels in a labeled graph. If the graph can be chosen strongly connected, the sofic shift is said to be irreducible. An irreducible sofic shift has a unique (up to isomorphisms of automata) minimal deterministic presentation called its right Fischer cover. A particular subclass of sofic shifts is the class of shifts of finite type which are defined by a finite set of forbidden blocks. Two sofic shifts X and Y are conjugate if there is a bijective block map from X onto Y. It is an open question to decide whether two sofic shifts are conjugate, even in the particular case of irreducible shifts of finite type.

Almost finite type shifts have been introduced in [13] (see also [15]). They constitute a meaningful intermediate class above the class of shifts of finite type for several reasons. For instance, if $\tilde{X}$ is the shift presented by the reversed presentation of a shift X that has almost finite type, then X and $\tilde{X}$ are conjugate [6]. Almost finite type shifts are of practical interest in coding for constrained channels. Sliding block decoding theorems hold in the case of almost finite type constraints while they do not hold beyond this class [10].

In this article, we first give a characterization of almost finite type shifts based on the syntactic semigroup of the shift. This semigroup S is the transition semigroup of the right Fischer cover of the irreducible sofic shift. The structure of a finite semigroup is determined by the Green's relations (denoted $\mathcal{R}$, $\mathcal{L}$, $\mathcal{H}, \mathcal{D}, \mathcal{J}$), see for instance [17]. We show that an irreducible sofic shift has almost finite type if and only if for any regular $\mathcal{H}$-class of S with image I and any $\mathcal{R}$-class of the $\mathcal{D}$-class of rank 1 with domain D, the intersection $D \cap I$ has at most one element. In general, the greatest cardinality of $D \cap I$, where I is an image and D is a domain as above, is called the degree of the shift. This enables the definition of a hierarchy of subclasses of irreducible sofic shifts with respect to this degree, where the lowest class (that with degree 1) is the class of almost

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 611–622, 2004.
© Springer-Verlag Berlin Heidelberg 2004

finite type shifts. In particular, we prove that conjugate irreducible sofic shifts have the same degree. This degree is thus a conjugacy invariant.

The proof of this invariant uses Nasu's Classification Theorem for sofic shifts [16] that extends William's one for shifts of finite type. This theorem says that two irreducible sofic shifts X, Y are conjugate if and only if there is a sequence of symbolic adjacency matrices of right Fischer covers $A = A_0, A_1, \ldots, A_{l-1}, A_l = B$, such that A_{i-1} and A_i are elementary strong shift equivalent for $1 \leq i \leq l$, where A and B are the adjacency matrices of the right Fischer covers of X and Y, respectively. This means that, for each i, there are two symbolic matrices U_i and V_i such that, after recoding the alphabets of A_{i-1} and A_i, one has $A_{i-1} = U_iV_i$ and $A_i = V_iU_i$. A bipartite shift is associated in a natural way to a pair of elementary strong shift equivalent and irreducible sofic shifts [16]. Another syntactic conjugacy invariant called the syntactic graph of the sofic shift was defined in [2]. We give an example of two non-almost finite type shifts with different degrees (and therefore not conjugate) that have the same syntactic graph.

Basic definitions related to symbolic dynamics are given in Section 2.1. We refer to [12] or [11] for more details. See also [13], [12, Section 13.1], [15], [10],[5], [6], [19] and [7] about almost finite type shifts. Basic definitions and properties related to finite semigroups and their structure are given in Section 2.2. We refer to [17, Chapter 3] for a more comprehensive expository. Nasu's Classification Theorem is recalled in Section 2.3. In Section 3, we define a hierarchy of irreducible sofic shifts. In Section 4, we consider the problem of characterizing classes of shifts (as the class of almost finite type shifts), by algebraic properties of the syntactic semigroup.

2 Definitions and Background

2.1 Almost Finite Type Shifts and Their Presentations

Let $\mathcal{A}$ be a finite alphabet, i.e. a finite set of symbols. The shift map $\sigma : \mathcal{A}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ is defined by $\sigma((a_i)_{i\in\mathbb{Z}}) = (a_{i+1})_{i\in\mathbb{Z}}$, for $(a_i)_{i\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}}$. If $\mathcal{A}^{\mathbb{Z}}$ is endowed with the product topology of the discrete topology on $\mathcal{A}$, a *shift* is a closed σ-invariant subset of $\mathcal{A}^{\mathbb{Z}}$.

A finite *automaton* is a finite multigraph labeled by $\mathcal{A}$. It is denoted $\mathsf{A} = (Q,E)$, where Q is a finite set of states, and E a finite set of edges labeled by $\mathcal{A}$. It is equivalent to a *symbolic adjacency* $(Q \times Q)$*-matrix* A, where A_{pq} is the finite formal sum of the labels of all the edges from p to q. A *sofic shift* is the set of the labels of all the bi-infinite paths on a finite automaton. If A is a finite automaton, we denote by X_A the sofic shift defined by the automaton A. Several automata can define the same sofic shift. They are also called *presentations* or *covers* of the sofic shift. We will assume that all presentations are *essential:* all states have at least one outgoing edge and one incoming edge. An automaton is *deterministic* if for any given state and any given symbol, there is at most one outgoing edge labeled by this given symbol. An automaton is *left*

closing with delay D if whenever two paths of length $D + 1$ end at the same state and have the same label, then they have the same final edge. An automaton is *left closing* if it is left-closing with some delay $D \geq 0$. A sofic shift is *irreducible* if it has a presentation with a strongly connected graph. Irreducible sofic shifts have a unique (up to isomorphisms of automata) *minimal deterministic presentation,* that is a deterministic presentation having the fewest states among all deterministic presentations of the shift. This presentation is called the *right Fischer cover* of the shift.

An irreducible sofic shift has *almost finite type* (AFT) if it has a deterministic and left-closing presentation. The class of almost finite type shifts was introduced by B. Marcus in [13], see also [15] and [12, Section 13.1].

Let $\mathsf{A} = (Q, E)$ be a deterministic automaton labeled by $\mathcal{A}$. The *square* of A is the deterministic automaton $(Q \times Q, F)$ where $(p, q) \xrightarrow{a} (p', q') \in F$ if and only if $p \xrightarrow{a} p'$ and $q \xrightarrow{a} q' \in E$. A *diagonal* state of the square of A is a state (p, p), with $p \in Q$.

An almost finite type shift is an irreducible shift whose right Fischer cover is left-closing. Thus the square of its right Fischer cover has no strongly connected component with at least one edge containing a non-diagonal state and admitting a path going from this component to a diagonal state (see for instance [13], [1]). Checking whether an irreducible sofic shift has almost finite type can thus be done in a quadratic time in the number of states of the right Fischer cover of the shift.

2.2 The Syntactic Semigroup of an Irreducible Sofic Shift

In this section, we recall the definition and the structure of the syntactic semigroup of an irreducible sofic shift [2].

Let $\mathsf{A} = (Q,E)$ be a finite deterministic (essential) automaton on the alphabet $\mathcal{A}$. Each finite word w of $\mathcal{A}^*$ defines a partial function from Q to Q. This function sends the state p to the state q, if w is the label of a path from p to q. The semigroup generated by all these functions is called the *transition semigroup* of the automaton. When X_A is not the full shift, the semigroup has a null element, denoted 0, which corresponds to words which are not factors of any bi-infinite word of X_A. The *syntactic semigroup* of an irreducible sofic shift is defined as the transition semigroup of its right Fischer cover.

Given a semigroup S, we denote by S^1 the following monoid: if S is a monoid, $S^1 = S$. If S is not a monoid, $S^1 = S \cup \{1\}$ together with the law $*$ defined by $x * y = xy$ if $x, y \in S$ and $1 * x = x * 1 = x$ for every $x \in S^1$.

We recall the *Green's relations* $\mathcal{R}$, $\mathcal{L}$, $\mathcal{H}$, $\mathcal{J}$, which are fundamental equivalence relations defined in a semigroup S. They are defined as follows. Let $x, y \in S$,

$$
\begin{aligned}
&x\mathcal{R}y \Leftrightarrow xS^1 = yS^1,\\
&x\mathcal{L}y \Leftrightarrow S^1x = S^1y,\\
&x\mathcal{J}y \Leftrightarrow S^1xS^1 = S^1yS^1,\\
&x\mathcal{H}y \Leftrightarrow x\mathcal{R}y \text{ and } x\mathcal{L}y.
\end{aligned}
$$

Another relation $\mathcal{D}$ is defined by:

$$x\mathcal{D}y \Leftrightarrow \exists z \in S \ x\mathcal{R}z \text{ and } z\mathcal{L}y.$$

In a finite semigroup $\mathcal{J} = \mathcal{D}$.

An *$\mathcal{R}$-class* is an equivalence class for a relation $\mathcal{R}$ (similar notations hold for the other Green's relations). An *idempotent* is an element $e \in S$ such that $ee = e$. A *regular* class is a class containing an idempotent. In a regular $\mathcal{D}$-class, any $\mathcal{H}$-class containing an idempotent is a maximal subgroup of the semigroup. Moreover, two regular $\mathcal{H}$-classes contained in a same $\mathcal{D}$-class are isomorphic (as groups), see for instance [17, Chapter 3 Proposition 1.8].

We say that two elements $x, y \in S$ are *conjugate* if there are elements $u, v \in S^1$ such that $x = uv$ and $y = vu$.

Let S be a transition semigroup of an automaton $\mathsf{A} = (Q, E)$ and $x \in S$. The *rank* of x is the cardinal of the image of x as a partial function from Q to Q. The *kernel* of x is the partition induced by the equivalence relation $\sim$ over the domain of x where $p \sim q$ if and only p, q have the same image under x. We describe the so called "egg-box" pictures with the sofic shifts of Figure 1 and Figure 2 which have almost finite type and not almost finite type, respectively.

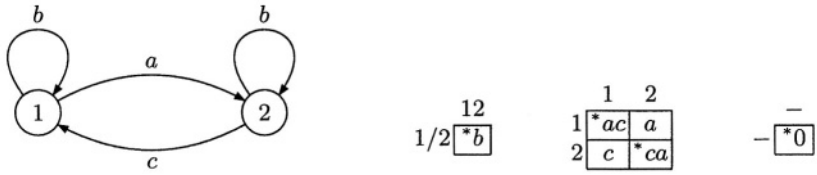


Fig. 1. An irreducible sofic shift which has almost finite type. Its syntactic semigroup is represented on the right part of the figure. It is composed of three $\mathcal{D}$-classes of rank 2, 1 and 0, respectively, represented by the above tables from left to right. Each square in a table represents an $\mathcal{H}$-class. Each row represents an $\mathcal{R}$-class and each column an $\mathcal{L}$-class. The common kernel of the elements in each row is written on the left of each row. The common image of the elements in each column is written above each column. Idempotents are marked with the symbol $*$. Each $\mathcal{D}$-class of this semigroup is regular.

The syntactic semigroup of an irreducible sofic shift has a unique $\mathcal{D}$-class of rank 1 which is regular (see for instance [3] or [4], and also [9]). Moreover, if u is a non null element of this semigroup, there is a word w such that uw has rank 1.

2.3 Nasu's Classification Theorem for Sofic Shifts

In this section, we recall Nasu's Classification Theorem for sofic shifts [16] (see also [12, Theorem 7.2.12]), which extends William's Classification Theorem for shifts of finite type (see [12, Theorem 7.2.7]).

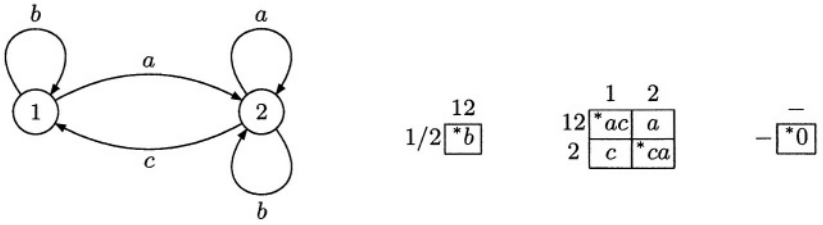


Fig. 2. An irreducible sofic shift which has not almost finite type. Indeed, there are two distinct left-infinite paths labelled ...*bbbbbba* ending at state 2. Also in this case, each $\mathcal{D}$-class is regular.

Let $X \subset \mathcal{A}^{\mathbb{Z}}$, $Y \subset \mathcal{B}^{\mathbb{Z}}$ be two shifts and m, a be nonnegative integers. A map $\phi : X \to Y$ is a (m, a)-*block map* (or (m, a)-*factor map)* if there is a map $\delta : \mathcal{A}^{m+a+1} \to \mathcal{B}$ such that $\phi((a_i)_{i\in\mathbb{Z}}) = (b_i)_{i\in\mathbb{Z}}$ where $\delta(a_{i-m} \dots a_{i-1}\ a_i\ a_{i+1} \dots a_{i+a}) = b_i$. A *block map* is a (m, a)-*block map* for some nonnegative integers m, a (respectively called its *memory* and *anticipation*). The well known theorem of Curtis, Hedlund, and Lyndon [8] asserts that continuous maps commuting with the shift map σ, are exactly block maps. A *conjugacy* is a one-to-one and onto block map (then, being a shift compact, also its inverse is a block map).

Having almost finite type is a property of shifts which is invariant under conjugacy [13].

Let A be the symbolic adjacency $(Q \times Q)$-matrix of an automaton A with entries in a finite alphabet $\mathcal{A}$. Let $\mathcal{B}$ be a finite alphabet and f a one-to-one map from $\mathcal{A}$ to $\mathcal{B}$. The map f is extended to a morphism from finite formal sums of elements of $\mathcal{A}$ to finite formal sums of elements of $\mathcal{B}$. We say that f *transforms* A into a symbolic $(Q \times Q)$-matrix B if $B_{pq} = f(A_{pq})$ for each $p, q \in Q$.

We now define the notion of strong shift equivalence between two symbolic adjacency matrices.

Let $\mathcal{A}$ and $\mathcal{B}$ be two finite alphabets. We denote by $\mathcal{AB}$ the set of words ab with $a \in \mathcal{A}$ and $b \in \mathcal{B}$.

Two symbolic matrices A and B with entries in $\mathcal{A}$ and $\mathcal{B}$ respectively, are *elementary strong shift equivalent* if there is a pair of symbolic matrices (U, V) with entries in disjoint alphabets $\mathcal{U}$ and $\mathcal{V}$ respectively, such that there is a one-to-one map from $\mathcal{A}$ to $\mathcal{UV}$ which transforms A into UV, and there is a one-to-one map from $\mathcal{B}$ to $\mathcal{VU}$ which transforms B into VU.

Two symbolic adjacency matrices A and B are *strong shift equivalent within right Fischer covers* if there is a sequence of symbolic adjacency matrices of right Fischer covers

$$A = A_0, A_1, \dots, A_{l-1}, A_l = B$$

such that for $1 \leq i \leq l$ the matrices A_{i-1} and A_i are elementary strong shift equivalent.

Theorem 1 (Nasu). *Let X and Y be irreducible sofic shifts and let A and B be the symbolic adjacency matrices of the right Fischer covers of X and Y,*

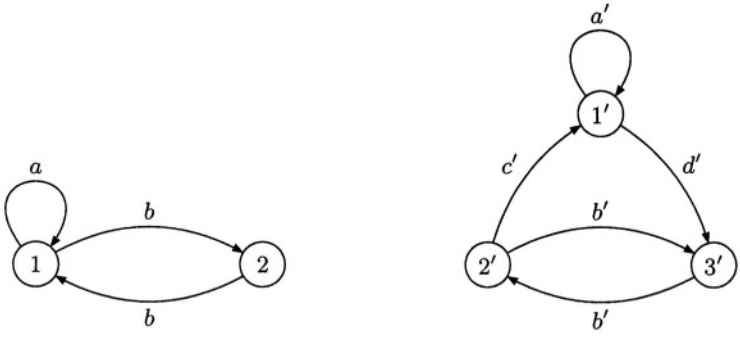


Fig. 3. Two conjugate shifts X and Y.

respectively. Then X and Y are conjugate if and only if A and B are strong shift equivalent within right Fischer covers.

Let us consider the two irreducible sofic shifts X and Y defined by the right Fischer covers in Figure 3. The symbolic adjacency matrices of these automata are respectively

$$A = \begin{bmatrix} a & b \\ b & 0 \end{bmatrix}, \quad B = \begin{bmatrix} a' & 0 & d' \\ c' & 0 & b' \\ 0 & b' & 0 \end{bmatrix}.$$

Then A and B are elementary strong shift equivalent with

$$U = \begin{bmatrix} u_1 & 0 & u_2 \\ 0 & u_2 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} v_1 & 0 \\ v_2 & 0 \\ 0 & v_2 \end{bmatrix}.$$

Indeed,

$$UV = \begin{bmatrix} u_1v_1 & u_2v_2 \\ u_2v_2 & 0 \end{bmatrix}, \quad VU = \begin{bmatrix} v_1u_1 & 0 & v_1u_2 \\ v_2u_1 & 0 & v_2u_2 \\ 0 & v_2u_2 & 0 \end{bmatrix}.$$

The one-to-one maps from $\mathcal{A} = \{a, b\}$ to $\mathcal{U}\mathcal{V}$ and from $\mathcal{B} = \{a', b', c', d'\}$ to $\mathcal{V}\mathcal{U}$ are described in the tables below.

a	u_1v_1
b	u_2v_2

,

a'	v_1u_1
b'	v_2u_2
c'	v_2u_1
d'	v_1u_2

.

An elementary strong shift equivalence between $\mathsf{A} = (Q, E)$ and $\mathsf{B} = (Q', E')$, enables the construction of an irreducible sofic shift Z on the alphabet $\mathcal{U} \cup \mathcal{V}$ as follows. The sofic shift Z is defined by the automaton $\mathsf{C} = (Q \cup Q', F)$, where the symbolic adjacency matrix C of C is

$$\begin{array}{c} \\ Q \\ Q' \end{array} \begin{array}{c} \begin{array}{cc} Q & Q' \end{array} \\ \begin{bmatrix} 0 & U \\ V & 0 \end{bmatrix} \end{array}.$$

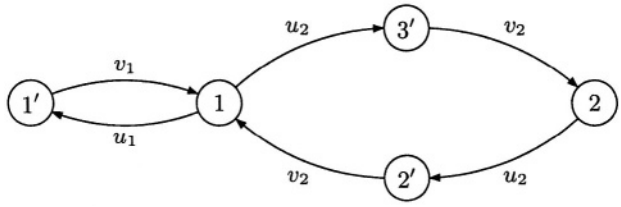


Fig. 4. The bipartite shift Z of the shifts X and Y in Figure 3.

The shift Z is called the *bipartite shift* defined by U, V (see Figure 4). An edge of C labeled by $\mathcal{U}$ goes from a state in Q to a state in Q'. An edge of C labeled by $\mathcal{V}$ goes from a state in Q' to a state in Q. Remark that C is a right Fischer cover (i.e. is minimal).

3 A Hierarchy of Irreducible Sofic Shifts

In this section, we first give a syntactic characterization of almost finite type shifts. Next, we define a hierarchy of irreducible sofic shifts.

Proposition 1. *Let X be an irreducible sofic shift and S its syntactic semigroup. Then X has almost finite type if and only if for any regular $\mathcal{H}$-class of S with image I and any $\mathcal{R}$-class of the $\mathcal{D}$-class of rank 1 with domain D, the intersection $D \cap I$ has at most one element.*

PROOF Let us assume that X has not almost finite type. Let A be the right Fischer cover of X. Then there are two states $p \neq q$ and a state r, two words u, v and two paths on A labeled uv as follows.

$$p \xrightarrow{u} p \xrightarrow{v} r$$
$$q \xrightarrow{u} q \xrightarrow{v} r$$

There is a positive integer n that u^n is a non null idempotent in S. Let I be the image of u^n. It contains p and q. Let w be a word of rank 1 in S. Since A has a strongly connected graph, there is a word w' such that the domain of $w'w$ contains the state r. The word $vw'w$ has rank 1 and its domain contains p and q. Thus the intersection of the domain of $vw'w$ and I contains at least two elements.

Conversely, let us assume that there is in S a regular $\mathcal{H}$-class H with image I and an $\mathcal{R}$-class R of the $\mathcal{D}$-class of rank 1 with domain D, such that $D \cap I$ has at least two elements. Let e be an idempotent of H. Then e induces the identity map on its image I. Let $p \neq q \in D \cap I$, and $v \in R$. Then there is a state r and two paths on A as follows

$$p \xrightarrow{e} p \xrightarrow{v} r$$
$$q \xrightarrow{e} q \xrightarrow{v} r$$

It follows that X has not almost finite type. □

For instance, the shift presented in Figure 2 has not almost finite type since it has a regular $\mathcal{H}$-class of rank 2 (containing the idempotent b) whose image is $\{1,2\}$. This image intersects the domain of $\{ac, a\}$ (which is a $\mathcal{R}$-class contained in the $\mathcal{D}$-class of rank 1), with $\{1,2\}$ as intersection.

We now introduce the following classification of irreducible sofic shifts. An irreducible sofic shift is *d-non-left closing* if its syntactic semigroup has a regular $\mathcal{H}$-class with image I and an $\mathcal{R}$-class of the $\mathcal{D}$-class of rank 1 with domain D, such that $D \cap I$ has d elements. An irreducible sofic shift has *degree* d if it is d-non-left closing with $d \geq 0$ and not d'-non-left closing for any $d' > d$. Notice that the degree of an irreducible sofic shift is always non null.

The following proposition allows us to prove that the class of irreducible sofic shifts with degree d is a class of shifts invariant under conjugacy. In this classification, the class of almost finite type shifts is the class of irreducible sofic shifts with degree 1. This enables to recover that the property of having almost finite type is a conjugacy invariant.

Proposition 2. *Let X and Y be two conjugate irreducible sofic shifts and let d be a positive integer. If X is d-non-left closing, then Y is d-non-left closing.*

Before proving Proposition 2, we recall some results from [2] about the syntactic semigroup of a bipartite shift. Let X (respectively Y) be an irreducible sofic shift whose symbolic adjacency matrices of its right Fischer cover is a $(Q \times Q)$-matrix (respectively $(Q' \times Q')$-matrix) denoted by A (respectively by B). We assume that A and B are elementary strong shift equivalent through a pair of matrices (U, V). The corresponding alphabets are denoted $\mathcal{A}$, $\mathcal{B}$, $\mathcal{U}$, and $\mathcal{V}$ as before. We denote by f a one-to-one map from $\mathcal{A}$ to $\mathcal{UV}$ which transforms A into UV and by g a one-to-one map from $\mathcal{B}$ to $\mathcal{VU}$ which transforms B into VU. Let Z be the bipartite irreducible sofic shift associated to U, V. We denote by S (respectively T, R) the syntactic semigroup of X (respectively Y, Z).

Let $w \in R$. If w is non null, the bipartite nature of Z implies that w is a function from $Q \cup Q'$ to $Q \cup Q'$ whose domain is included either in Q or in Q', and whose image is included either in Q or in Q'. If $w \neq 0$ with a domain included in P and an image included in P', we say that w has the *type* (P, P'). Remark that w has type (Q, Q) if and only if $w \neq 0$ and $w \in (f(\mathcal{A}))^*$, and w has type (Q', Q') if and only if $w \neq 0$ and $w \in (g(\mathcal{B}))^*$. Elements of R in a same non null $\mathcal{H}$-class have the same type.

Let $w = a_1 \dots a_n$ be an element of S, we define the element $f(w)$ as $f(a_1) \dots f(a_n)$. Note that this definition is consistent since if $a_1 \dots a_n = a'_1 \dots a'_m$ in S, then $f(a_1) \dots f(a_n) = f(a'_1) \dots f(a'_m)$ in R. Similarly we define an element $g(w)$ for any element w of T.

Conversely, let w be an element of R belonging to $f(\mathcal{A})^*$ $(\subset (\mathcal{UV})^*)$. Then $w = f(a_1) \dots f(a_n)$, with $a_i \in \mathcal{A}$. We define $f^{-1}(w)$ as $a_1 \dots a_n$. Similarly we define $g^{-1}(w)$. Again these definitions and notations are consistent. Thus f is a semigroup isomorphism from S to the subsemigroup of R of transition functions defined by the words in $(f(\mathcal{A}))^*$. Notice that $f(0) = 0$ if $0 \in S$. Analogously, g is a semigroup isomorphism from T to the subsemigroup of R of transition functions defined by the words in $(g(\mathcal{B}))^*$.

We now prove Proposition 2. PROOF[of Proposition 2] By Nasu's Theorem [16] we can assume, without loss of generality, that the symbolic adjacency matrices of the right Fischer covers of X and Y are elementary strong shift equivalent. We define the bipartite shift Z as above. We denote by S, T and R the syntactic semigroups of X, Y and Z respectively.

Let us assume that X is d-non-left closing. Thus S has a regular $\mathcal{H}$-class H with image I and an $\mathcal{R}$-class of the $\mathcal{D}$-class of rank 1 with domain D, such that $D \cap I$ has d elements. Let e be an idempotent of H. It induces the identity map on its image I.

The element $f(e)$ is an idempotent element of type (Q, Q) in R. Let $u_1v_1 \dots u_nv_n \in (\mathcal{U}\mathcal{V})^*$ such that $f(e) = u_1v_1 \dots u_nv_n$. We define an element $\bar{e}$ as $\bar{e} = v_1 \dots u_nv_nu_1$. Thus $f(e)u_1 = u_1\bar{e}$ in R. Remark that $\bar{e}$ depends on the choice of the word $u_1v_1 \dots u_nv_n$ representing $f(e)$ in R. Notice that $\bar{e}$ and $f(e)$ are conjugate. Indeed, if $w = v_1 \dots u_nv_n$, then $f(e) = u_1w$ and $\bar{e} = wu_1$. Hence, $\bar{e}^3 = wu_1wu_1wu_1 = wf(e)^2u_1 = wf(e)u_1 = wu_1wu_1 = \bar{e}^2$. We have $\bar{e}^2 \neq 0$ since $f(e) \neq 0$ and $f(e) = f(e)^2 = f(e)^3 = u_1\bar{e}^2w$. Thus $\bar{e}^2$ is an idempotent of R of type (Q', Q').

Let $I \cap D = \{x_1, \dots, x_d\}$. Then there is a word $z \in \mathcal{A}^*$ of rank 1, a state $x \in Q$ and a path on the right Fischer cover of X labeled z from any state in $I \cap D$ to x. Moreover, there are a letter $u \in \mathcal{U}$, a state $y \in Q'$ and an edge $x \xrightarrow{u} y$ in the right Fischer cover of Z. It follows that there are paths as follows in the right Fischer cover of Z.

$$
\begin{aligned}
&x_1 \xrightarrow{u_1} y_1 \xrightarrow{w} x_1 \xrightarrow{u_1} y_1 \xrightarrow{w} x_1 \xrightarrow{f(z)} x \xrightarrow{u} y\\
&x_2 \xrightarrow{u_1} y_2 \xrightarrow{w} x_2 \xrightarrow{u_1} y_2 \xrightarrow{w} x_2 \xrightarrow{f(z)} x \xrightarrow{u} y\\
&\vdots\\
&x_d \xrightarrow{u_1} y_d \xrightarrow{w} x_d \xrightarrow{u_1} y_d \xrightarrow{w} x_d \xrightarrow{f(z)} x \xrightarrow{u} y
\end{aligned}
$$

The states y_i, for $1 \leq i \leq d$, belong to Q'. Since the states x_i are distinct, also the states y_i are distinct. Indeed, let us assume for instance that $y_1 = y_2$. Then $x_1 = x_2$ by considering the paths labeled w from y_i to x_i for $i = 1, 2$. Thus, in the right Fischer cover of Z there are the following paths, for $1 \leq i \leq d$.

$$y_i \xrightarrow{(wu_1)^2} y_i \xrightarrow{wf(z)u} y$$

Since $\bar{e} = wu_1$ and $wf(z)u$ are contained in $(g(\mathcal{B}))^*$, the elements $e' = g^{-1}(\bar{e}^2) = g^{-1}((wu_1)^2)$ and $w' = g^{-1}(wf(z)u)$ are in T. Hence the following paths are in the right Fischer cover of Y, for $1 \leq i \leq d$.

$$y_i \xrightarrow{e'} y_i \xrightarrow{w'} y$$

Notice that e' is an idempotent of T. Since the graph of the right Fischer cover of Y is strongly connected, there is a word w'' of rank 1 whose domain contains y. This allows us to assume that the word w' is an element of rank 1 of T. Hence

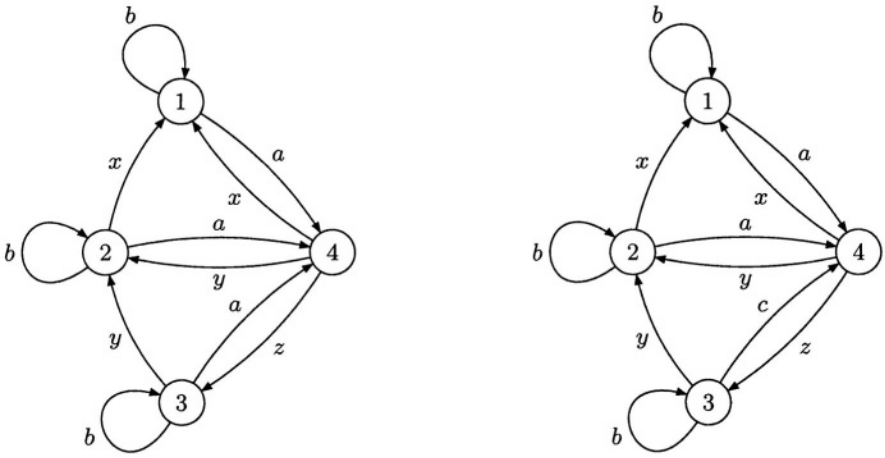


Fig. 5. The right Fischer covers of two non conjugate sofic shifts X (on the left) and Y (on the right), with $\mathcal{A} = \{a, b, x, y, z\}$ and $\mathcal{B} = \{a, b, c, x, y, z\}$. The shift X has degree 3 while the shift Y has degree 2.

the domain D' of the $\mathcal{R}$-class of w' and the image I' of the idempotent e', contain $\{y_1, \ldots, y_d\}$. We now prove that $D' \cap I'$ is exactly the set $\{y_1, \ldots, y_d\}$.

Suppose that there is a state $\bar{y} \in D' \cap I'$ such that $\bar{y} \neq y_i$ for each i. Hence the following path is in the right Fischer cover of Y.

$$\bar{y} \xrightarrow{e'} \bar{y} \xrightarrow{w'} y$$

Thus, in the right Fischer cover of Z there is the following path.

$$\bar{y} \xrightarrow{(wu_1)^2} \bar{y} \xrightarrow{wf(z)u} y$$

Let $\bar{x}$ be the final state of the path labelled by w and starting at $\bar{y}$. It follows that a path of the kind

$$\bar{y} \xrightarrow{w} \bar{x} \xrightarrow{u_1} \cdot \xrightarrow{w} \cdot \xrightarrow{u_1} \bar{y} \xrightarrow{w} \bar{x} \xrightarrow{f(z)} x$$

is in the right Fischer cover of Z (recall that $f(z)$ has rank 1). Being $\bar{x}$ in the image of $u_1 w u_1 w = f(e)^2 = f(e)$, we have that $\bar{x}$ in the image I of e. Moreover, $\bar{x}$ is in the domain of $f(z)$ and hence it is also in the domain D of z. This implies that $\bar{x}$ is one of the elements x_i and hence $\bar{y}$ is the corresponding y_i, which is a contradiction. Thus the cardinality of $D' \cap I'$ is d. □

We get the following corollary.

Corollary 1. *Let X be an irreducible sofic shift. Then its degree is invariant under conjugacy. Moreover, the increasing sequence $(d_1, d_2, \ldots, d_n)$ of positive integers such that X is d_i-non-left closing (where d_n is the degree of the shift), is invariant under conjugacy.*

We show in Figure 5 an example of two sofic shifts X and Y, where X has not almost finite type with degree 3, and Y has not almost finite type with degree 2. Thus these two shifts are not conjugate since their degrees are different. Remark that they have the same syntactic graph, which is another conjugacy invariant defined and described in [2].

There are irreducible sofic shifts with degree d for every $d > 1$. For instance, consider the right Fischer cover $\mathsf{A} = (\{1, 2, \ldots, d, d+1\}, E)$ on the alphabet $\mathcal{A} = \{a, b, c\}$, where the set of edges is $E = \{i \xrightarrow{b} i,\ i \xrightarrow{a} d+1 \mid i \neq d+1\} \bigcup \{i \xrightarrow{c} i-1 \mid 2 \leq i \leq d\} \bigcup \{d+1 \xrightarrow{c} d\}$. This right Fischer cover has degree d.

4 Links with Semigroup Theory

The above propositions have links with previous results in the theory of varieties of semigroups.

A finite biprefix code (see for instance [4]) defines an almost finite type shift in a natural way. It is known from [14] that, if X is a finite biprefix code and S the syntactic semigroup of X^+, eSe defines a semigroup of partial injective transformations for any idempotent e. Margolis [14] also showed that every semigroup of partial injective transformations divides a semigroup of partial injective transformations which is the syntactic semigroup of a finite biprefix code.

An equivalent formulation of Proposition 1 is the following.

Proposition 3. *Let X an irreducible sofic shift and S its syntactic semigroup. Then X has almost finite type if and only if for any idempotent $e \in S$, the semigroup eSe is a semigroup of partial one-to-one transformations.*

Thus, when S has almost finite type, the semigroup eSe is, for any idempotent e, a subsemigroup of an inverse semigroup. This implies that the semigroup S belongs to the variety of semigroups T such that for each idempotent e, the semigroup eTe is in the variety generated by inverse semigroups. We do not know whether this condition is sufficient to guarantee that X has almost finite type.

Acknowledgment. The authors wish to thank the reviewer who suggested the second statement of Corollary 1.

References

1. Marie-Pierre Béal. *Codage Symbolique.* Masson, 1993.
2. Marie-Pierre Béal, Francesca Fiorenzi, and Dominique Perrin. The syntactic graph of a sofic shift. In *STACS (Montpellier, 2004),* volume 2996 of *Lecture Notes in Comput. Sci.,* pages 282–293. Springer, Berlin, 2004.
3. Danièle Beauquier. Minimal automaton for a factorial transitive rational language. *Theoret. Comput. Sci.,* 67:65–73, 1989.
4. Jean Berstel and Dominique Perrin. *Theory of Codes.* Academic Press, New York, 1985.

5. F. Blanchard. Codes engendrant certains systèmes sofiques. *Theoret. Comput. Sci.,* 68(3):253–265, 1989.
6. Mike Boyle, Bruce P. Kitchens, and Brian H. Marcus. A note on minimal covers for sofic shifts. *Proc. Amer. Math. Soc.,* 95:403–411, 1985.
7. Doris Fiebig, Ulf-Rainer Fiebig, and Nataša Jonoska. Multiplicities of covers for sofic shifts. *Theoret. Comput. Sci.,* 1-2:349–375, 2001.
8. G. A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Math. Systems Theory,* 3:320–337, 1969.
9. Nataša Jonoska. A conjugacy invariant for reducible sofic shifts and its semigroup characterizations. *Israel J. Math.,* 106:221–249, 1998.
10. R. Karabed and Brian H. Marcus. Sliding-block coding for input-restricted channels. *IEEE Trans. Inform. Theory,* IT-34:2–26, 1988.
11. Bruce P. Kitchens. *Symbolic Dynamics: one-sided, two-sided and countable state Markov shifts.* Springer-Verlag, 1997.
12. Douglas A. Lind and Brian H. Marcus. *An Introduction to Symbolic Dynamics and Coding.* Cambridge, 1995.
13. Brian H. Marcus. Sofic systems and encoding data. *IEEE Trans. Inform. Theory,* IT-31:366–377, 1985.
14. Stuart W. Margolis. On the syntactic transformation semigroup of a language generated by a finite biprefix code. *Theoret. Comput. Sci.,* 21(2):225–230, 1982.
15. Masakazu Nasu. An invariant for bounded-to-one factor maps between transitive sofic subshifts. *Ergodic Theory Dynam. Systems,* 5:89–105, 1985.
16. Masakazu Nasu. Topological conjugacy for sofic systems. *Ergodic Theory Dynam. Systems,* 6(2):265–280, 1986.
17. Jean-Éric Pin. *Varieties of Formal Languages.* Plenum Publishing Corporation, New York, 1986.
18. B. Weiss. Subshifts of finite type and sofic systems. *Monats. für Math.,* 77:462–474, 1973.
19. Susan Williams. Covers of non-almost-finite-type systems. *Proc. Amer. Math. Soc.,* 104:245–252, 1988.

Membership and Reachability Problems for Row-Monomial Transformations

Alexei Lisitsa and Igor Potapov*

Department of Computer Science, University of Liverpool, Chadwick Building, Peach St, Liverpool L69 7ZF, U.K.,
{alexei,igor}@csc.liv.ac.uk

Abstract. In this paper we study the membership and vector reachability problems for labelled transition systems with row-monomial transformations. We show the decidability of these problems for row-monomial martix semigroups over rationals and extend these results to the wider class of matrix semigroups. After that we apply our methods to reachability problems for a class of transition systems which turn out to be equivalent to specific counter machines.

1 Introduction

In this paper we study the membership and vector reachability problems for labelled transition systems with row-monomial transformations.

We started our work on the membership and vector reachability problem for matrix semigroups that have natural and quite broad connections with combinatorics on words [2], the accessibility problem for linear sequential machines [13], the reachability problem in linear maps, verification of broadcasting protocols [9] etc. Then it turns out that these questions also are closely related to verification of counters automata and labelled transition systems.

The membership problem for a semigroup with only one generator ("is a matrix B a power of a matrix A") was known as the "orbit problem" and was shown to be decidable (in polynomial time) by Kannan and Lipton in 1986 [13]. The most natural generalization of the the "orbit problem" is the membership problem for matrix semigroups, given by a list of generators.

Problem 1. **The membership problem.** Let S be a given finitely generated semigroup of $n \times n$ matrices from $\mathbb{Q}^{n \times n}$. Determine whether a matrix M belongs to S. In other words, determine whether there exists a sequence of matrices $M_1, M_2, \ldots, M_k$ in S such that $M_1 \cdot M_2 \cdots M_k = M$

Paterson [15] shown that the problem is undecidable even for 3×3 integral matrices when considered a special case of the membership problem for matrix semigroups - the *mortality problem* (determination whether the zero matrix belongs to a matrix semigroup). It was shown in [4] that the mortality problem is

* Work partially supported by The Nuffield Foundation grant NAL/00684/G.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 623–634, 2004.
© Springer-Verlag Berlin Heidelberg 2004

undecidable even for a case of two generators where the dimension of the matrices is at least 45. This result was improved in [12] by reducing the dimension to 24. Also the mortality problem for a case of 2×2 matrices and the problem of checking whether the identity matrix belongs to a matrix semigroup are still open [2]. At the same time several decidable cases have been discovered. The decidability of the general membership problem for semigroups generated by *commutative matrices* [13], was shown in [3] for a case of two matrix generator and then extended in [1] for the general case of commutative matrices.

We show in this paper that the membership problem is decidable for any finitely generated row-monomial matrix semigroup over $\mathbb{Q}$. The elements of such semigroups are row-monomial matrices, i.e. matrices with exactly one non-zero rational entry in every row. Note that row-monomial matrix semigroups in general are not commutative. Row-monomial matrices appear naturally in the semigroup representations by matrices over groups with zero [8], due to the property that no *addition* operation is needed for the matrix multiplication. It is turned out also to be a crucial property that we use later for the decidability proofs.

Another generalization of the "orbit problem" is the vector reachability problem for a matrix semigroup or reachability in iterative maps.

Problem 2. **The vector reachability problem for matrix semigroups.** Let S be a given finitely generated semigroup of $n \times n$ matrices from $\mathbb{Q}^{n \times n}$ and vectors $\bar{x},\bar{y}$ from $\mathbb{Q}^n$. Decide whether there is a matrix $M \in S$ such that $M \cdot \bar{x} = \bar{y}$.

In other words it is equivalent to the following reachability problem for non-deterministic linear maps: "Given two vectors u and v in n-dimensional vector space over $\mathbb{Q}$ and a set A of linear transformations. Determine whether there exists a sequence of transformations from A such that maps v to u". In algebraic terms the vector reachability problem can be expressed as a problem of determining whether it is possible to get a vector u by an action of matrix semigroup on the initial vector v. This problem is also decidable for a case of semigroups generated by one matrix [13]. In order to show the decidablity for a case of non-deterministic linear row-monomial maps we use a technique similar to that we applied to the membership problem. We show that the vector reachability problem is decidable for any row-monomial semigroup over $\mathbb{Q}$.

An inspection of proofs about decidability results for row-monomial matrix semigroups over rationals reveals that these results can be generalized to the case of row-monomial matrix semigroups over commutative semigroup $\mathcal{S}$ satisfying some natural effectiveness conditions. As an instance of the general result we show the decidability of the membership and vector reachability problems for row-monomial matrix semigroups over $\mathcal{S}$, where $\mathcal{S}$ is an arbitrary finitely generated *commutative* matrix semigroup over an algebraic number field F.

Another natural generalization is considering an arbitrary finite graph structure, instead of a Cayley graph for semigroups, but still with labels from a semigroup $\mathcal{S}$. We show that this class of transition systems have decidable matrix and vector reachability problems, as well.

Also we found that subclass of such systems can be translated to counter machine model with guards of the form $x_i' = x_j \cdot c_{i,j}$, where $c_{i,j} \in \mathbb{Q}$, x_i' or x_i the

value, respectively, after or before the transition or equivalent counter machine with additive quards of the form $x_i' = x_j + c_{i,j}$ where $c_{i,j} \in \mathbb{Z}$. So we also have decidability of the reachability problem for these classes of counter machines.

This paper is organised as follows. Next section contains preliminaries. In the third section we give a technical background for row-monomial matrices and semigroups, that we use in Section 4 to prove the decidability results. The Section 5 contains a number of generalizations of systems with row-monomial transformations. The paper ends with some conclusions and directions for further research.

2 Preliminaries

In what follows we use traditional denotations $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ and $\mathbb{Q}^+$ for the sets of naturals (non-negative integers), integers, rationals and non-negative rationals, respectively. By $[n]$ we denote the initial segment $\{1, \ldots, n\}$ of positive integers. For any vectors $\bar{x}, \bar{y} \in \mathbb{Q}^n$ denote by $\bar{x} \cdot \bar{y}$ their component-wise multiplication, that is $(x_1, \ldots, x_n) \cdot (y_1, \ldots y_n) = (x_1 \cdot y_1, \ldots, x_n \cdot y_n)$. Further, let $\bar{x}^n = \underbrace{\bar{x} \cdot \ldots \bar{x}}_{n}$.

Semigroups. A *semigroup* is a pair $(S, \cdot)$, where S is a set and $\cdot$ is an associative binary operation on S. A semigroup $(S, \cdot)$ is generated by a set A of its elements iff every element of S is a finite product $a_{i_1} \cdot a_{i_2} \cdot \ldots \cdot a_{i_k}$ where $a_{i_j} \in A$.

Definition 1. *Let S is a semigroup generated by A. The (right) Cayley graph for a semigroup S with respect to A is a directed labelled graph defined as follows: V is the set of vertices, where $V = S$; $E = \cup_{a \in A} E_a$ is the set of labelled arcs, where $E_a = \{\langle s, sa \rangle | s \in S, a \in A\}$ is a set of arcs labelled by a. The left Cayley graph is defined analogously, with $E_a = \{\langle s, a \cdot s \rangle | s \in S, a \in A\}$*

Multiplicative subgroups of $\mathbb{Q}^+$ and additive groups $\mathbb{Z}^r$. Let P be a finite set of prime numbers $\{p_1, \ldots, p_r\}$. Denote by Q_P multiplicative group of rationals, generated by P, that is the set $\{p_1^{m_1} \cdot \ldots \cdot p_1^{m_r} \mid m_1, \ldots, m_r \in \mathbb{Z}\}$ together with multiplication as the group operation. Notice that Q_P is a finitely generated subgroup of $(\mathbb{Q}^+, \cdot)$.

It follows from the properties of multiplication, in particular from fundamental theorem of arithmetics that the mapping $\rho_P : \mathbb{Q}_P \to \mathbb{Z}^r$, $\rho(p_1^{m_1} \cdot \ldots \cdot p^{m_r}) = (m_1, \ldots m_r)$ is isomorphism between $(\mathbb{Q}_P, \cdot)$ and $(\mathbb{Z}_r, +)$.

Let $\mathbb{Z}_2$ be two-elements cyclic group $(\{0, 1\}, +^{\mathrm{mod}\ 2})$. The mapping $sign : \mathbb{Q} - \{0\} \to \mathbb{Z}_2$, $sign(q) = 0$ if $q > 0$ and $sign(q) = 1$ if $q < 0$ is homomorphism from $(\mathbb{Q} - \{0\}, \cdot)$ to $\mathbb{Z}_2$.

Linear and semilinear sets. A subset S of $\mathbb{N}^k$ is a *linear set* iff there exists vectors $v_0, v_1, \cdots, v_t$ in $\mathbb{N}^k$ such that $S = \{v | v = v_0 + a_1 v_1 + \cdots + a_t v_t, a_i \in \mathbb{N}\}$. The vectors v_0 (referred to as the constant vector) and $v_1, \cdots, v_t$ (referred to as the periods) are called the generators of the linear set S. The set $S \subseteq \mathbb{N}^k$ is semilinear if it is a finite union of linear sets.

Theorem 1. *([11]) The problem of emptiness of the intersection of two effectively definable semilinear sets is decidable.*

Parikh maps. Let $\Sigma = \{a_1, a_2, ..., a_n\}$ be an alphabet. For each word w in Σ^*, define the *Parikh map* of w to be $\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$. For a language $L \subseteq \Sigma^*$, the *Parikh map* of L is $\psi(L) = \{\psi(w) | w \in L\}$. The language L is semilinear if $\psi(L)$ is a semilinear set. The following theorem gives an effective characterization of the Parikh maps of *regular* languages.

Theorem 2. *[14] Let F be a one-way nondeterministic finite automaton. Then the Parikh map of the regular language L(F) of words accepted by F is a semilinear set effectively computable from F.*

3 Row-Monomial Matrices and Semigroups

In this section we give a technical background for row-monomial matrices and semigroups, that we use to prove the decidability results.

Definition 2. *A matrix $M \in \mathbb{Q}^{n\times n}$ is said to be monomial if it has exactly one non-zero entry in each row and in each column. If the matrix has exactly one no-zero entry in each row it is said to be row-monomial.*

Simple check shows that if $M_1, M_2 \in \mathbb{Q}^{n\times n}$ are row-monomial then $M_1 \times M_2$ is also row-monomial and thus all such matrices form a semigroup with matrix multiplication as its binary operation. Denote by S^n_{row} a semigroup of all row-monomial matrices from $\mathbb{Q}^{n\times n}$.

Definition 3. *A matrix semigroup of the dimension n is called row-monomial iff it is subsemigroup of S^n_{row}.*

With any row-monomial matrix $M = (a_{ij}) \in S^n_{row}$ we associate two mappings: *Position mapping* $p_M : [n] \to [n]$ defined as $p_M(i) = j$ iff $a_{ij} \neq 0$, and *Value mapping* $v_M : [n] \to \mathbb{Q}$ defined as $v_M(i) = a_{i\,p_M(i)}$, i.e. $v_M(i) = a_{ij}$ where $a_{ij} \neq 0$ Note that any row-monomial matrix M is determined uniquely by pair $\langle p_M, v_M\rangle$.

Let T_n be a transformation semigroup over finite set $[n] = \{1, \ldots, n\}$, that is the set of all functions from $[n]$ to $[n]$ with a binary operation $*$ defined via composition of functions: $f * g = g \circ f$, i.e. $(f * g)(i) = g(f(i))$.

Proposition 1. *The mapping $\tau : M \mapsto p_M$ is a homomorphism from S^n_{row} to T_n*

Proposition 2. *For row-monomial matrices M_1 and M_2 we have $v_{M_1\times M_2} = v_{M_2}(p_{M_1}) \cdot v_{M_1}$*

The row-monomial matrices (and corresponding linear mappings) and their multiplication (composition) have a natural graph representation. Although it is not used in the proofs, it is placed here to reveal an intuition of the "dynamics", which is hidden behind the matrix multiplication.

Definition 4. *For a row-monomial matrix $M = (a_{ij}) \in \mathbb{Q}^{n\times n}$ define its graph representation as a labelled bipartite graph g_M with the set of vertices $I \cup O$, where $I = \{i_1, \ldots, i_n\}$ are* input *vertices and $O = \{o_1, \ldots, o_n\}$ are* output *vertices. The vertices i_k and o_l are connected by an arc labelled by a rational m iff $a_{lk} = m \neq 0$.*

3.1 Graphs Associated with Row-Monomial Semigroups

Let S be row-monomial semigroup of the dimension n generated by set $A = \{M_1, \ldots, M_k\}$. We associate with S a directed labelled graph $G_S = (V, E)$ as follows: $V = \{p_M \mid M \in S\} \cup \{e\}$ is the set of vertices, where $e : [n] \to [n]$ is an identity mapping; $E = \cup_{M \in A} E_M$ is the set of arcs, where $E_M = \{\langle s, s * p_M \rangle \mid s \in V, M \in A\}$ is the set of arcs labelled by M. We call such labels *primary*.

Note 1. The graph G_S is very closely related to the right Cayley graph of $\tau(S) \cup e$ (see Proposition 1 for definition of τ) and has the same vertices. The difference is that the edges are labelled by the elements of $A \subset S$ and not by elements of $\tau(S) \cup e$. Thus, G_S represents the action of S on $\tau(S) \cup e$.

Extended labelled graph EG_S of a row-monomial semigroup S is an extension of G_S with the *secondary,* or *arithmetical* labels, which are n-tuples of rationals, i.e. the elements of $\mathbb{Q}^n$. For an arc $\langle s, s * p_M \rangle$ with the primary label M define its secondary label $l \in \mathbb{Q}^n$ as follows: $l = \langle v_M(s(1)), \ldots v_M(s(n)) \rangle$. Remember, that v_M is the value mapping of M and s is a function from $[n]$ to $[n]$.

Example 1. The figure 1 shows G_S and EG_S for the row-monomial semigroup generated by matrices $M_1 = \begin{pmatrix} a & 0 & 0 \\ b & 0 & 0 \\ 0 & 0 & c \end{pmatrix}$ and $M_2 = \begin{pmatrix} 0 & d & 0 \\ 0 & e & 0 \\ 0 & 0 & f \end{pmatrix}$.

Proposition 3. *Let S be a row-monomial semigroup generated by A. Then for any vertex s of the graph G_S and any sequence $D = M_{i_1}, \ldots M_{i_j}$ of elements from A there is a path in G_S outgoing from a vertex s with a sequence of primary labels D.*

Proof. By definition of G_S for every vertex s and label $M \in A$ there is an arc outgoing from s and labelled by M □.

Now we have a key lemma connecting the properties of pathes in EG_S with the properties of multiplication in S.

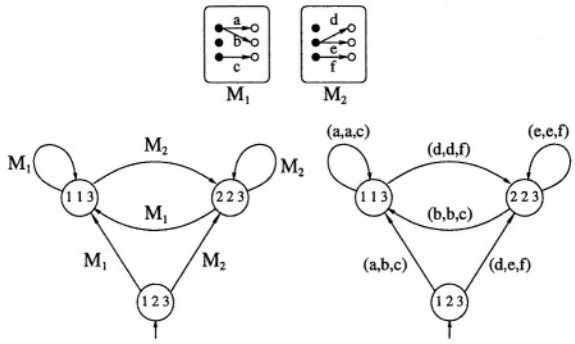


Fig. 1. Graphs associated with a row-monomial semigroup: G_S (left) and EG_S (right, primary labelling omitted)

Lemma 1. *For a row-monomial semigroup S let $M_{i_1}, \ldots, M_{i_j}$ be the sequence of primary labels, and $l_1 = \langle l_{11}, \ldots, l_{1n}\rangle, \ldots, l_j = \langle l_{j1}, \ldots, l_{jn}\rangle$ be the sequence of secondary labels along a path (of non-zero finite length) from e to some vertex s in the graph EG_S. Then we have $p_{M_{i_1}\times\ldots\times M_{i_j}} = s$ and for any $t \in [n]$ $v_{M_{i_1}\times\ldots\times M_{i_j}}(t) = \prod\limits_{1\le k\le j} l_{kt}$.*

Proof. Easy induction on the length of a path based on two observations: for an arc with the label M going from v to v' we have $v' = v * p_M = p_M \circ v$ (definition of G_S), and $p_{M_{i_1}\times\ldots\times M_{i_j}} = p_{M_{i_1}} * \ldots * p_{M_{i_j}} = p_{M_{i_j}} \circ \ldots \circ p_{M_{i_1}}$ (corollary of Proposition 1). The proof proceeds by induction on the length of a path. Let us fix some $t \in [n]$. For a path of length 1 having a primary label M_{i_1}, which starts from e we have a secondary label (by definition of secondary labels) $l_1 = \langle v_{M_{i_1}}(e(1)), \ldots, v_{M_{i_1}}(e(n))\rangle = \langle v_{M_{i_1}}(1), \ldots, v_{M_{i_1}}(n)\rangle$. It follows $v_{M_{i_1}}(t) = l_{1t} = \prod\limits_{1\le k\le 1} l_{kt}$. Consider now a path $s_0 = e \xrightarrow{M_{i_1}} s_1 \xrightarrow{M_{i_2}} \ldots \xrightarrow{M_{i_{j-1}}} s_{j-1} \xrightarrow{M_{i_j}} s_j$ of length $j > 1$. Then we have $v_{M_{i_1}\times\ldots\times M_{i_j}}(t) = = v_{M_{i_j}}(p_{M_{i_1}\times\ldots\times M_{i_{j-1}}}(t)) \cdot v_{M_{i_1}\times\ldots\times M_{i_{j-1}}}(t)$ (by Proposition 2) $= v_{M_{i_j}}(s_{j-1}(t)) \cdot v_{M_{i_1}\times\ldots\times M_{i_{j-1}}}(t)$ (by part 1 of the Lemma) $= l_{jt} \cdot v_{M_{i_1}\times\ldots\times M_{i_{j-1}}}(t)$ (by definition of secondary labels) $= l_{jt} \cdot \prod\limits_{1\le k\le j-1} l_{kt}$ (by induction assumption) $= \prod\limits_{1\le k\le j} l_{kt}$ □.

4 Decidability of the Membership and the Vector Reachability Problems

In this section we show that the membership and reachability problems are decidable for row-monomial matrix semigroups over $\mathbb{Q}$.

Theorem 3. *Any finitely generated row-monomial matrix semigroup over $\mathbb{Q}^+$ has decidable membership problem.*

Proof. Let S be a row-monomial semigroup generated by finite set of matrices of the dimension n $A = \{M_1, \ldots M_k\}$. Given an arbitrary row-monomial matrix M we have to decide whether $M \in S$, that is whether there is a sequence $M_{i_1}, \ldots, M_{i_j}$ of elements of A such that $M = M_{i_1} \times \ldots \times M_{i_j}$. By Lemma 1 and Proposition 3 this problem is equivalent to the following problem in terms of a graph EG_S:

Problem 3. **Given:** A matrix M. **Question:** Is there any path $s_0 = e \xrightarrow{l_1} s_1 \xrightarrow{l_2} \ldots \xrightarrow{l_{j-1}} s_{j-1} \xrightarrow{l_j} s_j = s$ in EG_S, with secondary labels $l_k = \langle l_{k1}, \ldots l_{kn}\rangle$, $k = 1, \ldots j$, such that

- $p_M = s$ *(Condition 1)*;
- $v_M(t) = \prod\limits_{1\le k\le j} l_{kt}$ for all $t \in [n]$ *(Condition 2)*.

Notice that for any $M \in S^n_{row}$ there is at most one vertex s in EG_S such that $s = p_M$. Since EG_S has only finitely many vertices, obtained constructively from

A, there is a simple decision procedure which determines whether such s exists. If there is no such a vertex then answer to the membership problem is negative: $M \notin S$. Otherwise, we have to check further whether there is a path from e to such s which satisfies the arithmetical constraints of the Condition 2.

Let $L = \{L_1 = \langle L_{11}, \ldots, L_{1n}\rangle \ldots L_m = \langle L_{m1}, \ldots, L_{mn}\rangle\}$ be the *finite* set of *all* secondary labels in EG_S. Let $x_1, \ldots, x_m$ be the multiplicities of labels $L_1, \ldots, L_m$, respectively, on some path from e to $s = P_M$. Then that path satisfies the Condition 2 if and only if these multiplicities satisfy the following equation:

$$L_1^{x_1} \cdot \ldots \cdot L_m^{x_m} = v_M \tag{1}$$

which is compact vector form of the following system of equations:

$$\begin{cases} (L_{11})^{x_1} \cdot (L_{21})^{x_2} \cdot \ldots \cdot (L_{m1})^{x_m} = v_M(1) \\ \vdots \\ (L_{1n})^{x_1} \cdot (L_{2n})^{x_2} \cdot \ldots \cdot (L_{mn})^{x_m} = v_M(n) \end{cases} \tag{2}$$

We transform now every *equation* of the system (2) into the equivalent *system* of linear Diophantine equations by using isomorphism ρ defined in Subsection 2. Let $P = \{p_1, \ldots, p_r\}$ be the set of all prime numbers in the factorizations of all rationals L_{ij} and $v_M(i)$ in (2). Then we replace every equation

$$(L_{1i})^{x_1} \cdot (L_{2i})^{x_2} \cdot \ldots \cdot (L_{mi})^{x_m} = v_M(i)$$

of the system (2) with the equivalent linear equation

$$x_1 \cdot \rho_P(L_{1i}) + x_2 \cdot \rho_P(L_{2i}) + \ldots + x_k \cdot \rho_P(L_{mi}) = \rho_P(v_M(i)) \tag{3}$$

which is a vector representation of a Diophantine system of linear equations:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m = b_1 \\ \vdots \\ a_{r1}x_1 + a_{r2}x_2 + \ldots + a_{rm}x_m = b_r \end{cases} \tag{4}$$

where $\langle a_{1j}, a_{2j}, \ldots a_{rj}\rangle = \rho_P(L_{ji})$ and $b_i = \rho_P(v_M(i))$.

When all equations in (2) are replaced we get a system of Diophantine $r \cdot n$ linear equations with integer coefficients in m unknowns $x_1, \ldots, x_m$ which is equivalent to (2) and (1).

Theorem 4. *(see e.g. [7,6]) The problem whether a Diophantine system of linear equations has a solution, is decidable and all its solutions form an effectively definable semilinear set.*

Then, by Theorem 4 a set of solutions for the equation (1) can be represented by an effectively computable semilinear set which we denote by $\mathcal{S}^{ar}$.

Further, all words in the alphabet L which one can read along the paths from e to s form a regular language which we denote by $\mathcal{L}_s$. By Theorem 2 Parikh map $\psi(\mathcal{L}_s)$ of $\mathcal{L}_s$ is effectively computable semilinear set and we have

by that an effective characterization of possible multiplicities of labels on all possible paths form e to s. It follows that there is a path from e to s satisfying Condition 2 iff $\mathcal{S}^{ar} \cap \psi(\mathcal{L}_s) \neq \emptyset$.

Since the problem of emptiness of intersection of two effectively definable semilinear sets is decidable (by Theorem 1), the problem of existence of such a path is decidable and, therefore, we get decidability of membership problem for finitely generated row-monomial semigroups over $\mathbb{Q}^+$. □

Now we extend the result of Theorem 3 to the general case of row-monomial matrices over $\mathbb{Q}$.

Theorem 5. *Any finitely generated row-monomial matrix semigroup over* $\mathbb{Q}$ *has decidable membership problem.*

Proof. Let S be row-monomial semigroup of the dimension n over $\mathbb{Q}$, generated by finite set of matrices $A = \{M_1, \ldots M_k\}$. We need to construct an extended version of EG_S which incorporates the information on algebraic signs. The signs of rational values will be represented by the elements of the group $\underbrace{\mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2}_{n} = \mathbb{Z}_2^n$. Now we construct the graph $EG_S^* = (V, E)$ containing all information on how S acts on $(\tau(S) \cup e) \times \mathbb{Z}_2^n$.

- The set of vertices $V = \{\langle p_M, b\rangle \mid M \in S, b \in \mathbb{Z}_2^n\} \cup \{\langle e, \bar{0}\rangle\}$, where $e : [n] \to [n]$ is an identity mapping and $\bar{0}$ is the zero element in the group $\mathbb{Z}_2^n$;
- The set of arcs $E = \cup_{M \in A} E_M$, where $E_M = \{\langle (s, b)(s * p_M, b + sign(l))\rangle \mid (s, b) \in V, M \in A$ and l is the secondary (arithmetical) label of $\langle s, s * p_M\rangle$ in $EG_S\}$ is the set of arcs labelled by M. Here $sign(l)$ is a homomorphic image of $l \in \mathbb{Q}^n$ in $\mathbb{Z}_2^n$.

Similarly to the proof of Theorem 3, the question whether some matrix M belong to the finitely generated row-monomial semigroup S is equivalent now to the following problem of existence of suitable paths in the graph EG_G^*.

Problem 4. **Given:** A matrix M. **Question:** Is there any path $s_0 = (e, \bar{0}) \xrightarrow{l_1} s_1 \xrightarrow{l_2} \ldots \xrightarrow{l_{j-1}} s_{j-1} \xrightarrow{l_j} s_j = s$ in EG_S^*, with secondary labels $l_k = \langle l_{k1}, \ldots l_{kn}\rangle$, $k = 1, \ldots j$, such that

- $(p_M, (sign(v_M(1)), \ldots, sign(v_M(n)))) = s$ *(Condition 1')*;
- $|v_M(t)| = |\prod_{1 \leq k \leq j} l_{kt}|$ for all $t \in [n]$ *(Condition 2')*.

The rest of the proof repeats all arguments of the proof of Theorem 3. □.

Theorem 6. *Any finitely generated row-monomial matrix semigroup over* $\mathbb{Q}$ *has decidable vector reachability problem.*

Proof. Fix some finitely generated row-monomial semigroup S of the dimension n. Reduce the vector reachability problem for S to the finite set of problems, whose questions are "Is there any matrix $M \in S$ with a given p_M such that $\bar{y} = M \cdot \bar{x}$?". Notice that there is only finitely many different p_M for the elements of semigroup S.

Further, for $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{y} = (y_1, \ldots, y_n) = M \cdot \bar{x}$ we have $y_i = v_M(i) \cdot x_{p_M(i)}$, $i = 1 \ldots n$. Thus, for given $\bar{x}, \bar{y}$ and fixed p_M we get the system *SE* of linear equations with n unknowns $v_M(i)$. If such a system is inconsistent (e.g. if $y_i \neq 0$ and $x_{p_M(i)} = 0$) then there is no matrix M with position mapping p_M satisfying $\bar{y} = M \cdot \bar{x}$. If the system is consistent, then we reduce our problem to the question on whether the semigroup S contains a matrix M with given p_M and v_M (solution the of *SE*). If $v_M(i)$ is defined for all i then by Theorem 5 this problem is decidable. If some $v_M(i)$ are undefined (e.g. if $y_i = 0$ and $x_{p_M(i)} = 0$) then we reduce the problem to the finitely many modified variants of Problem 4 from the proof of Theorem 5 as follows. Consider all possible variants of Condition 1' which are consistent with the obtained solution $(v_M(1), \ldots, v_M(n))$ of the system *SE* and with Condition 2' applied only to those t for which $v_M(t)$ is defined. It is easy to see that initial reachability problem has positive answer iff at least one of these variants has a positive solution. □

If all matrices $\{M_1, \ldots M_k\}$ in the generating set of some row-monomial semigroup S are in fact *monomial,* then one may consider group structure on the elements of *S.* The following theorem is an easy corollary of Theorems 5 and 6.

Theorem 7. *Let S be a matrix group generated by a finite set A of monomial matrices. Then the membership and reachability problems are decidable for S.*

5 More General Result for Matrix Semigroups

An inspection of the proofs of Theorems 3 and 6 reveals that only a few properties of rational numbers were actually used. Because of that one can generalize these theorems to the following

Theorem 8. *Let $\mathcal{S}$ be any countable commutative semigroup with the following properties: semigroup operation (multiplication) is effectively computable and the set of positive integer solutions of an equation $a_1^{x_1} \cdot \ldots \cdot a_k^{x_k} = a_t$ with $a_1, \ldots, a_k, a_t \in \mathcal{S}$ is effectively computable from the equation semilinear set. Then any finitely-generated row-monomial matrix semigroup over $\mathcal{S}$ has decidable membership and vector reachability problems.*

Proof. The proof repeats the proofs of Theorem 3 and 6, except two parts. First we assume everywhere the elements of $\mathcal{S}$ instead of elements of $\mathbb{Q}^+$ or $\mathbb{Q}$. Second we go directly from the equation (1) to its set of solutions $\mathcal{S}^{ar}$ using the second assumption of the theorem. We again use the fact that the intersection of two semilinear sets is effectively constructible semilinear set and the emptiness problem for a semilinear set is decidable [11]. Finally note that the requirement of commutativity is essential, because otherwise one can not reduce Condition 2 to the equation (1) (see the proof of Theorem 3). □

As an instance of the general result consider the case when $\mathcal{S}$ is an arbitrary finitely generated *commutative* matrix semigroup of the dimension n over an algebraic number field *F.* It has been shown in [1] that for any multiplicative

equation of the above form, the set of its solutions is an effectively computable semilinear set. Then from Theorem 8 we have

Proposition 4. *Let $\mathcal{K} \subseteq (F^{n\times n})^{m\times m}$ be a finitely generated row-monomial matrix semigroup of the dimension m over $\mathcal{S}$. Then K has decidable membership and vector reachability problems.*

Notice, that row-monomial $m \times m$-matrices over $\mathcal{S}$, i.e. block matrices with the entries being themselves $n \times n$-matrices (elements of $\mathcal{S}$) are *not necessarily* row-monomial if considered as *$(n \cdot m \times n \cdot m)$-matrices over F.*

As an interesting application of Proposition 4 consider the membership and vector reachability problems for matrices over complex numbers. It is well-known that the multiplicative (semi-)group of complex numbers is isomorphic to a multiplicative (semi-)group of two-dimensional matrices with the mapping $a + bi \mapsto \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$ being an isomorphism. The proposition 4 implies now

Proposition 5. *Any finitely generated row-monomial matrix semigroup over* rational complex *numbers has decidable membrship and vector reachability problems.*

The row-monomial transformations on counter machines. Now we consider another generalization of the decision algorithms which are represented in Sections 4 and 5. This generalization is based on the idea of changing structural constraints of matrix semigroups towards the analysis of row-monomial transformations on arbitrary graph structure. The class of graphs associated with matrix semigroups (Subsection 3.1) is quite limited. The natural extension of such matrix multiplication system is an arbitrary labelled directed graph G where each label is a row-monomial matrix over $\mathcal{S}$. Similarly to the membership problem in matrix semigroups for a matrix M we can define the "matrix reachability problem" in the more general system with an arbitrary labelled directed graph.

Problem 5. **The matrix reachability problem.** Given a row-monomial matrix M over $\mathcal{S}$ and an arbitrary labelled directed finite graph G where each label is a row-monomial matrix over $\mathcal{S}$. Determine whether there exists a sequence of labels $M_1, M_2, \ldots, M_k$ on the finite connecting path in a graph G starting from an initial node, such that $M_1 \times M_2 \times \ldots \times M_k = M$.

Theorem 9. *Let G be an arbitrary directed graph labelled by row-monomial matrices with elements from a commutative semigroup $\mathcal{S}$ satisfying the conditions from Theorem 8. The matrix reachability and vector reachability problems for G are decidable.*

It is not difficult to translate the model of directed graph labelled by row-monomial matrices over $\mathbb{Q}$ to closely related model of counter machine in which a transition can be fired only if the values of counters satisfy some guards [5]. Actually the "row-monomial transformations" on counters correspond to the guards of the form $x_i' = x_j \cdot c_{i,j}$, where x_i' or x_i the value of counter i respectively after or

before the transition. According to isomorphism ρ between $(Q_P, \cdot)$ and $(Z_r, +)$ (defined in Subsection 2) we can convert each guard of the form $x_i' = x_j \cdot c_{i,j}$ to the set of guards of the form $x_i' = x_j + c_{i,j}$ and getting to an equivalent additive system. This translation shows that the vector reachability problem for a graph G can be formulated as a reachability problem for a class of counter machine. The reachability problem for multi counter machine is defined as follows: "Given initial state and set of counter values $C_{initial}$ for a counter machine A. Determine whether there is a reachable final state with the set of counter values C_{final}".

Proposition 6. *Let A be a counter machine*

- **i.** *The reachability problem for a counter machine with guards of the form* $x_i' = x_j \cdot c_{i,j}$, *where* $c_{i,j} \in \mathbb{Q}$ *is decidable.*
- **ii.** *The reachability problem for a counter machine with guards of the form* $x_i' = x_j + c_{i,j}$, *where* $c_{i,j} \in \mathbb{Z}$ *is decidable.*

The decidability results from Theorem 9 and Proposition 6 follow from the existence of decision algorithms for matrix semigroups. The only difference is that the new algorithms should take into account the structure of the graph. But this can be incorporated into existing methods in a straightforward way.

Actually the set of reachable states for counter machines from the Proposition 6.ii is definable in Presburger arithmetics. This fact can be extracted from Theorem 4 of Finkel and Leroux in [10]. So the reachability for such class of machines can be determined by checking the formulae in Presburger arithmetics In contrast to their solution we have used more specific methods.

Column-monomial case. The class of column-monomial matrices is a natural counterpart of the class of row-monomial matrices. The matrix transpose A^T of any row-monomial matrix A is a column-monomial and vice versa. All results on decidability of *membership* and *matrix reachability problems* for row-monomial matrices can be easily carried out to the case of column-monomial matrices using classical identity $(A \times B)^T = B^T \times A^T$.

As to the vector reachability problems for column-monomial matrices the situation is more complicated. In order to compute the result of application of a column-monomial linear mapping to a vector, one needs *both multiplication and addition* operations, unlike the row-monomial case, where only multiplication is needed. One may show that vector reachability problems may be reduced in that case to the *parameterized versions* of membership (or, more general, matrix reachability) problems, where a target matrix may include indeterminate entries. We leave the investigation of related decidability and algorithmic issues for the further work.

6 Conclusion

We have shown in this paper that the restriction to the row-monomial matrices makes many algorithmic problems for matrices decidable (unlike the general case). We started with showing decidability of membership and vector reachability problems for row-monomial martix semigroups over rationals. We then

carried out these results to the wider class of semigroups of matrices with entries being from any semigroup satisfying some natural effectiveness conditions. Thereby we extend the class of (non-commutative, in general) matrix semigroups where the membership problem is known to be decidable.

Then we have shown that the proposed methods may be extended to show the decidability of more general matrix reachability problems, which in fact are equivalent to the reachability problems for specific counter automata. It is turned out that we get some already known results about automata with integer counters, but using a different and more specific methods.

There are several directions for further work. First, to investigate to what extent the proposed methods can be used to demonstrate decidability of other, not necessarily row-monomial classes of matrix semigroups and related vector reachability problems. Second, to investigate the vector reachability problems for column-monomial matrices. Third, taking counter automata perspective, to investigate whether one can handle more general guards in such automata using the methods we have presented here.

References

1. L. Babai, R. Beals, J. Cai, G. Ivanyos, E. M. Luks Multiplicative equations over commuting matrices. ACM-SIAM symposium on Discrete algorithms, (1996) 28-30
2. J.Bestel and Karhumaki. Combinatorics on Words - A Tutorial, Bulletin of the EATCS, February (2003) 178 - 228
3. Jin-yi Cai, Richard J. Lipton, and Yechezkel Zalcstein. The complexity of the membership problem for 2-generated commutative semigroups of rational matrices. In 35th Annual Symposium on Foundations of Computer Science (1994) 135-142
4. J. Cassaigne and J. Karhumaki. Examples of undecidable problems for 2-generator matrix semigroups. Theoretical Computer Science, 204(1), (1998) 29-34
5. H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. CAV'98, LNCS 1427, (1998) 268-279
6. E. Contejean and H. Devie. An efficient algorithm for solving systems of Diophantine equations. Information and Computation, 113(1), (1994) 143-172
7. E. Contekean, F.Ajil. Complete Solving of Linear Diophantine Equations and Inequations without Adding Variables. CP95, LNCS 976, (1995) 1-17
8. A.H. Clifford and G.B. Preston. The algebraic theory of semigroups, Volume I, American Mathematical Society, Providence, Rhode Island, (1961)
9. J. Esparza, A.Finkel and R. Mayr. On the Verification of Broadacst Protocols. In Proc. LICS'99 (1999) 352-359
10. A. Finkel and J. Leroux. How to compose Presburger-accelerations: Applications to broadcast protocols. FSTTCS02, LNCS 2556, (2002) 145-156
11. S. Ginsburg. The Mathematical Theory of Context-Free Languages. McGraw-Hill, New York, (1966)
12. V.Halava, T.Harju. Mortality in Matrix Semigroups, American Mathematical Monthly, Vol 108 No. 7, (2001) 649-653
13. R. Kannan and R. J. Lipton. Polynomial-time algorithm for the orbit problem, Journal of the ACM (JACM), Volume 33 , Issue 4, (1986) 808 - 821
14. R. Parikh. On context-free languages, J. ACM, 13, (1966), 570-581
15. M. Paterson. Unsolvability in 3×3 matrices. Studies in Applied Mathematics, 49 (1970), 105-107

On Pseudovarieties of Semiring Homomorphisms

Libor Polák*

Department of Mathematics, Masaryk University
Janáčkovo nám 2a, 662 95 Brno, Czech Republic
polak@math.muni.cz

Abstract. We formulate and prove here an Eilenberg- and Reiterman-type theorems for pseudovarieties of idempotent semiring homomorphisms. We also initiate the study of quite significant classes of languages – the so-called multiliteral varieties of regular languages. Finally we show how other Eilenberg-type theorems follow from our considerations.

Keywords: Varieties of languages, pseudovarieties of homomorphisms

MSC 2000 Classification: 68Q45 Formal languages and automata

1 Introduction

The usage of (ordered) syntactic monoids is a well-established tool of the algebraic theory of regular languages – see Pin [3]. In [4] the author introduced the so-called syntactic semiring of a language under the name syntactic semilattice-ordered monoid. The main result of that paper is an Eilenberg-type theorem giving a one-to-one correspondence between the so-called conjunctive varieties of regular languages and pseudovarieties of idempotent semirings. At that time we did not have any significant examples from the language theory point of view. The latest version of [4] presents a characterization of the complements of +-languages formed by all $XA^* \cup A^*Y \cup Z$ where $X, Y, Z \subseteq A^+$ are finite.

The next author's contribution [5] studies the relationship between the syntactic monoids and syntactic semirings. In the meantime the syntactic semirings showed to be useful when considering language equations [6] and the so-called universal automaton of a given language [7]. Recently Straubing [8] introduced the so-called pseudovarieties of monoid homomorphisms. This research together with an equational characterization of those classes by Kunc [2] led us to the present investigations.

We recall preliminaries in the next section. Section 3 is devoted to an Eilenberg-type theorem for pseudovarieties of idempotent semiring homomorphisms. The next section deals with a modification of the approach of [2] for semirings; in fact we succeeded to simplify the proofs. In Section 5 we present several important examples of so-called multiliteral varieties of languages and we start to characterize some of them equationally. In the last section we show how to derive the previous Eilenberg-type theorems from our results.

* The author acknowledges the support of the Grant no. 201/01/0323 of the Grant Agency of the Czech Republic.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 635–647, 2004.
© Springer-Verlag Berlin Heidelberg 2004

2 Preliminaries

For a finite alphabet A we denote A^+ and $A^* = A^+ \cup \{1\}$ the free semigroup and the free monoid over A. The length of $u \in A^*$ is denoted by $|u|$ and $|A|$ is the cardinality of A. For a mapping $\xi : X \to Y$ we put
$\ker \xi = \{ (x, x') \in X \times X \mid \xi(x) = \xi(x') \}$ and $\operatorname{im} \xi = \{ \xi(x) \mid x \in X \}$.

An *idempotent semiring* is a structure $(S, \cdot, \vee)$ where
$(S, \cdot)$ is a monoid with the neutral element 1,
$(S, \vee)$ is a semilattice with the smallest element 0,
$(\forall a, b, c \in S)(a(b \vee c) = ab \vee ac$ and $(a \vee b)c = ac \vee bc)$,
and $(\forall a \in S)\ a0 = 0a = 0$.
Such a structure becomes an ordered monoid with respect to the relation $\leq$ defined by $a \leq b \iff a \vee b = b$, $a, b \in S$. An *ideal* J of an idempotent semiring S is a subset of S such that $a \in J$, $b \in S$, $b \leq a$ implies $b \in J$, and $a, b \in J$ implies $a \vee b \in J$. The ideal of S generated by a set $T \subseteq S$ is denoted by $(T]$. In a finite idempotent semiring S it is possible to define for each $a \in S$ the element $a^* = 1 \vee a \vee a^2 \vee \ldots$ (the join is finite). A *homomorphism* $\sigma : S \to T$ should satisfy $\sigma(1) = 1$, $\sigma(0) = 0$.

Let $\mathcal{S}$ be the class of all finite idempotent semirings. Let $A^\square$ denote the set of all finite subsets of A^*. Note that this set with the operations $U \cdot V = \{ uv \mid u \in U, v \in V \}$ and the usual union form a free idempotent semiring over the set A. Consequently, a homomorphism $\phi : A^\square \to S$ into $S \in \mathcal{S}$ is given by arbitrary assignments $\phi(\{a\}) = s$ $(a \in A, s \in S)$. Notice that $\emptyset^\square = \{\emptyset, \{1\}\}$ and that one can identify the elememts of $A^\square$ with polynomials over A with coefficients in the Boolean semiring.

Let $\mathfrak{S} = \{ \phi : A^\square \twoheadrightarrow S \mid A$ is a finite set and $S \in \mathcal{S} \}$ be the class of all surjective semiring homomorphisms from a finitely generated free idempotent semiring onto a finite idempotent semiring. We consider a category $\mathbb{D}$ of homomorphisms between finitely generated free idempotent semirings, i.e. objects are all semirings $A^\square$ where A is a finite set and the sets of morphisms $\mathbb{D}(B^\square, A^\square)$ consist of certain semiring homomorphisms from $B^\square$ to $A^\square$.

Basic examples are the category $\mathbb{D}_{all}$ of all semiring homomorphisms, and the categories $\mathbb{D}_{mi}$, $\mathbb{D}_{ml}$ and $\mathbb{D}_l$ of all *monoid induced, multi-literal* and *literal* homomorphisms, respectively. For finite alphabets A, B :
$f \in \mathbb{D}_{mi}(B^\square, A^\square)$ iff for each $b \in B$ there is $u \in A^*$ such that $f(\{b\}) = \{u\}$,
$f \in \mathbb{D}_{ml}(B^\square, A^\square)$ iff for each $b \in B$ there are $a_1, \ldots, a_k \in A$ such that $f(\{b\}) = \{a_1, \ldots, a_k\}$, and
$f \in \mathbb{D}_l(B^\square, A^\square)$ iff for each $b \in B$ there is $a \in A$ such that $f(\{b\}) = \{a\}$.

A class $\mathfrak{X} \subseteq \mathfrak{S}$ is a $\mathbb{D}$-*pseudovariety of semiring homomorphisms* if it satisfies :

(H) for each $(\phi : A^\square \twoheadrightarrow S) \in \mathfrak{X}$ and a surjective semiring homomorphism $\sigma : S \twoheadrightarrow T$ we have $\sigma\phi \in \mathfrak{X}$,

($S_{\mathbb{D}}$) for each $f \in \mathbb{D}(B^\square, A^\square)$ and $(\phi : A^\square \twoheadrightarrow S) \in \mathfrak{X}$ we have $(\phi f : B^\square \twoheadrightarrow \operatorname{im}(\phi f)) \in \mathfrak{X}$,

(P) for each non-negative integer m and for each system $\phi_1 : A^\square \twoheadrightarrow S_1, \ldots, \phi_m : A^\square \twoheadrightarrow S_m \in \mathfrak{X}$ we have $((\phi_1, \ldots, \phi_m) : A^\square \twoheadrightarrow \operatorname{im}(\phi_1, \ldots, \phi_m)) \in \mathfrak{X}$ (here

$(\phi_1, \ldots, \phi_m)(U) = (\phi_1(U), \ldots, \phi_m(U)) \in S_1 \times \cdots \times S_m$, $U \in A^{\square}$, and for $m = 0$ we have $(A^{\square} \twoheadrightarrow \{1\}) \in \mathfrak{X}$).

For finite sets A and B, a semiring homomorphism $f : (B^{\square}, \cdot, \cup) \to (A^{\square}, \cdot, \cup)$ and $K \subseteq A^*$, $L \subseteq B^*$ we define
$f^{[-1]}(K) = \{ v \in B^* \mid f(\{v\}) \subseteq K \}$, $f^{(-1)}(K) = \{ v \in B^* \mid f(\{v\}) \cap K \neq \emptyset \}$
and $f(L) = \bigcup \{ f(\{v\}) \mid v \in L \}$.
Notice that the last formula extends the domain of f from finite subsets of A^* to arbitrary ones. Further, for any $L, M \subseteq B^*$ we have $f(LM) = f(L)f(M)$ and for the complement K^{c} of a language $K \subseteq A^*$, it holds $f^{[-1]}(K^{\mathsf{c}}) = (f^{(-1)}(K))^{\mathsf{c}}$.

For a multiliteral homomorphism $f : B^{\square} \to A^{\square}$ we define the *dual* multiliteral homomorphism $\widehat{f} : A^{\square} \to B^{\square}$ by $b \in \widehat{f}(\{a\})$ iff $a \in f(\{b\})$ for $a \in A$, $b \in B$. Notice that $f^{(-1)}(K) = \widehat{f}(K)$ for each $K \subseteq A^*$.

A language $L \subseteq A^*$ defines the *syntactic congruence* $\sim_L$ on $(A^{\square}, \cdot, \cup)$ by $\{u_1, \ldots, u_k\} \sim_L \{v_1, \ldots, v_l\}$ if and only if

$$(\forall p, q \in A^*) (pu_1q, ..., pu_kq \in L \iff pv_1q, ..., pv_lq \in L).$$

The factor-structure $(A^{\square}, \cdot, \cup)/\sim_L$ is called the *syntactic semiring* of L; we denote it by $(\mathsf{S}(L), \cdot, \vee)$. The mapping $\phi_L : A^{\square} \twoheadrightarrow \mathsf{S}(L)$, $U \mapsto U \sim_L$ is a surjective semiring homomorphism. We call it the *syntactic homomorphism.*

A *quotient* of a language $L \subseteq A^*$ is a set $p^{-1}Lq^{-1} = \{ u \in A^* \mid puq \in L \}$ where $p, q \in A^*$, $p^{-1}L$ is a *left quotient* and Lq^{-1} is a *right quotient* of L.

3 Eilenberg-Type Theorems

A *class* of (regular) languages is an operator $\mathscr{L}$ assigning to every finite set A a set $\mathscr{L}(A)$ of regular languages over the alphabet A.

Such a class is a *conjunctive variety* if
(i) each $\mathscr{L}(A)$ is closed with respect to finite intersections and quotients, and
(ii) for each finite sets A and B and $f : B^{\square} \to A^{\square}$, $K \in \mathscr{L}(A)$ implies $f^{[-1]}(K) \in \mathscr{L}(B)$.

It is a $\mathbb{D}$-*variety* if (i) is true and (ii) is satisfied for $f \in \mathbb{D}(B^{\square}, A^{\square})$.

Finally, a class $\mathscr{L}$ is a *multiliteral variety* if
(i') each $\mathscr{L}(A)$ is closed with respect to finite unions and quotients, and
(ii) for each finite sets A and B and $f \in \mathbb{D}_{ml}(A^{\square}, B^{\square})$, $K \in \mathscr{L}(A)$ implies $f(K) \in \mathscr{L}(B)$.

We put $\mathscr{K} \leq \mathscr{L}$ if for each finite set A we have $\mathscr{K}(A) \subseteq \mathscr{L}(A)$ and define $\mathscr{L}^{\mathsf{c}}$ by $\mathscr{L}^{\mathsf{c}}(A) = \{ L^{\mathsf{c}} \mid L \in \mathscr{L}(A) \}$.

Fix the category $\mathbb{D}$. We can assign to any class of languages $\mathscr{L}$ the pseudovariety

$$\mathbf{S}(\mathscr{L}) = \langle \{ (\mathsf{S}(L), \cdot, \vee) \mid A \text{ a finite set}, L \in \mathscr{L}(A) \} \rangle_{\mathcal{S}}$$

of idempotent semirings generated by all syntactic semirings of members of $\mathscr{L}$, and the $\mathbb{D}$-pseudovariety

$$\mathbf{S}_{\mathbb{D}}(\mathscr{L}) = \langle \{ \phi_L \mid A \text{ a finite set}, L \in \mathscr{L}(A) \} \rangle_{\mathcal{C},\mathbb{D}}$$

of idempotent semiring homomorphisms generated by all syntactic semiring homomorphisms of members of $\mathcal{L}$.

Conversely, for a class $\mathcal{X}$ of idempotent semirings and a finite set A, we put

$$(\mathsf{L}(\mathcal{X}))(A) = \{ L \subseteq A^* \mid (\mathsf{S}(L), \cdot, \vee) \in \mathcal{X} \}$$

and similarly for a class $\mathfrak{X}$ of idempotent semiring homomorphisms and a finite set A, we put

$$(\mathbf{L}(\mathfrak{X}))(A) = \{ L \subseteq A^* \mid \phi_L \in \mathfrak{X} \},\ (\mathbf{L}^c(\mathfrak{X}))(A) = \{ L \subseteq A^* \mid \phi_{L^c} \in \mathfrak{X} \}.$$

Theorem 1. *(i) The assignments* $\mathcal{L} \mapsto \mathsf{S}(\mathcal{L})$ *and* $\mathcal{X} \mapsto \mathsf{L}(\mathcal{X})$ *are mutually inverse bijections between the conjunctive varieties of languages and pseudovarieties of finite idempotent semirings.*

(ii) The assignments $\mathcal{L} \mapsto \mathbf{S}_{\mathbb{D}}(\mathcal{L})$ *and* $\mathfrak{X} \mapsto \mathbf{L}(\mathfrak{X})$ *are mutually inverse bijections between the* $\mathbb{D}$*-varieties of languages and* $\mathbb{D}$*-pseudovarieties of homomorphisms of idempotent semirings.*

(iii) The assignments $\mathcal{L} \mapsto \mathbf{S}_{\mathbb{D}_{ml}}(\mathcal{L}^c)$ *and* $\mathfrak{X} \mapsto \mathbf{L}^c(\mathfrak{X})$ *are mutually inverse bijections between the multiliteral varieties of languages and* $\mathbb{D}_{ml}$*-pseudovarieties of homomorphisms of idempotent semirings.*

Proof. (i) is the main result of [4].

(ii) : We use several lemmas. Lemma 1 assures that starting from a $\mathbb{D}$-pseudovariety of semiring homomorphisms, the class $\mathbf{L}(\mathfrak{X})$ is a $\mathbb{D}$-variety of languages. Lemma 2 shows that this assignment is one-to-one. The series of Lemmas 3–7 culminates in the result saying that this mapping is a surjective one and that $\mathbf{S}_{\mathbb{D}}$ is its inverse.

(iii) : Take (ii) for $\mathbb{D}_{ml}$ and use complements of considered languages. □

Lemma 1. *Let* $\mathfrak{X}$ *be a* $\mathbb{D}$*-pseudovariety of semiring homomorphisms. Then* $\mathbf{L}(\mathfrak{X})$ *is a* $\mathbb{D}$*-variety of languages.*

Proof. Intersections: Let A be a finite set, let $K, L \in (\mathbf{L}(\mathfrak{X}))(A)$. Then $\sim_K \cap \sim_L \subseteq \sim_{K \cap L}$. Therefore $\sigma : (U\sim_K, U\sim_L) \mapsto U\sim_{K\cap L}$ ($U \in A^{\square}$) is a surjective homomorphism of $\mathsf{im}(\phi_K, \phi_L)$ onto $\mathsf{S}(K \cap L)$ such that $\sigma(\phi_K, \phi_L) = \phi_{K\cap L}$ and thus $\phi_{K\cap L} \in \mathfrak{X}$.

Quotients: Notice that for each $L \subseteq A^*$, $a \in A$ we have $\sim_L \subseteq \sim_{a^{-1}L}$. Therefore there is a surjective homomorphism $\sigma : \mathsf{S}(L) \twoheadrightarrow \mathsf{S}(a^{-1}L)$ such that $\sigma\phi_L = \phi_{a^{-1}L}$ and thus $\phi_{a^{-1}L} \in \mathfrak{X}$. Similarly for La^{-1}.

Preimages: Let $K \in (\mathbf{L}(\mathfrak{X}))(A)$, $f \in \mathbb{D}(B^{\square}, A^{\square})$ and $L = f^{[-1]}(K)$. Direct calculations yield that $\ker(\phi_K f) \subseteq \ker \phi_L$, and so there is a surjective semiring homomorphism $\sigma : \mathsf{im}(\phi_K f) \twoheadrightarrow \mathsf{S}(L)$ such that $\sigma\phi_K f = \phi_L$. Thus $\phi_L \in \mathfrak{X}$. □

Lemma 2. *Let* $\mathfrak{X}, \mathfrak{Y}$ *be* $\mathbb{D}$*-pseudovarieties of semiring homomorphisms. Then* $\mathbf{L}(\mathfrak{X}) = \mathbf{L}(\mathfrak{Y})$ *implies* $\mathfrak{X} = \mathfrak{Y}$.

Proof. Suppose $\mathbf{L}(\mathfrak{X}) \leq \mathbf{L}(\mathfrak{Y})$ and let $(\phi : A^{\square} \twoheadrightarrow S) \in \mathfrak{X}$ and $s \in S$. Put $L_s = \{ u \in A^* \mid \phi(\{u\}) \subseteq (s] \}$. One easily verifies that $\ker\phi \subseteq \ker\phi_{L_s}$ and thus there is a surjective homomorphism $\sigma : S \twoheadrightarrow \mathsf{S}(L_s)$ such that $\sigma\phi = \phi_{L_s}$. Therefore $\phi_{L_s} \in \mathfrak{X}$ and $L_s \in (\mathsf{L}(\mathfrak{X}))(A) \subseteq (\mathsf{L}(\mathfrak{Y}))(A)$. Consequently, $\phi_{L_s} \in \mathfrak{Y}$.

Let $S = \{s_1, \ldots, s_m\}$. Direct computations lead to $\ker(\phi_{L_{s_1}}, \ldots, \phi_{L_{s_m}}) \subseteq \ker\phi$ and thus there is a surjective homomorphism $\tau : \mathsf{im}(\phi_{L_{s_1}}, \ldots, \phi_{L_{s_m}}) \twoheadrightarrow S$ such that $\phi = \tau(\phi_{L_{s_1}}, \ldots, \phi_{L_{s_m}})$. Consequently, $\phi \in \mathfrak{Y}$. □

We define the $\mathbb{D}$-divisibility relation on the class $\mathfrak{S}$ as follows : $\phi : A^{\square} \twoheadrightarrow S$ *$\mathbb{D}$-divides* $\psi : B^{\square} \twoheadrightarrow T$ if there exist $f \in \mathbb{D}(A^{\square}, B^{\square})$ and $\sigma : \mathsf{im}(\psi f) \twoheadrightarrow S$ such that $\phi = \sigma\psi f$. We say that ϕ $\mathbb{D}$-divides ψ *using* f and σ.

Lemma 3. *Let $K \subseteq A^*$, $\phi : B^{\square} \twoheadrightarrow T$, $f \in \mathbb{D}(A^{\square}, B^{\square})$. The following conditions are equivalent :*
(i) ϕ_K $\mathbb{D}$-divides ϕ using f and some $\sigma : \mathsf{im}(\phi f) \twoheadrightarrow \mathsf{S}(K)$,
(ii) there exists an ideal N of T such that $K = \{ u \in A^* \mid (\phi f)(\{u\}) \in N \}$.

Proof. (i) $\Rightarrow$ (ii) : Take for N the ideal of T generated by the ideal $\sigma^{-1}(\{ \phi_K(\{u_1, \ldots, u_k\}) \mid k \geq 0,\ u_1, \ldots, u_k \in K \})$ of $\mathsf{im}(\phi f)$.
(ii) $\Rightarrow$ (i) follows from $\ker(\phi f) \subseteq \ker\phi_K$. □

Lemma 4. *If $K \subseteq A^*$, $L \subseteq B^*$ and ϕ_K $\mathbb{D}$-divides ϕ_L using f and σ, then there exist $m \geq 1$ and $p_1, \ldots, p_m, q_1, \ldots, q_m \in B^*$ such that*

$$K = f^{[-1]}(\textstyle\bigcap_{i=1,\ldots,m} p_i^{-1} L q_i^{-1}).$$

Proof. By Lemma 3, there exists an ideal N of $\mathsf{S}(L)$ such that $K = \{ u \in A^* \mid (\phi_L f)(\{u\}) \in N \}$. This can be expressed also as $K = f^{[-1]}(\{ v \in B^* \mid \{v\} \in \phi_L^{-1}(N) \})$. The finiteness of $\mathsf{S}(L)$ implies that $N = (\phi_L(U)]$ for some $U \in B^{\square}$. Further, for each $V \in B^{\square}$ we have $\phi_L(V) \leq \phi_L(U)$ if and only if $V \subseteq \bigcap_{p,q \in B^*,\ U \subseteq p^{-1}Lq^{-1}} p^{-1} L q^{-1}$. Therefore this intersection equals $\{ v \in B^* \mid \{v\} \in \phi_L^{-1}(N) \}$. Finally notice that there are only finitely many quotients of a regular language. □

For a class $\mathfrak{L}$ of languages we define

$$\mathbf{E}_{\mathbb{D}}(\mathfrak{L}) = \{ \psi \in \mathfrak{S} \mid \text{ if } L \subseteq A^* \text{ and } \phi_L \text{ } \mathbb{D}\text{-divides } \psi, \text{ then } L \in \mathfrak{L}(A) \} .$$

Lemma 5. *Let $\mathfrak{L}$ be a $\mathbb{D}$-variety of languages. Then $\mathbf{E}_{\mathbb{D}}(\mathfrak{L})$ is a $\mathbb{D}$-pseudovariety of semiring homomorphisms.*

Proof. (H) : Let $(\phi : A^{\square} \twoheadrightarrow S) \in \mathbf{E}_{\mathbb{D}}(\mathfrak{L})$ and let $\sigma : S \twoheadrightarrow T$ be a surjective semiring homomorphisms. We show that $\sigma\phi \in \mathbf{E}_{\mathbb{D}}(\mathfrak{L})$.

Indeed, let $L \subseteq B^*$ and let ϕ_L $\mathbb{D}$-divide $\sigma\phi$ using f and τ. Then ϕ_L $\mathbb{D}$-divides ϕ using f and $\tau\sigma$ and thus $L \in \mathfrak{L}(B)$.

($\mathrm{S}_{\mathbb{D}}$) : Let $(\phi : A^{\square} \twoheadrightarrow S) \in \mathbf{E}_{\mathbb{D}}(\mathfrak{L})$ and let $f \in \mathbb{D}(B^{\square}, A^{\square})$. We show that $\phi f \in \mathbf{E}_{\mathbb{D}}(\mathfrak{L})$.

Indeed, let $\phi_L : C^{\square} \twoheadrightarrow \mathsf{S}(L)$ $\mathbb{D}$-divide ϕf using $g \in \mathbb{D}(C^{\square}, B^{\square})$ and σ. Then ϕ_L $\mathbb{D}$-divides ϕ using fg and σ. Thus $L \in \mathcal{L}(A)$.

(P) : For the sake of transparency we consider the case $m = 2$. So let $\phi : A^{\square} \twoheadrightarrow S$, $\psi : A^{\square} \twoheadrightarrow T$ be elements of $\mathbf{E}_{\mathbb{D}}(\mathcal{L})$. We show that $(\phi, \psi) \in \mathbf{E}_{\mathbb{D}}(\mathcal{L})$.

Indeed, let $\phi_L : B^{\square} \twoheadrightarrow \mathsf{S}(L)$ $\mathbb{D}$-divide (ϕ, ψ) using f and σ. By Lemma 3 there is an ideal N in $\mathsf{im}(\phi, \psi)$ such that $L = \{ v \in B^* \mid ((\phi, \psi)f)(\{v\}) \in N \}$. Finiteness of $\mathsf{im}(\phi, \psi)$ gives the existence of $s \in S$, $t \in T$ such that $N = ((s, t)]$. Put $L_1 = \{ v \in B^* \mid (\phi f)(\{v\}) \in (s] \}$ and $L_2 = \{ v \in B^* \mid (\psi f)(\{v\}) \in (t] \}$. Then by Lemma 3, ϕ_{L_1} $\mathbb{D}$-divides ϕ and ϕ_{L_2} $\mathbb{D}$-divides ψ and so $L_1, L_2 \in \mathcal{L}(B)$.

Further, $((\phi, \psi)f)^{-1}((s] \times T) = \{ \{v_1, \ldots, v_l\} \in B^{\square} \mid l \geq 0,\ v_1, \ldots, v_l \in L_1 \}$ and $((\phi, \psi)f)^{-1}(S \times (t]) = \{ \{v_1, \ldots, v_l\} \in B^{\square} \mid l \geq 0,\ v_1, \ldots, v_l \in L_2 \}$ and thus $L = L_1 \cap L_2 \in \mathcal{L}(B)$. □

Lemma 6. *Let $\mathcal{L}$ be a $\mathbb{D}$-variety of languages. Then $\mathbf{S}_{\mathbb{D}}(\mathcal{L}) \leq \mathbf{E}_{\mathbb{D}}(\mathcal{L})$.*

Proof. Let B be a finite set and let $L \in \mathcal{L}(B)$. We show that $\phi_L \in \mathbf{E}_{\mathbb{D}}(\mathcal{L})$. Indeed, if $\phi_K : A^{\square} \twoheadrightarrow \mathsf{S}(K)$ $\mathbb{D}$-divides ϕ_L then by Lemma 4, $K \in \mathcal{L}(A)$. When generating $\mathbf{S}_{\mathbb{D}}(\mathcal{L})$ we stay in $\mathbf{E}_{\mathbb{D}}(\mathcal{L})$ due to Lemma 5. □

Lemma 7. *Let $\mathcal{L}$ be a $\mathbb{D}$-variety of languages. Then $\mathcal{L} = \mathbf{L}(\mathbf{S}_{\mathbb{D}}(\mathcal{L}))$.*

Proof. The "$\leq$" is obvious. So let A be a finite set and let $L \in (\mathbf{L}(\mathbf{S}_{\mathbb{D}}(\mathcal{L})))(A)$. By Lemma 6, $L \in (\mathbf{L}(\mathbf{E}_{\mathbb{D}}(\mathcal{L})))(A)$ and therefore $\phi_L \in \mathbf{E}_{\mathbb{D}}(\mathcal{L})$. Now ϕ_L $\mathbb{D}$-divides itself and so $L \in \mathcal{L}(A)$. □

4 Pseudoidentities and Reiterman-Type Theorem

This section modifies the approach by Kunc (see [2]) concerning pseudovarieties of monoid homomorphisms to the case of semirings.

Recall that $(\pi_S)_{S \in \mathcal{S}}$ is an *n-ary implicit operation* ($n \geq 0$) for the class $\mathcal{S}$ of all finite idempotent semirings if $\pi_S : S^n \to S$ ($S \in \mathcal{S}$) is a mapping and for each semiring homomorphism $\sigma : S \to T$ and $s_1, \ldots, s_n \in S$ we have

$$\sigma(\pi_S(s_1, \ldots, s_n)) = \pi_T(\sigma(s_1), \ldots, \sigma(s_n)) .$$

In particular, if S is a substructure of T, then π_S is the restriction of π_T to S, and similarly for the product $S \times T$:
$\pi_{S \times T}((s_1, t_1), \ldots, (s_n, t_n)) = (\pi_S(s_1, \ldots, s_n),\ \pi_T(t_1 \ldots, t_n))$.

We denote the set of all n-ary implicit operations for $\mathcal{S}$ by $\mathcal{I}_n$. Widely used is $x^{\omega} \in \mathcal{I}_1$ where $x_S^{\omega}(a)$ for $a \in S$ is the only idempotent in the subsemigroup of S generated by a.

An *n-ary pseudoidentity* is an ordered pair $\pi = \rho$ of n-ary implicit operations. Let $v_1, v_2, \ldots$ be a fixed sequence of pairwise different variables and let $V_n = \{v_1, \ldots, v_n\}$ for each $n \geq 0$. Let $\mathbb{D}$ be a category of of homomorphisms of free finitely generated idempotent semirings. The pseudoidentity $\pi = \rho$ is *$\mathbb{D}$-satisfied*

in a semiring homomorphism $(\phi : A^{\square} \twoheadrightarrow S) \in \mathfrak{S}$ if for each $f \in \mathbb{D}(V_n^{\square}, A^{\square})$ we have

$$\pi_S((\phi f)(v_1), \ldots, (\phi f)(v_n)) = \rho_S((\phi f)(v_1), \ldots, (\phi f)(v_n)) \ .$$

We write $\phi \models_{\mathbb{D}} \pi = \rho$ in such a case and for a set Π of pseudoidentities $\phi \models_{\mathbb{D}} \Pi$ means $(\forall (\pi = \rho) \in \Pi)\ \phi \models_{\mathbb{D}} \pi = \rho$. Let $\mathsf{Mod}_{\mathbb{D}}(\Pi) = \{ \phi \in \mathfrak{S} \mid \phi \models_{\mathbb{D}} \Pi \}$.

In the case $(\phi : A^{\square} \twoheadrightarrow S) \models_{\mathbb{D}_l} \pi = \rho$ we say that ϕ satisfies the pseudoidentity $\pi = \rho$ *literalry.*

Theorem 2. *Let Π be a set of pseudoidentities. Then the class $\mathfrak{X} = \mathsf{Mod}_{\mathbb{D}}(\Pi)$ is a $\mathbb{D}$-pseudovariety of semiring homomorphisms.*

Proof. It is enough to consider $\Pi = \{\pi = \rho\}$, $\pi, \rho \in \mathcal{I}_n$.

(H) : Let $(\phi : A^{\square} \twoheadrightarrow S) \models_{\mathbb{D}} \pi = \rho$, let $\sigma : S \twoheadrightarrow T$ be a surjective semiring homomorphism and let $f \in \mathbb{D}(V_n^{\square}, A^{\square})$. Then $\sigma\phi \models_{\mathbb{D}} \pi = \rho$ since
$\pi_T((\sigma\phi f)(v_1), \ldots, (\sigma\phi f)(v_n)) = \sigma(\pi_S((\phi f)(v_1), \ldots, (\phi f)(v_n)))$
$= \sigma(\rho_S((\phi f)(v_1), \ldots, (\phi f)(v_n))) = \rho_T((\sigma\phi f)(v_1), \ldots, (\sigma\phi f)(v_n))$.

($\mathbb{S}_{\mathbb{D}}$) : Let $(\phi : A^{\square} \twoheadrightarrow S) \models_{\mathbb{D}} \pi = \rho$, let $f \in \mathbb{D}(B^{\square}, A^{\square})$. Then $\phi f \models_{\mathbb{D}} \pi = \rho$ since, for each $g \in \mathbb{D}(V_n^{\square}, B^{\square})$,
$\pi_{\mathsf{im}\,(\phi f)}(((\phi f)g)(v_1), \ldots, ((\phi f)g)(v_n)) = \pi_S(((\phi f)g)(v_1), \ldots, ((\phi f)g)(v_n))$
$= \pi_S((\phi(fg))(v_1), \ldots, (\phi(fg))(v_n)) = \rho_S((\phi(fg))(v_1), \ldots, (\phi(fg))(v_n))$
$= \rho_S(((\phi f)g)(v_1), \ldots, ((\phi f)g)(v_n)) = \rho_{\mathsf{im}\,(\phi f)}(((\phi f)g)(v_1), \ldots, ((\phi f)g)(v_n))$.

(P) : Let $(\phi_i : A^{\square} \twoheadrightarrow S_i) \models_{\mathbb{D}} \pi = \rho$ for $i = 1, \ldots m$. Then $(\phi_1, \ldots, \phi_m) \models_{\mathbb{D}} \pi = \rho$ since for each $f \in \mathbb{D}(V_n^{\square}, A^{\square})$,
$\pi_{\mathsf{im}\,(\phi_1, \ldots, \phi_m)}(((\phi_1, \ldots, \phi_m)f)(v_1), \ldots, ((\phi_1, \ldots, \phi_m)f)(v_n))$
$= \pi_{S_1 \times \ldots \times S_m}(((\phi_1, \ldots, \phi_m)f)(v_1), \ldots, ((\phi_1, \ldots, \phi_m)f)(v_n))$
$= (\pi_{S_1}((\phi_1 f)(v_1), \ldots, (\phi_1 f)(v_n)), \ldots, \pi_{S_m}((\phi_1 f)(v_1), \ldots, (\phi_1 f)(v_n)))$
$= (\rho_{S_1}((\phi_1 f)(v_1), \ldots, (\phi_1 f)(v_n)), \ldots, \rho_{S_m}((\phi_1 f)(v_1), \ldots, (\phi_1 f)(v_n))) = \ldots$
$= \rho_{\mathsf{im}\,(\phi_1, \ldots, \phi_m)}(((\phi_1, \ldots, \phi_m)f)(v_1), \ldots, ((\phi_1, \ldots, \phi_m)f)(v_n))$. □

The non-trivial part of our Reiterman-type theorem needs a preparation. We define the multiplication on the set $\mathcal{I}_n$ by

$$(\pi\rho)_S(s_1, \ldots, s_n) = \pi_S(s_1, \ldots, s_n) \cdot \rho_S(s_1, \ldots, s_n), \ S \in \mathcal{S}, \ s_1, \ldots, s_n \in S$$

and similarly for the operation of join. Then the structure $(\mathcal{I}_n, \cdot, \vee)$ is again an idempotent semiring. There is also a natural metric on $\mathcal{I}_n$; it is defined by

$$d(\pi, \rho) = 2^{-r(\pi,\rho)} \quad \text{where} \quad r(\pi, \rho) = \min\{ |S| \mid S \in \mathcal{S}, \ \pi_S \neq \rho_S \}, \ 2^{-\infty} = 0 \ .$$

Finite semirings are endowed by discrete topologies. Since $V_n^{\square}$ is a free idempotent semiring over the set $\{v_1, \ldots, v_n\}$, the mapping

$$v_i \mapsto (i\text{-th } n\text{-ary projection on } S)_{S \in \mathcal{S}}, \ i = 1, \ldots, n$$

extends to a homomorphism $V_n^{\square} \to \mathcal{I}_n$; we denote it by ν_n.

Lemma 8 (Almeida [1] 3.4.5, 3.4.7, 3.4.13). *The structure $\mathcal{I}_n$ is a compact idempotent semiring. The substructure $\nu_n(V_n^{\square})$ is dense in $\mathcal{I}_n$. Moreover, for every homomorphism $\phi : V_n^{\square} \to S$, there exists exactly one continuous homomorphism $\mu_n : \mathcal{I}_n \to S$ such that $\mu_n \nu_n = \phi$. It is given by $\pi \mapsto \pi_S(\phi(v_1), \ldots, \phi(v_n))$, $\pi \in \mathcal{I}_n$.* □

Let $\mathbb{D}$ be a category of homomorphisms of free finitely generated idempotent semirings, let $\mathfrak{X} \subseteq \mathfrak{S}$ and let n be a natural number. We put

$$J_n^{\mathbb{D},\mathfrak{X}} = \{ (\pi, \rho) \in \mathcal{I}_n \times \mathcal{I}_n \mid \text{ for all } (\phi : A^{\square} \twoheadrightarrow S) \in \mathfrak{X} \text{ we have } \phi \models_{\mathbb{D}} \pi = \rho \} .$$

Clearly, $J_n^{\mathbb{D},\mathfrak{X}}$ is a congruence on $\mathcal{I}_n$ and we define also a metric on $\mathcal{I}_n^{\mathbb{D},\mathfrak{X}} = \mathcal{I}_n / J_n^{\mathbb{D},\mathfrak{X}}$ by

$$d(\pi J_n^{\mathbb{D},\mathfrak{X}}, \rho J_n^{\mathbb{D},\mathfrak{X}}) = 2^{-r} \quad \text{where } r = \min\{ |S| \mid (\phi : A^{\square} \twoheadrightarrow S) \in \mathfrak{X},\ \phi \not\models_{\mathbb{D}} \pi = \rho \} .$$

Denote $\nu_n^{\mathbb{D},\mathfrak{X}}$ the natural projection of $\mathcal{I}_n$ onto $\mathcal{I}_n^{\mathbb{D},\mathfrak{X}}$.

The following lemma is a crucial one.

Lemma 9. *The homomorphism $\nu_n^{\mathbb{D},\mathfrak{X}} : \mathcal{I}_n \to \mathcal{I}_n^{\mathbb{D},\mathfrak{X}}$ is a continuous mapping and thus $\mathcal{I}_n^{\mathbb{D},\mathfrak{X}}$ is again compact. Moreover, for every $(\phi : V_n^{\square} \twoheadrightarrow S) \in \mathfrak{S}$ such that $\phi \models_{\mathbb{D}} J_n^{\mathbb{D},\mathfrak{X}}$, there exists exactly one continuous homomorphism $\mu_n^{\mathbb{D},\mathfrak{X}} : \mathcal{I}_n^{\mathbb{D},\mathfrak{X}} \to S$ such that $\mu_n^{\mathbb{D},\mathfrak{X}} \nu_n^{\mathbb{D},\mathfrak{X}} \nu_n = \phi$. It is given by $\pi J_n^{\mathbb{D},\mathfrak{X}} \mapsto \pi_S(\phi(v_1), \ldots, \phi(v_n))$, $\pi \in \mathcal{I}_n$.*

Proof. Let $\pi, \rho \in \mathcal{I}_n$, $\varepsilon > 0$, $d(\pi, \rho) = 2^{-r} < \varepsilon$. Then for all $S \in \mathcal{S}$ with $|S| < r$ we have $\pi_S = \rho_S$ and thus also $d(\pi J_n^{\mathbb{D},\mathfrak{X}} \rho J_n^{\mathbb{D},\mathfrak{X}}) < \varepsilon$. Thus the mapping $\nu_n^{\mathbb{D},\mathfrak{X}}$ is continuous.

Due to $\ker \nu_n^{\mathbb{D},\mathfrak{X}} \subseteq \ker \mu_n$ the mapping $\mu_n^{\mathbb{D},\mathfrak{X}} : \pi J_n^{\mathbb{D},\mathfrak{X}} \mapsto \pi_S(\phi(v_1), \ldots, \phi(v_n))$ is well-defined and it is the only one satisfying $\mu_n^{\mathbb{D},\mathfrak{X}} \nu_n^{\mathbb{D},\mathfrak{X}} = \mu_n$. We will show next that it is continuous. Indeed, take a subset C of S. Using Lemma 8, the set $\mu_n^{-1}(C)$ is closed in a compact space and thus also compact. Therefore $(\mu_n^{\mathbb{D},\mathfrak{X}})^{-1}(C) = \nu_n^{\mathbb{D},\mathfrak{X}}(\mu_n^{-1}(C))$ is compact and therefore closed.

Also the uniqueness follows from Lemma 8. □

We are prepared to prove the main theorem of this section.

Theorem 3. *Let $\mathbb{D}$ be a category of homomorphisms of free finitely generated idempotent semirings containing all the bijections and let $\mathfrak{X} \subseteq \mathfrak{S}$ be a $\mathbb{D}$-pseudovarity. Then there exists a set Σ of pseudoidentities such that $\mathfrak{X} = \mathsf{Mod}_{\mathbb{D}}(\Sigma)$.*

Proof. Due to our restriction on the category $\mathbb{D}$, the $\mathbb{D}$-pseudovariety $\mathfrak{X}$ is fully characterized by its homomorphisms of the form $\phi : V_n^{\square} \twoheadrightarrow S$, n a natural number. Let

$$\Pi = \{ (\pi, \rho) \in \mathcal{I}_n \mid n \text{ a natural number}, (\forall\, \phi : V_n^{\square} \twoheadrightarrow S \in \mathfrak{X}) \ \phi \models_{\mathbb{D}} \Pi \} .$$

Clearly, $\mathfrak{X} \subseteq \mathsf{Mod}_{\mathbb{D}}(\Pi)$. So let $(\phi : V_n^{\square} \twoheadrightarrow S) \in \mathfrak{S}$, $\phi \models_{\mathbb{D}} \Pi$. The homomorphism $\mu_n^{\mathbb{D},\mathfrak{X}}$ from Lemma 9 is a continuous mapping from a compact space to a discrete finite space and thus there exists $\varepsilon_0 > 0$ such that

$$(\forall\, \pi, \rho \in \mathcal{I}_n) \ (\ d(\pi J_n^{\mathbb{D},\mathfrak{X}} \rho J_n^{\mathbb{D},\mathfrak{X}}) < \varepsilon_0 \ \Rightarrow \ \mu_n^{\mathbb{D},\mathfrak{X}}(\pi J_n^{\mathbb{D},\mathfrak{X}}) = \mu_n^{\mathbb{D},\mathfrak{X}}(\rho J_n^{\mathbb{D},\mathfrak{X}}) \ ) .$$

Let r be a natural number such that $2^{-r} \leq \varepsilon_0$ and let $\mathfrak{X}_0$ be the a set of representatives of all isomorphism classes of homomorphisms from $\mathfrak{X}$ of the form $\psi : V_n^{\square} \twoheadrightarrow T$ where $|T| \leq r$ (i.e., for every $(\psi : V_n^{\square} \twoheadrightarrow T) \in \mathfrak{X}$ with $|T| \leq r$ there exists $(\psi' : V_n^{\square} \twoheadrightarrow T') \in \mathfrak{X}_0$ and an isomorphism $\alpha : T' \to T$ such that $\psi = \alpha\psi'$).

The product $\chi = (\psi)_{\psi \in \mathfrak{X}_0} : V_n^{\square} \twoheadrightarrow \mathsf{im}\,\chi \subseteq \prod_{\psi \in \mathfrak{X}_0} \mathsf{im}\,\psi$ is an element of $\mathfrak{X}$. Again by Lemma 9, there exists a homomorphism $\xi_n^{\mathbb{D},\mathfrak{X}} : \mathcal{I}_n^{\mathbb{D},\mathfrak{X}} \mapsto \mathsf{im}\,\xi$ such that $\xi_n^{\mathbb{D},\mathfrak{X}} \nu_n^{\mathbb{D},\mathfrak{X}} \nu_n = \chi$. In order to verify $\phi \in \mathfrak{X}$, it is enough to show that $\ker \xi_n^{\mathbb{D},\mathfrak{X}} \subseteq \ker \mu_n^{\mathbb{D},\mathfrak{X}}$. So let $\pi, \rho \in \mathcal{I}_n$, $\mu_n^{\mathbb{D},\mathfrak{X}}(\pi J_n^{\mathbb{D},\mathfrak{X}}) \neq \mu_n^{\mathbb{D},\mathfrak{X}}(\rho J_n^{\mathbb{D},\mathfrak{X}})$. Then $d(\pi J_n^{\mathbb{D},\mathfrak{X}}, \rho J_n^{\mathbb{D},\mathfrak{X}}) > \varepsilon_0 \geq 2^{-r}$ and there exists $(\psi : V_m^{\square} \to T) \in \mathfrak{X}$ with $|T| \leq r$, $\psi \not\models_{\mathbb{D}} \pi = \rho$. Thus there exists $f \in \mathbb{D}(V_n^{\square}, V_m^{\square})$ such that

$$\pi_T((\psi f)(v_1), \ldots, (\psi f)(v_n)) \neq \rho_T((\psi f)(v_1), \ldots, (\psi f)(v_n)) \ .$$

Let α be as above and let p be the projection of $\mathsf{im}\,\chi$ onto T'. Finally,
$(\alpha p)(\pi_{\mathsf{im}\,\chi}(\chi(v_1), \ldots, \chi(v_n)) = \pi_{im(\psi f)}((\psi f)(v_1), \ldots, (\psi f)(v_n))$
$= \pi_T((\psi f)(v_1), \ldots, (\psi f)(v_n))$ and similarly for ρ instead of π. Thus

$$\pi_{\mathsf{im}\,\chi}(\chi(v_1), \ldots, \chi(v_n)) \neq \rho_{\mathsf{im}\,\chi}(\chi(v_1), \ldots, \chi(v_n)) \text{ and } \xi_n^{\mathbb{D},\mathfrak{X}}(\pi) \neq \xi_n^{\mathbb{D},\mathfrak{X}}(\rho) \ .$$

□

5 Some Multiliteral Varieties of Languages

We define the classes of languages $\mathcal{S}(n)$, $\mathcal{H}_{1/2}(k)$, $\mathcal{H}_{3/2}(k)$, $\mathcal{H}_{1/2}$, $\mathcal{H}_{3/2}$, $\mathcal{L}(k,m,l)$, $\mathcal{L}(m)$ below. It is obvious that all of them are multiliteral varieties of languages.

For a finite set A and $n, k, l \geq 0$, $m \geq 1$ put
$\mathcal{S}(n)(A)$ = class of all languages over A of star height $\leq n$ (see Pin [3]),
$\mathcal{H}_{1/2}(k)(A)$ = all finite unions of $A^* a_1 A^* a_2 \ldots a_p A^*$, where $0 \leq p \leq k$, $a_1, \ldots, a_p \in A$,
$\mathcal{H}_{3/2}(k)(A)$ = all finite unions of $B_0^* a_1 B_1^* a_2 \ldots a_p B_p^*$, where $0 \leq p \leq k$, $a_1, \ldots, a_p \in A$ and $B_0, \ldots, B_p \subseteq A$,
$\mathcal{L}(k,m,l)$ = all finite unions of $uB_1^* \ldots B_m^* v$, where $u, v \in A^*, |u| \leq k, |v| \leq l, B_1, \ldots, B_m \subseteq A$.

Further, put $\mathcal{H}_{1/2}(A) = \bigcup_{k \geq 0} \mathcal{H}_{1/2}(k)(A)$, $\mathcal{H}_{3/2}(A) = \bigcup_{k>0} \mathcal{H}_{3/2}(k)(A)$, $\mathcal{L}(m)(A) = \bigcup_{k,l \geq 0} \mathcal{L}(k,m,l)(A)$.

At present we are not able to characterize the membership in $\mathcal{S}(n)(A)$ by pseudoidentities. The varieties $\mathcal{H}_{1/2}$ and $\mathcal{H}_{3/2}$ are members of the so-called Straubing-Thérien hierarchy (see [3]). We consider pseudoidentities for the classes $\mathcal{L}(k,m,l)$ and $\mathcal{L}(m)$ below. In particular, we get the decidability results for $m = 1$ and $m = 2$.

Let $n, m \geq 1$. We fix notation for certain n-ary implicit operations :
$\pi_n = x_1 \ldots x_n$,

$$\rho_{n,m} = \bigvee_{1 \leq r_1 < r_2 < \ldots < r_t = n,\ t \leq m} (x_1 \ldots x_{r_1})^\omega (x_{r_1+1} \ldots x_{r_2})^\omega \ldots (x_{r_{t-1}+1} \ldots x_{r_t})^\omega \ .$$

Further, for implicit operations $\pi, \rho \in \mathcal{I}_n$ and $k, l \geq 0$ choose variables $y_1, \ldots, y_k, z_1, \ldots, z_l$ which do not appear in $\pi = \rho$ and define $(\pi = \rho)(k, l)$ as $y_1 \ldots y_k \pi z_1 \ldots z_l = y_1 \ldots y_k \rho z_1 \ldots z_l$.

Denoting $T = \{ \phi_{L^c}(\{a\}) \mid a \in A\}$, it is possible to calculate the sets $T^2, T^3, \ldots, T^\omega$ in $2^{\mathsf{S}(L^c)}$. Recall that $\sigma \leq \tau$ is an abbreviation for $\sigma \vee \tau = \tau$.

Theorem 4. *(i) A language $L \subseteq A^*$ is a member of $\mathcal{L}(k, m, l)(A)$ if and only if the syntactic homomorphism ϕ_{L^c} of the complement of L literalry satisfies $(x \leq 1)(k, l)$, $(\pi_n \leq \rho_{n,m})(k, l)$ for all $n \geq 1$.*

(ii) In (i) it is enough to consider $n \leq |A|$ for $m = 1$, and only $n \leq 2|A|$ for $m = 2$.

(iii) When checking the validity of $(x \leq 1)(k, l)$ or $(\pi_n \leq \rho_{n,m})(k, l)$ in ϕ_{L^c} is suffices to substitute for $y_1 \ldots y_k$ and $z_1 \ldots z_l$ only the elements of T^k and T^l, respectively.

(iv) A language $L \subseteq A^*$ is an element of $\mathcal{L}(m)(A)$ if and only if ϕ_{L^c} satisfies $(x \leq 1)(1, 1)$ and $(\pi_n \leq \rho_{n,m})(1, 1)$ for all n with the substitutions $t, t' \in T^\omega$ for y_1, z_1 and literal substitutions for $x_1, x_2, \ldots$.

Proof. We will use the following observation several times. It holds
$\{u\} \sim_{L^c} \leq \{v_1, \ldots, v_l\} \sim_{L^c}$ in $\mathsf{S}(L^c)$ if and only if
$(\forall p, q \in A^*)(\, u \in p^{-1}Lq^{-1}$ implies $\exists\, i \in \{1, \ldots, l\}$ such that $v_i \in p^{-1}Lq^{-1}\,)$.

(i) : Let $L \in \mathcal{L}(k, m, l)(A)$ and let $pa_1 \ldots a_k b c_1 \ldots c_l q \in uB_1^* \ldots B_m^* v \subseteq L$ where $a_1, \ldots, a_k, b, c_1, \ldots, c_l \in A$, $B_1, \ldots, B_m \subseteq A$, $p, q, u, v \in A^*$, $|u| \leq k$, $|v| \leq l$. Then also $pa_1 \ldots a_k c_1 \ldots c_l q \in uB_1^* \ldots B_m^* v$ and so ϕ_{L^c} literarly satisfies the pseudoidentity $(x \leq 1)(k, l)$.

Analogously, let $n \geq 1$ and let $pa_1 \ldots a_k b_1 \ldots b_n c_1 \ldots c_l q \in uB_1^* \ldots B_m^* v \subseteq L$ where $b_1, \ldots, b_n \in A$. This leads to a decomposition of the segment $b_1 \ldots b_n$ into $t \leq m$ parts $(b_1 \ldots b_{r_1}) \cdot (b_{r_1+1} \ldots b_{r_2}) \cdot \ldots \cdot (b_{r_{t-1}+1} \ldots b_{r_t})$ such that $1 \leq r_1 < r_2 < \ldots < r_t = n$, $b_1 \ldots b_{r_1} \in B_{s_1}^*$, $b_{r_1+1} \ldots b_{r_2} \in B_{s_2}^*, \ldots$, $b_{r_{t-1}+1} \ldots b_{r_t} \in B_{s_t}^*$ where $1 \leq s_1 < s_2 < \ldots s_t \leq m$ and thus

$$a_1 \ldots a_k (b_1 \ldots b_{r_1})^{\beta_1} (b_{r_1+1} \ldots b_{r_2})^{\beta_2} \ldots (b_{r_{t-1}+1} \ldots b_{r_t})^{\beta_t} c_1 \ldots c_l \in p^{-1}Lq^{-1}$$

for arbitrary $\beta_1, \ldots, \beta_t \geq 1$ and so the pseudoidentity $(\pi_n \leq \rho_{n,m})(k, l)$ is literally satisfied in ϕ_{L^c}.

Conversely, let ϕ_{L^c} literalry satisfy the pseudoidentities $(x \leq 1)(k, l)$ and $(\pi_n \leq \rho_{n,m})(k, l)$ for all $n \geq 1$. Let
$K = \bigcup \{\, uB_1^* \ldots B_m^* v \subseteq L \mid u, v \in A^*, |u| \leq k, |v| \leq l, B_1, \ldots, B_m \subseteq A \,\}$.
Notice that there are only finitely many summands here. Clearly $K \subseteq L$ and we show the opposite inclusion.

Obviously, each $w \in L$ with $|w| \leq k + l$ is an element of K. So let $w = a_1 \ldots a_k b_1 \ldots b_n\ c_1 \ldots c_l \in L$ with $a_1, \ldots, a_k, b_1, \ldots, b_n, c_1, \ldots, c_l \in A$. There exist $1 \leq r_1 < r_2 < \ldots < r_t = n$ with $t \leq m$ such that $(\forall\, \alpha \geq 1)(\exists\, \delta \geq 1)$
$a_1 \ldots a_k (b_1 \ldots b_{r_1})^\delta (b_{r_1+1} \ldots b_{r_2})^\delta \ldots (b_{r_{t-1}+1} \ldots b_{r_t})^\delta c_1 \ldots c_l \in L$.
Indeed, let $(b_1 \ldots b_{r_1})^\omega \sim_L (b_1 \ldots b_{r_1})^{\beta_1}, (b_{r_1+1} \ldots b_{r_2})^\omega \sim_L (b_{r_1+1} \ldots b_{r_2})^{\beta_2}$, $\ldots, (b_{r_{t-1}+1} \ldots b_{r_t})^\omega \sim_L (b_{r_{t-1}+1} \ldots b_{r_t})^{\beta_t}$.
Take for δ a common multiple of $\beta_1, \ldots, \beta_t$ exceeding α.

The first pseudoidentity allows us to delete repeatedly in $a_1 \dots a_k d_1 \dots d_s c_1 \dots c_l \in L$ any of the d's, which leads to $a_1 \dots a_k (b_1 + \dots + b_{r_1})^* (b_{r_1+1} + \dots + b_{r_2})^* \dots (b_{r_{t-1}+1} + \dots + b_{r_t})^* c_1 \dots c_l \subseteq L$.

(ii) : Let K be defined as in (i). Suppose that there is a word $w \in L \setminus K$. Take a shortest one. By the proof of (i), $|w| > |A| + k + l$ in the case of $m = 1$ and $|w| > 2|A| + k + l$ for $m = 2$.

$m = 1$: Then $w = a_1 \dots a_k w_1 b w_2 b w_3 c_1 \dots c_l$ for some $a_1, \dots, a_k, b, c_1, \dots c_l \in A$, $w_1, w_2, w_3 \in A^*$. Due to the first pseudoidentity and the choice of w we have that $w' = a_1 \dots a_k w_1 b w_2 w_3 c_1 \dots c_l \in uB^*v \subseteq L$ for some $u, v \in A^*$, $B \subseteq A$, $|u| \leq k$, $|v| \leq l$. Then also $w \in uB^*v$ – a contradiction.

$m = 2$: Then $w = a_1 \dots a_k w_1 b w_2 b w_3 b w_4 c_1 \dots c_l$ for some $a_1, \dots, a_k, b, c_1, \dots,$
$c_l \in A$, $w_1, \dots, w_4 \in A^*$. As in case $m = 1$ we have $w' = a_1 \dots a_k w_1 b w_2 w_3 b w_4 c_1 \dots c_l \in uB_1^* B_2^* v \subseteq L$ for some $u, v \in A^*$, $B_1, B_2 \subseteq A$, $|u| \leq k$, $|v| \leq l$. Then also $w \in uB_1^* B_2^* v$ – a contradiction.

(iii) is clear.

(iv) : Obviously, if $k \leq k', l \leq l'$ then $\mathcal{L}(k, m, l)(A) \subseteq \mathcal{L}(k', m, l')(A)$. Let $T^\omega = T^\tau$. By (iii), $L \in \mathcal{L}(k, m, l)(A)$ for some k, l if and only if $L \in \mathcal{L}(r, m, r)(A)$. □

Notice that we did not need to substitute into pseudoidentities in (i) in a "multiliteral" way – literal substitutions were sufficient.

6 Pseudovarieties of Monoid Homomorphisms

We show here that the Eilenberg-type theorem for pseudovarieties of monoid homomorphism obtained in [8] is special cases of our present results.

Let $\mathcal{M}$ be the class of all finite monoids and let

$$\mathfrak{M} = \{ \kappa : A^* \twoheadrightarrow M \mid A \text{ is a finite set and } M \in \mathcal{M} \} .$$

In paper mentioned above the author considers a category $\mathbb{C}$ of homomorphism between finitely generated free monoids an so-called $\mathbb{C}$-pseudovarieties of monoid homomorphisms. In fact, our "$\mathbb{D}$-notions" are straightforward modifications of "$\mathbb{C}$-notions".

Realize that such category $\mathbb{C}$ determines a category $\mathbb{C}_\square$ of finitely generated free idempotent semirings by

$$\mathbb{C}_\square(B^\square, A^\square) = \{ p_\square \mid p \in \mathbb{C}(B^*, A^*) \}, \; p_\square(\{u_1, \dots, u_k\}) = \{p(u_1), \dots, p(u_k)\} .$$

Notice that each $(\phi : A^\square \twoheadrightarrow S) \in \mathfrak{S}$ determines a homomorphism $(\phi_* : A^* \twoheadrightarrow S_*) \in \mathfrak{M}$ by

$$\phi_*(u) = \phi(\{u\}), \; S_* = \{ \phi(\{u\}) \mid u \in A^* \} .$$

Also a class $\mathfrak{X} \subseteq \mathfrak{S}$ determines $\mathfrak{X}_* \subseteq \mathfrak{M}$. In particular, the semiring syntactic homomorphism $\phi_L : A^\square \twoheadrightarrow \mathsf{S}(L)$, $L \subseteq A^*$, determines a monoid homomorphism

$(\phi_L)_*$ which followed by an appropriate monoid isomorphism gives the monoid syntactic homomorphism κ_L for L. Moreover, a class $\mathfrak{P} \subseteq \mathfrak{M}$ determines $\mathfrak{P}_\square = \{\phi \in \mathfrak{X} \mid \psi_* \in \mathfrak{P}\}$.

For a monoid M, let $P(M)$ be the set of all subsets of M. It carries the structure of an idempotent semiring with respect to the obvious multiplication and the operation of union.

Each $(\phi : A^\square \twoheadrightarrow S) \in \mathfrak{S}$ determines

$$\phi_P : A^\square \twoheadrightarrow P(S_*), \quad \{u_1, \ldots, u_k\} \mapsto \{\phi(\{u_1\}), \ldots, \phi(\{u_k\})\} .$$

We formulate the following axiom concerning a class $\mathfrak{X} \subseteq \mathfrak{S}$:

$$(*) \quad \phi \in \mathfrak{X} \quad \text{implies} \quad \phi_P \in \mathfrak{X} .$$

It is quite straightforward to prove:

Theorem 5. *Let $\mathbb{C}$ be a category of finitely generated free monoids. The mappings*

$$\mathfrak{P} \mapsto \mathfrak{P}_\square \quad \textit{and} \quad \mathfrak{X} \mapsto \mathfrak{X}_*$$

are mutually inverse bijections between the class of all $\mathbb{C}$-pseudovarieties of monoid homomorphisms and the class of all $\mathbb{C}_\square$-pseudovarieties of idempotent semirings homomorphisms satisfying the axiom $(*)$. Moreover, for each $\mathbb{C}$-pseudovariety of monoid homomorphisms $\mathfrak{P}$ and any finite set A we have

$$\{ L \subseteq A^* \mid \kappa_L \in \mathfrak{P} \} = \{ L \subseteq A^* \mid \phi_L \in \mathfrak{P}_\square \} .$$

It is shown in [8] how to derive the classical Eilenberg's theorem from the theorem presented there. Quite similarly also the main result from [4] is a special case of the present results:

Theorem 6. *The mapping*

$$\mathfrak{X} \mapsto \{ S \in \mathcal{S} \mid \textit{there exists } (\phi : A^\square \twoheadrightarrow S) \in \mathfrak{X} \}$$

is a bijection between the class of all $\mathbb{D}_{all}$-pseudovarieties of semiring homomorphisms and the class of all pseudovarieties of idempotent semirings.

References

1. Almeida, J.; *Finite Semigroups and Universal Algebra,* World Scientific, 1994
2. Kunc, M.; Equational description of pseudovarieties of homomorphisms, RAIRO - Theoretical Informatics and Applications 37 (2003), 243–254
3. Pin, J.-E.; Syntactic semigroups, Chapter 10 in *Handbook of Formal Languages,* G. Rozenberg and A. Salomaa eds, Springer, 1997
4. Polák, L.; A classification of rational languages by semilattice-ordered monoids, Archivum Mathematicum, to appear, see also `http://www.math.muni.cz/~polak`
5. Polák, L.; Syntactic semiring of a language, *Proc. Mathematical Foundations of Computer Science 2001,* Springer Lecture Notes in Computer Science, Vol. 2136, 2001, pages 611–620

6. Polák, L.; Syntactic semiring and language equations, in *Proc. of the Seventh International Conference on Implementation and Application of Automata, Tours 2002,* Springer Lecture Notes in Computer Science, Vol. 2608, pages 182–193 (2003)
7. Polák, L.; Syntactic semiring and universal automaton, *Proc. Developments Language Theory, Szeged 2003,* Springer Lecture Notes in Computer Science, Vol. 2710, pp. 411–422 (2003)
8. Straubing, H.; On logical descriptions of regular languages, *Proc. LATIN 2002,* Springer Lecture Notes in Computer Science, Vol. 2286, 2002, pages 528–538

An Algebraic Generalization of ω-Regular Languages*

Zoltán Ésik[1] and Werner Kuich[2]

[1] University of Szeged
ze@inf.u-szeged.hu
[2] Technische Universität Wien
kuich@tuwien.ac.at

Abstract. This paper continues the algebraic theory of Ésik, Kuich [9] on semiring-semimodule pairs and quemirings that is applicable to languages that contain finite and infinite words. The main advantage is that we get rid of the idempotency assumption for the semimodule needed at several places in Ésik, Kuich [9].
Additionally, we consider linear systems as a generalization of rightlinear grammars. Moreover, we develop an algorithm that constructs, for a given finite automaton, an equivalent one without ε-moves.

1 Introduction

We extend the algebraic theory of Ésik, Kuich [9] on semiring-semimodule pairs and quemirings that is applicable to languages that contain finite and infinite words (see Perrin, Pin [13]). We consider the same finite automata as in Ésik, Kuich [9] but define their behavior in a different way. The difference to a similar result of Ésik, Kuich [9] is that we get rid of the assumption that the semimodule has to be idempotent. Hence, we can apply our results to power series with coefficients in, e.g., $\mathbb{N}^\infty$. The paper is an extended abstract of Ésik, Kuich [10].

The paper consists of this and two more sections. We assume the reader of this paper to be familiar with the definitions of Ésik, Kuich [8,9]. But to increase readibility, we repeat the necessary definitions concerning semiring-semimodule pairs and quemirings in this section.

In Section 2 we consider finite automata over generalized starquemirings. The main result of this paper is then a Kleene Theorem for these finite automata. We can prove that this Kleene Theorem is equivalent to the Kleene Theorem (Theorem 3.9) of Ésik, Kuich [9]. Since in our arguments we make use of only a few simple equations of the star and omega operations, the Kleene Theorem obtained has a very general scope. (See Ésik, Kuich [9] for applications involving the Kleene Theorems of Bouyer, Petit [2] and Droste, Kuske [5].) In Section 3 we consider linear systems and their solutions, and obtain some normal forms.

* Partially supported by Aktion Österreich-Ungarn, Wissenschafts- und Erziehungskooperation, Projekt 53ÖU1.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 648–659, 2004.
© Springer-Verlag Berlin Heidelberg 2004

These systems can be considered as a generalization of rightlinear grammars generating finite and infinite words. Moreover, we develop an algorithm that constructs, for a given finite automaton, an equivalent one without ε-moves. We then apply our results to formal power series, finite automata and rightlinear grammars.

Suppose that S is a semiring and V is a commutative monoid written additively. We call V a (left) *S-semimodule* if V is equipped with a (left) action

$$S \times V \to V$$
$$(s, v) \mapsto sv$$

subject to the following rules:

$$s(s'v) = (ss')v, \quad (s + s')v = sv + s'v, \quad s(v + v') = sv + sv',$$
$$1v = v, \quad 0v = 0, \quad s0 = 0,$$

for all $s, s' \in S$ and $v, v' \in V$. When V is an S-semimodule, we call (S, V) a *semiring-semimodule pair.*

Suppose that (S, V) is a semiring-semimodule pair such that S is a starsemiring and S and V are equipped with an omega operation $^\omega : S \to V$. Then we call (S, V) a *starsemiring-omegasemimodule pair.*

Ésik, Kuich [8] define a *complete semiring-semimodule pair* to be a semiring-semimodule pair (S, V) such that S is a complete semiring and V is a complete monoid, and an *infinite product operation* $\prod$ is defined, mapping infinite sequences over S to V. Moreover, the infinite sums and products have to satisfy certain conditions assuring that computations with these obey the usual laws.

Suppose that (S, V) is complete. Then we define

$$s^* = \sum_{i \geq 0} s^i, \qquad s^\omega = \prod_{i \geq 1} s,$$

for all $s \in S$. This turns (S, V) into a starsemiring-omegasemimodule pair. Observe that by Ésik, Kuich [8], $(\mathbb{B}\langle\langle A^* \rangle\rangle, \mathbb{B}\langle\langle A^\omega \rangle\rangle)$ and $(\mathbb{N}^\infty\langle\langle A^* \rangle\rangle, \mathbb{N}^\infty\langle\langle A^\omega \rangle\rangle)$ are complete semiring-semimodule pairs.

Following Bloom, Ésik [1] we define a matrix operation $^\omega : S^{n \times n} \to V^{n \times 1}$ on a starsemiring-omegasemimodule pair (S, V) as follows. When $n = 0$, M^ω is the unique element of V^0, and when $n = 1$, so that $M = (a)$, for some $a \in S$, $M^\omega = (a^\omega)$. Assume now that $n > 1$ and decompose M into blocks a, b, c, d with a of dimension 1×1 and d of dimension $(n-1) \times (n-1)$: $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then $M^\omega = \begin{pmatrix} (a + bd^*c)^\omega + (a + bd^*c)^* bd^\omega \\ (d + ca^*b)^\omega + (d + ca^*b)^* ca^\omega \end{pmatrix}$.

Moreover, we define matrix operations $^{\omega_k} : S^{n \times n} \to V^{n \times 1}$, $0 \leq k \leq n$, as follows. Assume that $M \in S^{n \times n}$ is decomposed into blocks a, b, c, d with a of dimension $k \times k$ and d of dimension $(n-k) \times (n-k)$: $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then $M^{\omega_k} = \begin{pmatrix} (a + bd^*c)^\omega \\ d^*c(a + bd^*c)^\omega \end{pmatrix}$. Observe that $M^{\omega_0} = 0$ and $M^{\omega_n} = M^\omega$.

A *Conway semiring* is a starsemiring satisfying

$$(a+b)^* = (a^*b)^*a^*\,, \quad (ab)^* = 1 + a(ba)^*b\,.$$

for all $a, b \in S$ (see Conway [4]). Following Bloom, Ésik [1], we call a starsemiring-omegasemimodule pair (S, V) a *Conway semiring-semimodule pair* if S is a Conway semiring and if the omega operation satisfies the following *sum-ω* and *product-ω equations:*

$$(a+b)^\omega = (a^*b)^\omega + (a^*b)^*a^\omega\,, (ab)^\omega = a(ba)^\omega\,,$$

for all $a, b \in S$. By Ésik, Kuich [8], each complete semiring-semimodule pair is a Conway semiring-semimodule pair.

Suppose that (S, V) is a semiring-semimodule pair and consider $T = S \times V$. Define on T the operations

$$(s,u)\cdot(s',v) = (ss', u+sv), \quad (s,u)+(s',v) = (s+s', u+v)$$

and constants $0 = (0,0)$ and $1 = (1,0)$. Equipped with these operations and constants, T satisfies the equations

$$(x+y)+z = x+(y+z), \quad x+y = y+x, \quad x+0 = x, \tag{1}$$
$$(x\cdot y)\cdot z = x\cdot(y\cdot z), \quad x\cdot 1 = x, \quad 1\cdot x = x, \tag{2}$$
$$(x+y)\cdot z = (x\cdot z)+(y\cdot z), \tag{3}$$
$$0\cdot x = 0. \tag{4}$$

Elgot[6] also defined the unary operation ¶ on T: $(s,u)¶ = (s,0)$. Thus, ¶ selects the "first component" of the pair (s,u), while multiplication with 0 on the right selects the "second component", for $(s,u)\cdot 0 = (0,u)$, for all $u \in V$. The new operation satisfies:

$$x¶\cdot(y+z) = (x¶\cdot y)+(x¶\cdot z), \tag{5}$$
$$x = x¶ + (x\cdot 0), \quad x¶\cdot 0 = 0, \tag{6}$$
$$(x+y)¶ = x¶ + y¶, \quad (x\cdot y)¶ = x¶\cdot y¶. \tag{7}$$

Note that when V is idempotent, also

$$x\cdot(y+z) = x\cdot y + x\cdot z$$

holds.

Elgot[6] defined a *quemiring* to be an algebraic structure T equipped with the above operations $\cdot, +, ¶$ and constants 0,1 satisfying the equations (1)–(4) and (5)–(7). It follows from the axioms that $x¶¶ = x¶$, for all x in a quemiring T. Moreover, $x¶ = x$ iff $x\cdot 0 = 0$.

When T is a quemiring, $S = T¶ = \{x¶ \mid x \in T\}$ is easily seen to be a semiring. Moreover, $V = T0 = \{x\cdot 0 \mid x \in T\}$ contains 0 and is closed under $+$, moreover, $sx \in V$ for all $s \in S$ and $x \in V$. Each $x \in T$ may be written in a

unique way as the sum of an element of $T\P$ and a sum of an element of $T0$, viz. $x = x\P + x \cdot 0$. Sometimes, we will identify $S \times \{0\}$ with S and $\{0\} \times V$ with V. It is shown in Elgot [6] that T is isomorphic to the quemiring $S \times V$ determined by the semiring-semimodule pair (S, V).

Suppose now that (S, V) is a starsemiring-omegasemimodule pair. Then we define on $T = S \times V$ a *generalized star operation:*

$$(s, v)^{\otimes} = (s^*, s^\omega + s^* v)$$

for all $(s, v) \in T$.

2 Finite Automata and a Kleene Theorem

In this section we consider finite automata over quemirings and prove a Kleene Theorem. Throughout this section, (S, V) denotes a starsemiring-omegasemimodule pair and T denotes the generalized starquemiring $S \times V$ (see Elgot [6] and Ésik, Kuich [8]). Moreover, S' denotes a subset of S and we assume $0^\omega = 0$.

A *finite S'-automaton* (*over the quemiring T*)

$$\mathfrak{A} = (n, I, M, P, k)$$

is given by

(i) a finite *set of states* $\{1, \ldots, n\}$, $n \geq 1$,
(ii) a *transition matrix* $M \in (S' \cup \{0, 1\})^{n \times n}$,
(iii) an *initial state vector* $I \in (S' \cup \{0, 1\})^{1 \times n}$,
(iv) a *final state vector* $P \in (S' \cup \{0, 1\})^{n \times 1}$,
(v) a *set of repeated states* $\{1, \ldots, k\}$, $k \geq 0$.

The *behavior* of $\mathfrak{A}$ is an element of T and is defined by

$$||\mathfrak{A}||_1 = IM^*P + IM^{\omega_k}.$$

If $\mathfrak{A} = (n, (i_1\ i_2), \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}, k)$, where $i_1 \in (S' \cup \{0,1\})^{1 \times k}$, $i_2 \in (S' \cup \{0,1\})^{1 \times (n-k)}$, $a \in (S' \cup \{0,1\})^{k \times k}$, $b \in (S' \cup \{0,1\})^{k \times (n-k)}$, $c \in (S' \cup \{0,1\})^{(n-k) \times k}$, $d \in (S' \cup \{0,1\})^{(n-k) \times (n-k)}$, $p_1 \in (S' \cup \{0,1\})^{k \times 1}$, $p_2 \in (S' \cup \{0,1\})^{(n-k) \times 1}$, we write also

$$\mathfrak{A} = (n; i_1, i_2; a, b, c, d; p_1, p_2; k).$$

As usual, such a finite S'-automaton can be depicted as a *directed graph.* The nodes of the graph correspond to the states of the automaton. A node (state) q with $I_q \neq 0$ (resp. $P_q \neq 0$) is called *initial* (resp. *final*). The edges (q_1, q_2) of the directed graph correspond to the transitions from state q_1 to state q_2 unequal to 0 and are labeled by $M_{q_1 q_2}$. The path consisting of edges $(q_0, q_1), (q_1, q_2), \ldots, (q_{n-1}, q_n)$ has by definition the *weight* $M_{q_0 q_1} M_{q_1 q_2} \ldots M_{q_{n-1} q_n}$.

Let (S, V) be now a complete semiring-semimodule pair and consider a finite S'-automaton $\mathfrak{A} = (n; i_1, i_2; a, b, c, d; p_1, p_2; k)$, the matrix $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with

$$M^{\omega_k} = \begin{pmatrix} (a^*bd^*c)^\omega + (a^*bd^*c)^*a^\omega \\ cd^*(a^*bd^*c)^\omega + cd^*(a^*bd^*c)^*a^\omega \end{pmatrix}$$

and the directed graph of $\mathfrak{A}$. In the first summand of the entries of M^{ω_k}, the blocks b and c occur infinitely often, i. e., the i-the row of the first summand is the sum of the weights of all infinite paths starting in state i and passing infinitely often through the repeated states in $\{1, \ldots, k\}$ and the nonrepeated states in $\{k+1, \ldots, n\}$. In the second summand of the entries of M^{ω_k}, the block a occurs infinitely often and the blocks b and c occur only finitely often, i.e., the i-th row of the second summand is the sum of the weights of all infinite paths starting in state i and passing infinitely often through the repeated states in $\{1, \ldots, k\}$ and only finitely often through the nonrepeated states in $\{k+1, \ldots, n\}$. Hence, the i-th row of the first and the second summand of the entries of M^{ω_k} are sums of weights of disjoint sets of infinite paths. But by Ésik, Kuich [9], each weight of an infinite path is counted at least once. Hence, we have the following result: *If (S, V) is a complete semiring-semimodule pair and $\mathfrak{A}$ is a finite S'-automaton, then $||\mathfrak{A}||$ is the sum of the weights of all finite paths from an initial state to a final state multiplied properly by the initial and final weights of these states and of the weights of all infinite paths starting at an initial state, passing infinitely often through a repeated state, and properly multiplied by the initial weight of this initial state.*

In Ésik, Kuich [9], the behavior of a finite S'-automaton $\mathfrak{A}$ was defined in a different way. But we can show that, in case (S, V) is a Conway semiring-semimodule pair and V is an idempotent semimodule, both behaviors coincide.

A subsemiring of S is *rationally closed* if it is closed under the operations $+, \cdot, ^*$. A subquemiring of $S \times V$ is *ω-rationally closed* if it is closed under the operations $+, \cdot, \P, ^\otimes$. By definition, $\mathfrak{Rat}(S')$ (resp. ω-$\mathfrak{Rat}(S')$) is the smallest rationally (resp. ω-rationally) closed semiring (resp. quemiring) that contains S'.

For the remainder of this section we assume that (S, V) is a starsemiring-omegasemimodule pair such that S is a Conway semiring and $0^\omega = 0$. We now will prove a Kleene Theorem: Let $a \in S \times V$. Then $a \in \omega$-$\mathfrak{Rat}(S')$ iff a is the behavior of a finite S'-automaton. To achieve this result we need a few theorems and corollaries.

Let $\mathfrak{A} = (n, I, M, P, k)$ be a finite S'-automaton. It is called *normalized* if (i) $n \geq 2$ and $k \leq n-2$; (ii) $I_{n-1} = 1$ and $I_j = 0$ for $j \neq n-1$; (iii) $P_n = 1$ and $P_j = 0$ for $j \neq n$; (iv) $M_{i,n-1} = 0$ and $M_{n,i} = 0$ for all $1 \leq i \leq n$. Two finite S'-automata $\mathfrak{A}$ and $\mathfrak{A}'$ are *equivalent* if $||\mathfrak{A}||_1 = ||\mathfrak{A}'||_1$.

Theorem 2.1. *Each finite S'-automaton $\mathfrak{A} = (n, I, M, P, k)$ is equivalent to a normalized finite S'-automaton $\mathfrak{A}' = (n+2, I', M', P', k)$.*

Lemma 2.2. *If* $\mathfrak{A} = (n; i_1, i_2; a, b, c, d; p_1, p_2; k)$ *is a finite* S'*-automaton then*

$$||\mathfrak{A}||_1 = i_1(a + bd^*c)^*(p_1 + bd^*p_2) + i_2d^*c(a + bd^*c)^*(p_1 + bd^*p_2) + i_2d^*p_2 + i_1(a + bd^*c)^\omega + i_2d^*c(a + bd^*c)^\omega .$$

Let $\mathfrak{A} = (n; i_1, i_2; a, b, c, d; f, g; m)$ and $\mathfrak{A}' = (n'; h, i; a', b', c', d'; p_1, p_2; k)$ be finite S'-automata. Then we define the finite S'-automata $\mathfrak{A} + \mathfrak{A}'$ and $\mathfrak{A} \cdot \mathfrak{A}'$ to be

$$\mathfrak{A} + \mathfrak{A}' = (n + n'; (i_1\ h), (i_2\ i); \begin{pmatrix} a & 0 \\ 0 & a' \end{pmatrix}, \begin{pmatrix} b & 0 \\ 0 & b' \end{pmatrix}, \begin{pmatrix} c & 0 \\ 0 & c' \end{pmatrix}, \begin{pmatrix} d & 0 \\ 0 & d' \end{pmatrix}; \begin{pmatrix} f \\ p_1 \end{pmatrix}, \begin{pmatrix} g \\ p_2 \end{pmatrix}, m + k)$$

and

$$\mathfrak{A} \cdot \mathfrak{A}' = (n + n'; (i_1\ 0), (i_2\ 0); \begin{pmatrix} a & fh \\ 0 & a' \end{pmatrix}, \begin{pmatrix} b & fi \\ 0 & b' \end{pmatrix}, \begin{pmatrix} c & gh \\ 0 & c' \end{pmatrix}, \begin{pmatrix} d & gi \\ 0 & d' \end{pmatrix}; \begin{pmatrix} 0 \\ p_1 \end{pmatrix}, \begin{pmatrix} 0 \\ p_2 \end{pmatrix}, m + k) .$$

For the definition of $\mathfrak{A} \cdot \mathfrak{A}'$ we assume that either $\begin{pmatrix} f \\ g \end{pmatrix} (h\ i) \in (S' \cup \{0, 1\})^{n \times n'}$ or $\mathfrak{A}'$ is normalized. Observe that the definitions of $\mathfrak{A} + \mathfrak{A}'$ and $\mathfrak{A} \cdot \mathfrak{A}'$ (and of $\mathfrak{A}^\otimes$ which is defined below) are the usual ones except that certain rows and columns are permuted. These permutations are needed since the set of repeated states of a finite S'-automaton is always a set $\{1, \ldots, k\}$. (See also Theorems 3.3–3.7 of Ésik, Kuich [9].)

Theorem 2.3. *Let* $\mathfrak{A}$ *and* $\mathfrak{A}'$ *be finite* S'*-automata. Then* $||\mathfrak{A} + \mathfrak{A}'||_1 = ||\mathfrak{A}||_1 + ||\mathfrak{A}'||_1$ *and* $||\mathfrak{A} \cdot \mathfrak{A}'||_1 = ||\mathfrak{A}||_1 \cdot ||\mathfrak{A}'||_1$.

Proof. We prove only $||\mathfrak{A} \cdot \mathfrak{A}'||_1 = ||\mathfrak{A}||_1 \cdot ||\mathfrak{A}'||_1$ and first compute $||\mathfrak{A} \cdot \mathfrak{A}'||_1 \cdot 0$. The transition matrix of $\mathfrak{A} \cdot \mathfrak{A}'$ is given by

$$M = \begin{pmatrix} a & fh & b & fi \\ 0 & a' & 0 & b' \\ c & gh & d & gi \\ 0 & c' & 0 & d' \end{pmatrix} .$$

We now compute the first $m + k$ entries of $M^{\omega_{m+k}}$. This column vector of dimension $m + k$ is given by

$$\left(\begin{pmatrix} a & fh \\ 0 & a' \end{pmatrix} + \begin{pmatrix} b & fi \\ 0 & b' \end{pmatrix} \begin{pmatrix} d^* & d^*gid'^* \\ 0 & d'^* \end{pmatrix} \begin{pmatrix} c & gh \\ 0 & c' \end{pmatrix} \right)^\omega =$$

$$\begin{pmatrix} a + bd^*c & (f + bd^*g)(h + id'^*c') \\ 0 & a' + b'd'^*c' \end{pmatrix}^\omega =$$

$$\begin{pmatrix} (a + bd^*c)^\omega + (a + bd^*c)^*(f + bd^*g)(h + id'^*c')(a' + b'd'^*c')^\omega \\ (a' + b'd'^*c')^\omega \end{pmatrix} .$$

The last $n+n'-(m+k)$ entries of $M^{\omega_{m+k}}$ are given by the product of

$$\begin{pmatrix} d & gi \\ 0 & d' \end{pmatrix}^* \begin{pmatrix} c & gh \\ 0 & c' \end{pmatrix} = \begin{pmatrix} d^*c & d^*g(h+id'^*c') \\ 0 & d'^*c' \end{pmatrix}$$

with the column vector computed above. Hence, we obtain

$$\begin{aligned} &||\mathfrak{A}\cdot\mathfrak{A}'||_1\cdot 0 = (i_1\ 0\ i_2\ 0)M^{\omega_{m+k}} = \\ &i_1(a+bd^*c)^\omega + i_1(a+bd^*c)^*(f+bd^*g)(h+id'^*c)(a'+b'd'^*c')^\omega + \\ &i_2d^*c(a+bd^*c)^\omega + i_2d^*c(a+bd^*c)^*(f+bd^*g)(h+id'^*c')(a'+b'd'^*c')^\omega + \\ &i_2d^*g(h+id'^*c')(a'+b'd'^*c')^\omega . \end{aligned}$$

On the other side, we obtain by Lemma 2.2

$$\begin{aligned} &||\mathfrak{A}||_1\cdot||\mathfrak{A}'||_1\cdot 0 = ||\mathfrak{A}||_1\cdot 0 + ||\mathfrak{A}||_1\P\cdot||\mathfrak{A}'||_1\cdot 0 = \\ &i_1(a+bd^*c)^\omega + i_2d^*c(a+bd^*c)^\omega + (i_1(a+bd^*c)^*(f+bd^*g)+ \\ &i_2d^*c(a+bd^*c)^*(f+bd^*g)+i_2d^*g)(h+id'^*c')(a'+b'd'^*c')^\omega . \end{aligned}$$

Hence, $||\mathfrak{A}\cdot\mathfrak{A}'||_1\cdot 0 = ||\mathfrak{A}||_1\cdot||\mathfrak{A}'||_1\cdot 0$.

We now compute $||\mathfrak{A}\cdot\mathfrak{A}'||_1\P$. If, in the transition matrix M of $\mathfrak{A}\cdot\mathfrak{A}'$ we commute the $m+1,\ldots,m+k$ row and column with the $m+k+1,\ldots,n+k$ row and column, and do the same with the initial and final vector we obtain by the permutation identity (see Conway [4], Ésik, Kuich [7])

$$||\mathfrak{A}\cdot\mathfrak{A}'||_1\P = (i_1\ i_2\ 0\ 0)\begin{pmatrix} a & b & fh & fi \\ c & d & gh & gi \\ 0 & 0 & a' & b' \\ 0 & 0 & c' & d' \end{pmatrix} =$$

$$(i_1\ i_2)\begin{pmatrix} a & b \\ c & d \end{pmatrix}^*\begin{pmatrix} f \\ g \end{pmatrix}(h\ i)\begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}^*\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} =$$

$$||\mathfrak{A}||_1\P\cdot||\mathfrak{A}'||_1\P = ||\mathfrak{A}||_1\cdot||\mathfrak{A}'||_1\P .$$

Hence, $||\mathfrak{A}\cdot\mathfrak{A}'||_1 = ||\mathfrak{A}||_1\cdot||\mathfrak{A}'||_1$. □

Let $\mathfrak{A}=(n;h,i;a,b,c,d;f,g;k)$ be a finite S'-automaton and write $I=(h\ i)$, $M=\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $P=\begin{pmatrix} f \\ g \end{pmatrix}$. Then we define the finite S'-automaton $\mathfrak{A}^\otimes$ to be

$$\begin{aligned} \mathfrak{A}^\otimes = (1+n+n;(1\ 0),(0\ 0);&\begin{pmatrix} 0 & h \\ 0 & a \end{pmatrix},\begin{pmatrix} i & I \\ b & 0 \end{pmatrix}, \\ &\begin{pmatrix} 0 & c \\ P & 0 \end{pmatrix},\begin{pmatrix} d & 0 \\ 0 & M \end{pmatrix};\begin{pmatrix} 1 \\ 0 \end{pmatrix},\begin{pmatrix} 0 \\ 0 \end{pmatrix};1+k). \end{aligned}$$

Theorem 2.4. *Let $\mathfrak{A}$ be a finite S'-automaton. Then $||\mathfrak{A}^\otimes||_1 = ||\mathfrak{A}||_1^\otimes$.*

Proof. Let $\mathfrak{A}$ be defined as above. Let

$$M' = \begin{pmatrix} 0 & h & i & I \\ 0 & a & b & 0 \\ 0 & c & d & 0 \\ P & 0 & 0 & M \end{pmatrix} .$$

We first compute $||\mathfrak{A}^{\otimes}||_1\P$. Observe that M' can be written as

$$M' = \begin{pmatrix} 0 & I & I \\ 0 & M & 0 \\ P & 0 & M \end{pmatrix}$$

and that $||\mathfrak{A}^{\otimes}||_1\P = (M'^*)_{11}$. We obtain

$$(M'^*)_{11} = \left((I\ I)\begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}^* \begin{pmatrix} 0 \\ P \end{pmatrix}\right)^* = (IM^*P)^* = (||\mathfrak{A}||_1\P)^* = ||\mathfrak{A}||_1^{\otimes}\P.$$

We now compute the first $1+k$ entries of $M'^{\omega_{1+k}}$. This column vector of dimension $1+k$ is given by

$$\left(\begin{pmatrix} 0 & h \\ 0 & a \end{pmatrix} + \begin{pmatrix} i & I \\ b & 0 \end{pmatrix}\begin{pmatrix} d & 0 \\ 0 & M \end{pmatrix}^* \begin{pmatrix} 0 & c \\ P & 0 \end{pmatrix}\right)^{\omega} =$$

$$\begin{pmatrix} IM^*P & h + id^*c \\ 0 & a + bd^*c \end{pmatrix}^{\omega}.$$

Hence, $||\mathfrak{A}^{\otimes}||_1 \cdot 0 = (M^{\omega_{1+k}})_1 = (IM^*P)^{\omega} + (IM^*P)^*(h + id^*c)(a + bd^*c)^{\omega}$. By definition, $||\mathfrak{A}||_1^{\otimes}\cdot 0 = (||\mathfrak{A}||_1\P)^{\omega} + (||\mathfrak{A}||_1\P)^*||\mathfrak{A}||_1 \cdot 0$. Thus $||\mathfrak{A}||_1^{\otimes}\cdot 0 = (IM^*P)^{\omega} + (IM^*P)^*(h + id^*c)(a + bd^*c)^{\omega} = ||\mathfrak{A}^{\otimes}||_1 \cdot 0$ and we obtain $||\mathfrak{A}^{\otimes}||_1 = ||\mathfrak{A}||_1^{\otimes}$. □

Theorem 2.5. *Let $\mathfrak{A} = (n, I, M, P, k)$ be a finite S'-automaton. Then there exists a finite S'-automaton $\mathfrak{A}\P$ such that $||\mathfrak{A}\P||_1 = ||\mathfrak{A}||_1\P$.*

Proof. $||\mathfrak{A}||_1\P = (n, I, M, P, 0)$. □

Theorem 2.6. *Let $a \in S' \cup \{0, 1\}$. Then there exists a finite S'-automaton $\mathfrak{A}_a$ such that $||\mathfrak{A}_a||_1 = a$.*

Proof. Let $\mathfrak{A}_a = \left(2, \begin{pmatrix} 0 & a \\ 0 & 0 \end{pmatrix}, (1\ 0), \begin{pmatrix} 0 \\ 1 \end{pmatrix}, 0\right)$.

Then $||\mathfrak{A}_a||_1 = (1\ 0)\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = a$. □

Corollary 2.7. *The behaviors of finite S'-automata form an ω-rationally closed quemiring that contains S'.*

Theorem 2.8 (Kleene Theorem). *Let (S, V) be a starsemiring-omegasemimodule pair, where S is a Conway semiring and $0^{\omega} = 0$. Then the following statements are equivalent to the statements of Theorem 3.9, Theorem 3.10 and Corollary 3.11 of Ésik, Kuich [9] for $(s, v) \in S \times V$:*

(i) $(s, v) = ||\mathfrak{A}||_1$, *where $\mathfrak{A}$ is a finite S'-automaton,*
(ii) $(s, v) \in \omega\text{-}\mathfrak{Rat}(S')$.

3 Linear Systems

An *S'-linear system (with variables $z_1, \ldots, z_n$ over the quemiring $S \times V$)* is given by

$$My + P = y,$$

where $M \in (S' \cup \{0,1\})^{n \times n}$, $P \in (S' \cup \{0,1\})^{n \times 1}$ and $y = (z_1, \ldots, z_n)^T$. A column vector $\sigma \in (S \times V)^{n \times 1}$ is called *solution* to this S'-linear system if $M\sigma + P = \sigma$.

Theorem 3.1. *Let (S, V) be a Conway semiring-semimodule pair. Then, for $0 \leq k \leq n$, $M^{\omega_k} + M^* P$ is a solution of the S'-linear system $My + P = y$.*

For the remainder of this section we consider a complete semiring-semimodule pair $(S\langle\langle A^* \rangle\rangle, S\langle\langle A^\omega \rangle\rangle)$ (e.g., that of Example 2.2 of Ésik, Kuich [9]). Here S is a complete semiring and A is an alphabet.

Consider an S'-linear system $My + P = y$ as defined above for $S' = S\langle A \cup \varepsilon \rangle$. Write this system in the form

$$z_i = \sum_{1 \leq j \leq n} \sum_{x \in A \cup \varepsilon} (M_{ij}, x) x z_j + \sum_{x \in A \cup \{\varepsilon\}} (P_i, x) x, \quad 1 \leq i \leq n.$$

To this system we associate the rightlinear grammars $G_i = (\{z_1, \ldots, z_n\}, A, R, z_i)$, $1 \leq i \leq n$, with weights in the semiring S, where $R = \{z_i \to (M_{ij}, x) x z_j \mid 1 \leq j \leq n,\ x \in A \cup \{\varepsilon\}\} \cup \{z_i \to (P_i, x) x \mid x \in A \cup \{\varepsilon\}\}$. Here (M_{ij}, x) and (P_i, x) are the weights of the productions $z_i \to x z_j$ and $z_i \to x$, respectively. Furthermore, let $\mathfrak{A}_i^k = (n, e_i, M, P, k)$ be finite S'-automata, $1 \leq i \leq n$, for some fixed $k \in \{0, \ldots, n\}$, where e_i is the i-th row vector of unity.

Consider now a finite derivation with respect to G_i:

$$z_i \Rightarrow (M_{i,i_1}, x_1) x_1 z_{i_1} \Rightarrow \cdots \Rightarrow (M_{i,i_1}, x_1) \ldots (M_{i_{m-1},i_m}, x_m) x_1 \ldots x_m z_{i_m} \Rightarrow (M_{i,i_1}, x_1) \ldots (M_{i_{m-1},i_m}, x_m)(P_{i_m}, x_{m+1}) x_1 \ldots x_m x_{m+1}$$

generating the word $x_1 \ldots x_m x_{m+1}$ with weight $(M_{i,i_1}, x_1) \ldots (M_{i_{m-1},i_m}, x_m) \cdot (P_{i_m}, x_{m+1})$. This finite derivation corresponds to the following finite path in the directed graph of $\mathfrak{A}_i^k$: $(z_i, x_1, z_{i_1}), \ldots, (z_{i_{m-1}}, x_m, z_{i_m})$ with weight $(M_{i,i_1}, x_1) \ldots (M_{i_{m-1},i_m}, x_m)$, initial weight 1 and final weight $(P_{i_m}, x_{m+1}) x_{m+1}$. Consider now an infinite derivation with respect to G_i:

$$z_i \Rightarrow (M_{i,i_1}, x_1) x_1 z_{i_1} \Rightarrow \cdots \Rightarrow (M_{i,i_1}, x_1) \ldots (M_{i_{m-1},i_m}, x_m) x_1 \ldots x_m z_{i_m} \Rightarrow \ldots$$

generating $x_1 x_2 \ldots x_m \ldots$ with weight $(M_{i,i_1}, x_1) \ldots (M_{i_{m-1},i_m}, x_m) \ldots$. This infinite derivation corresponds to the following infinite path in the directed graph of $\mathfrak{A}_i^k$:

$$(z_i, x_1, z_{i_1}), \ldots, (z_{i_{m-1}}, x_m, z_{i_m}), \ldots$$

with weight $(M_{i,i_1}, x_1) \ldots (M_{i_{m-1},i_m}, x_m) \ldots$ and initial weight 1.

Hence, we obtain, by the first result of this section, the following result for G_i and $\mathfrak{A}_i^k$ as defined above: *If S is a complete semiring and $1 \leq i \leq n$, $0 \leq k \leq n$,*

then, for $w \in A^*$, $(||\mathfrak{A}_i^k||_1, w) = ((M^*P)_i, w)$ *is the sum of the weights of all finite derivations of* w *with respect to* G_i*; and for* $w \in A^\omega$, $(||\mathfrak{A}_i^k||_1, w) = ((M^{\omega_k})_i, w)$ *is the sum of the weights of all infinite derivations of* w *with respect to* G_i *such that at least one of the variables of* $\{z_1, \ldots, z_k\}$ *appears infinitely often in these infinite derivations.*

In particular, if $S = \mathbb{N}^\infty$ and $M_{ij}, P_i \in \{0,1\}$, $1 \leq i,j \leq n$, then we get the following result: *For* $w \in A^*$, $(||\mathfrak{A}_i^k||_1, w) = ((M^*P)_i, w)$ *is the number of finite derivations of* w *with respect to* G_i*; and for* $w \in A^\omega$, $(||\mathfrak{A}_i^k||_1, w) = ((M^{\omega_k})_i, w)$ *is the number of all infinite derivations of* w *with respect to* G_i *such that at least one of the variables of* $\{z_1, \ldots, z_k\}$ *appears infinitely often in these infinite derivations.*

We now want to delete ε-moves in finite $S\langle A \cup \varepsilon\rangle$-automata without changing their behavior.

Theorem 3.2. *Let* $(S\langle\langle A^*\rangle\rangle, S\langle\langle A^\omega\rangle\rangle)$ *be a Conway semiring-semimodule pair, where* $\varepsilon^\omega = 0$*, and consider a finite* $S\langle A \cup \varepsilon\rangle$*-automaton* $\mathfrak{A} = (n, I, M, P, k)$*. Then there exists an finite* $S\langle A \cup \varepsilon\rangle$*-automaton* $\mathfrak{A}' = (n, I', M', P', k)$ *with* $||\mathfrak{A}'||_1 = ||\mathfrak{A}||_1$ *satisfying the following conditions: (i)* $M' \in (S\langle A\rangle)^{n\times n}$*, (ii)* $I' \in (S\langle\varepsilon\rangle)^{1\times n}$*, (iii)* $P' \in (S\langle\varepsilon\rangle)^{n\times 1}$*.*

Proof. Without loss of generality we assume by Theorem 2.1 that $I \in (S\langle\varepsilon\rangle)^{1\times n}$ and $P \in (S\langle\varepsilon\rangle)^{n\times 1}$. Let $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, where a is $k \times k$ and d is $(n-k)\times(n-k)$. Let $a = a_0 + a_1$, $b = b_0 + b_1$, $c = c_0 + c_1$, $d = d_0 + d_1$, such that the supports of the entries of a_0, b_0, c_0, d_0 (resp. a_1, b_1, c_1, d_1) are subsets of $\{\varepsilon\}$ (resp. A). Since $\varepsilon^\omega = 0$, we obtain $(a_0 + b_0 d_0^* c_0)^\omega = 0$.

Define the matrices M_{01}, M_{02} and M_1 to be $M_{01} = \begin{pmatrix} 0 & b_0 \\ 0 & d_0 \end{pmatrix}$, $M_{02} = \begin{pmatrix} a_0 & 0 \\ c_0 & 0 \end{pmatrix}$ and $M_1 = \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}$. We now specify the finite $S\langle A \cup \varepsilon\rangle$-automaton $\mathfrak{A}'$: $I' = I(M_{01}^* M_{02})^*$, $M' = M_{01}^* M_1 (M_{01}^* M_{02})^*$ and $P' = M_{01}^* P$. The behavior of $\mathfrak{A}'$ is then given by

$$
\begin{aligned}
&||\mathfrak{A}'||_1 = I'M'^* P' + I'M'^{\omega_k} = \\
&I(M_{01}^* M_{02})^* (M_{01}^* M_1 (M_{01}^* M_{02})^*)^* M_{01}^* P + \\
&I(M_{01}^* M_{02})^* (M_{01}^* M_1 (M_{01}^* M_{02})^*)^{\omega_k} = \\
&I(M_{01} + M_{02} + M_1)^* P + I(M_{01} + M_{02} + M_1)^{\omega_k} = \\
&IM^* P + IM^{\omega_k} = ||\mathfrak{A}||_1 .
\end{aligned}
$$

Here we have applied Theorem 2.10 of Ésik, Kuich [10] in the third equality. □

Theorem 3.3. *Let* $(S\langle\langle A^*\rangle\rangle, S\langle\langle A^\omega\rangle\rangle)$ *be a Conway semiring-semimodule pair, where* $\varepsilon^\omega = 0$*, and consider a finite* $S\langle A \cup \varepsilon\rangle$*-automaton* $\mathfrak{A} = (n, I, M, P, k)$*. Then there exists an finite* $S\langle A \cup \varepsilon\rangle$*-automaton* $\mathfrak{A}' = (n+1, I', M', P', k)$ *with* $||\mathfrak{A}'||_1 = ||\mathfrak{A}||_1$ *satisfying the following conditions: (i)* $M' \in (S\langle A\rangle)^{(n+1)\times(n+1)}$*, (ii)* $I'_j = 0$*,* $1 \leq j \leq n$*, and* $I'_{n+1} = \varepsilon$*, (iii)* $P' \in (S\langle\varepsilon\rangle)^{(n+1)\times 1}$*.*

Proof. We assume that $\mathfrak{A}$ satisfies the conditions of Theorem 3.2. We specify $\mathfrak{A}'$ by $I' = (0\ \varepsilon)$, $M' = \begin{pmatrix} M & 0 \\ IM & 0 \end{pmatrix}$ and $P' = \begin{pmatrix} P \\ IP \end{pmatrix}$. We compute $M'^* = \begin{pmatrix} M^* & 0 \\ IMM^* & 1 \end{pmatrix}$ and, for $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, $I = (i_1\ i_2)$,

$$IM'^{\omega_k} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ i_1a + i_2c & i_1b + i_2d & 0 \end{pmatrix}^{\omega_k} =$$

$$\begin{pmatrix} (a + bd^*c)^\omega \\ d^*c(a + bd^*c)^\omega \\ (i_1(a + bd^*c) + i_2d^*c)(a + bd^*c)^\omega \end{pmatrix} = \begin{pmatrix} M^{\omega_k} \\ IM^{\omega_k} \end{pmatrix}.$$

Hence, $||\mathfrak{A}'||_1 = IMM^*P + IP + IM^{\omega_k} = ||\mathfrak{A}||_1$. □

In the case of the Boolean semiring, the finite $\mathbb{B}\langle\langle \Sigma \cup \varepsilon \rangle\rangle$-automata of Theorem 3.3 are nothing else than the finite automata introduced by Büchi [3].

In the case of the semiring $\mathbb{N}^\infty$ we get the following result: *The constructions of Theorems 3.2 and 3.3 do not change, for $w \in \Sigma^*$ (resp. for $w \in \Sigma^\omega$), in the digraphs of the finite automata, the number of finite paths with label w from an initial state to a final state (resp. the number of infinite paths with label w starting in an initial state and passing infinitely often through a repeated state).*

Given a rightlinear grammar $G_i = (\{z_1, \ldots, z_n\}, A, R, z_i)$, $1 \leq i \leq n$, as above, and $k \in \{0, \ldots, n\}$, $L(G_i)_k$ is defined to be the weighted language

$$L(G_i)_k = \{((M^*P)_i, w)w \mid w \in A^*\} \cup \{((M^{\omega_k})_i, w)w \mid w \in A^\omega\}.$$

The next theorem, Theorem 3.4, shows that such weighted languages can be generated by rightlinear grammars with weights in the semiring S which have only two types of productions: $z_i \to axz_j$ and $z_i \to a\varepsilon$, where $a \in S$ and $x \in A$. Hence, in such rightlinear grammars there are no productions $z_i \to az_j$. Corollary 3.5 shows then, that the two types of productions can be chosen as $z_i \to axz_j$ and $z_i \to ax$, where $a \in S$ and $x \in A$. (Of course, ε is no longer derived.)

Theorem 3.4. *Let $(S\langle\langle A^* \rangle\rangle, S\langle\langle A^\omega \rangle\rangle)$ be a Conway semiring-semimodule pair, where $\varepsilon^\omega = 0$, consider an $S\langle\langle A \cup \varepsilon \rangle\rangle$-linear system $My + P = y$, where $M \in (S\langle\langle A \cup \varepsilon \rangle\rangle)^{n \times n}$, $P \in (S\langle\langle A \cup \varepsilon \rangle\rangle)^{n \times 1}$, and $y = (z_1, \ldots, z_n)^T$ and let $i \in \{1, \ldots, n\}$. Then there exists an $S\langle\langle A \cup \varepsilon \rangle\rangle$-linear system $M'y' + P' = y'$, where $M' \in (S\langle\langle A \rangle\rangle)^{(n+1)\times(n+1)}$, $P' \in (S\langle\langle \varepsilon \rangle\rangle)^{(n+1)\times 1}$, and $y' = \begin{pmatrix} y \\ z_{n+1} \end{pmatrix}$ such that $(M'^{\omega_k} + M'^*P')_{n+1} = (M^{\omega_k} + M^*P)_i$.*

Proof. Consider the finite $S\langle\langle A \cup \varepsilon \rangle\rangle$-automaton $\mathfrak{A}_i^k = (n, e_i, M, P, k)$, whose behavior is $||\mathfrak{A}_i^k||_1 = (M^*P)_i + M^{\omega_k}$. Starting with $\mathfrak{A}_i^k$, perform the constructions of Theorems 3.2 and 3.3. This yields a finite $S\langle\langle A \cup \varepsilon \rangle\rangle$-automaton $\mathfrak{A}' = (n+1, e_{n+1}, M', P', k)$ with behavior $||\mathfrak{A}'||_1 = (M'^*P')_{n+1} + (M'^{\omega_k})_{n+1} = ||\mathfrak{A}_i^k||_1$. □

Corollary 3.5. *Let $(S\langle\langle A^*\rangle\rangle, S\langle\langle A^\omega\rangle\rangle)$ be a Conway semiring-semimodule pair, where $\varepsilon^\omega = 0$, consider an $S\langle A \cup \varepsilon\rangle$-linear system $My + P = y$, where $M \in (S\langle A \cup \varepsilon\rangle)^{n\times n}$, $P \in (S\langle A \cup \varepsilon\rangle)^{n\times 1}$, and $y = (z_1, \ldots, z_n)^T$ and let $i \in \{1, \ldots, n\}$. Then there exists an $S\langle A \cup \varepsilon\rangle$-linear system $M'y' + P' = y'$, where $M' \in (S\langle A\rangle)^{(n+1)\times(n+1)}$, $P' \in (S\langle A\rangle)^{(n+1)\times 1}$, and $y' = \begin{pmatrix} y \\ z_{n+1} \end{pmatrix}$ such that $(M'^{\omega_k} + M'^* P')_{n+1} = (M^{\omega_k} + MM^*P)_i$.*

Proof. Let $M'y' + P'' = y'$ be the $S\langle A \cup \varepsilon\rangle$-linear system constructed according to Theorem 3.4 from $My + P = y$. Consider the $S\langle A \cup \varepsilon\rangle$-linear system $M'y' + P' = y'$, where $P' = M'P''$. Then $(M'^{\omega_k} + M'^* P')_{n+1} = (M'^{\omega_k} + M'^* M' P'')_{n+1} = (M^{\omega_k} + MM^*P)_i$. □

If we consider $\mathbb{N}^\infty\langle\Sigma \cup \varepsilon\rangle$-linear systems we obtain the following statement about the derivations with respect to the rightlinear grammars G_i defined above: *The constructions of Theorem 3.4 and Corollary 3.5 do not change, for $w \in \Sigma^+$ (resp. for $w \in \Sigma^\omega$), the number of finite derivations of w with respect to G_i (resp. the number of infinite derivations of w with respect to G_i such that at least one of the variables of $\{z_1, \ldots, z_n\}$ appears infinitely often in these infinite derivations).*

Hence, the constructions transform unambiguous grammars into unambiguous grammars.

References

1. Bloom, S. L., Ésik, Z.: Iteration Theories. EATCS Monographs on Theoretical Computer Science. Springer, 1993.
2. Bouyer, P, Petit, A.: A Kleene/Büchi-like theorem for clock languages. J. of Automata, Languages and Combinatorics 7(2002) 167–186.
3. Büchi, J. R.: On a decision method in restricted second order arithmetic. In: Proc. Int. Congr. Logic, Methodology and Philosophy of Science, 1960. Stanford University Press, 1962, pp. 1–11.
4. Conway, J. H.: Regular Algebra and Finite Machines. Chapman & Hall, 1971.
5. Droste, M., Kuske, D.: Skew and infinitary formal power series. Technical Report 2002–38, Department of Mathematics and Computer Science, University of Leicester.
6. Elgot, C.: Matricial theories. J. Algebra 42(1976) 391–422.
7. Ésik, Z., Kuich, W.: Inductive *-semirings. To appear in Theoretical Computer Science.
8. Ésik, Z., Kuich, W.: On iteration semiring-semimodule pairs. To appear.
9. Ésik, Z., Kuich, W.: A semiring-semimodule generalization of ω-regular languages I. Technical Report, Technische Universität Wien, 2003.
10. Ésik, Z., Kuich, W.: A semiring-semimodule generalization of ω-regular languages II. Technical Report, Technische Universität Wien, 2003.
11. Kuich, W., Salomaa, A.: Semirings, Automata, Languages. EATCS Monographs on Theoretical Computer Science, Vol. 5. Springer, 1986.
12. Park, D.: Concurrency and automata on infinite sequences. LNCS 104(1981) 167–183.
13. Perrin, D., Pin, J.-E.: Infinite Words. Elsevier, 2004.

A Protocol for Serializing Unique Strategies

Marcel Crasmaru[1], Christian Glaßer[2]*, Kenneth W. Regan[3]**, and Samik Sengupta[3]

[1] Tokyo Inst. of Technology,
marcelis.titech.ac.jp
[2] Universität Würzburg,
glasserinformatik.uni-wuerzburg.de
[3] University at Buffalo,
{regan,samik}cse.buffalo.edu

Abstract. We devise an efficient protocol by which a series of two-person games G_i with unique winning strategies can be combined into a single game G with unique winning strategy, even when the result of G is a non-monotone function of the results of the G_i that is unknown to the players. In computational complexity terms, we show that the class UAP of Niedermeier and Rossmanith [10] of languages accepted by unambiguous polynomial-time alternating TMs is self-low, i.e., $\mathrm{UAP}^{\mathrm{UAP}} = \mathrm{UAP}$. It follows that UAP contains the Graph Isomorphism problem, nominally improving the problem's classification into SPP by Arvind and Kurur [2] since UAP is a subclass of SPP [10]. We give some other applications, oracle separations, and results on problems related to unique-alternation formulas.

1 Introduction

Consider the simple form of Nim where players alternate removing 1 or 2 stones from an initial pile of N stones, and the player who takes the last stone wins. A position N of this game is a win for the player to move iff N is not a multiple of 3, and the unique winning strategy for this player is to take the number that leaves a multiple of 3. Moreover, if the first player errs, the second player has a unique winning rejoinder. This extends to say that in the entire game tree of *all* possible plays from any initial position N, every non-terminal position reached either loses for the player to move or has a unique winning move. We term a game with this property *globally unique,* insofar as winning strategies wherever they exist are unique.

A two-person game is definable formally via an *alternating Turing machine* (ATM) M, which is coded like a nondeterministic TM but has separate *existential* and *universal* states. By convention the $\exists$-player moves first—i.e., the start state is existential—and the $\exists$-player wins a play from a starting game position

* Supported by a postdoctoral grant from the German Academic Exchange Service (Deutscher Akademischer Austauschdienst—DAAD).
** Supported in part by NSF grants INT 9726724 and CCR-9821040.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 660–672, 2004.
© Springer-Verlag Berlin Heidelberg 2004

x iff the computation path given by the players' moves ends in an accepting final configuration of M. Nim is coded by an ATM M that on input 0^N uses $\exists$-states and $\forall$-states that give options to mark off 1 or 2 cells, halts on reading the blank at the end of the input, and accepts iff it just came from an $\exists$-state. The definition of x being a first-player win, i.e. being in the language $L(M)$, recursively defines an existential configuration to be positive iff it has at least one move to a positive configuration, and a universal configuration to be positive iff all moves lead to a positive configuration. Then $x \in L(M)$ iff the start configuration on input x is positive. Niedermeier and Rossmanith [10] defined:

Definition 1. *An ATM is* unambiguous *if every positive non-final existential configuration has exactly one move to a positive configuration, and every negative non-final universal configuration has exactly one move to a negative configuration.*

They denoted by UAP the class of languages accepted by unambiguous polynomial-time bounded ATMs, since it is the natural alternating analogue of the class UP of languages accepted by unambiguous NTMs. They showed that $\mathrm{FewP} \subseteq \mathrm{UAP} \subseteq \mathrm{SPP}$.[1] The polynomial time bound on the ATMs enforces that each play of the corresponding game finishes within a reasonable amount of time. Thus languages in UAP comprise all denotations of reasonable globally-unique games.

The main intellectual problem solved by this paper is whether highly general combinations of globally-unique games are globally unique. Suppose Player 1 and Player 2 play a tournament G^* of Nim-style games $G_1, \ldots, G_m$, under the strange condition that the winner of the tournament is determined by an unknown Boolean function $f(x_1, \ldots, x_m)$ of the results of the individual games. The function f from Player 1's perspective may even be non-monotone, meaning that it could benefit Player 1 to play to *lose* the first game in order to win the tournament. Note that in Definition 1 there is no constraint on existential configurations (Player 1 to move) that are negative, nor on universal configurations (Player 2 to move) that are positive. What general rule and incentive can one give to induce uniquely optimal play in a position of G_1 where the player to move is losing?

Difficulties can be seen in the case of just two games where f is OR, so that Player 1 wins G^* if he wins either G_1 or G_2. The initial position of G^* may be a pair of initial positions σ_1, σ_2 of G_1 and G_2 respectively, except that σ_2 may not be known to the players at the start of G_1, and may even depend on the *outcome* of the actual play of G_1. If σ_1 and σ_2 are (both known and are) first-player wins, Player 1 seems to have two winning strategies: win G_1, or not care about G_1 and win G_2. We can attempt to force unique play in G by requiring

[1] FewP is the class of languages accepted by polynomial-time NTMs that have only polynomially-many accepting computations on any input. SPP [5] comprises those L defined by polynomial-time NTMs whose number of accepting computations either equals or surpasses by 1 its number of rejecting computations, with the latter case putting an input x into L.

Player 1 to win the winnable game with lowest index. But now what happens if Player 1 makes a mistaken move in some position π_1 of G_1 (possibly $\pi_1 = \sigma_1$ itself), and the given σ_2 turns out to be a second-player win? Now Player 2 seems to have two winning strategies: point out Player 1's mistake in G_1 or wait for G_2. Because G_1 is a globally-unique game, Player 2 *can* prove the mistake by switching roles with Player 1 and showing that some other move wins from π_1. We can stipulate that Player 2 must take the "prove a mistake" option when it works. But what if Player 2 errs by not doing so? Since any error must be punished uniquely, we must allow Player 1 to be able to win by pointing out his own mistake in G_1 at π_1. In case σ_2 turns out to be a first-player win after all, we must similarly stipulate that Player 1 must take the point-out-mistake option. But what if Player 1 now errs by not doing so? Can we avoid an infinite regress of artificial new rules?

When the number m of constituent games varies with the complexity parameter "n," there is also the danger that permitting players to review previous play will lead to exponential backtracking. The protocol for converting general alternating-P (i.e., PSPACE) computations into plays of globally-unique games in [1] involves exponential backtracking—it cannot be improved to polynomial unless PSPACE collapses into UAP, hence into SPP, thence into $\oplus$P and PP, which is generally disbelieved. The only advantage in our setting is that the individual games G_i already have global uniqueness of winning strategies.

Main Theorem 1. Given unambiguous polynomial-time ATMs defining games $G_1, \ldots, G_m$, and a polynomial-time referee function f, we can construct an unambiguous polynomial-time ATM M^* such that for all starting positions $x_1, \ldots, x_m$ of $G_1, \ldots, G_m$ respectively, $(x_1, \ldots, x_m) \in L(M^*) \iff (x_1, \ldots, x_m)$ wins for the $\exists$-player in G^*. Moreover, only a final deterministic part of M^* depends on f.

We prove this in greater generality by allowing $x_1, \ldots, x_m$ to arise dynamically from an overall starting position x given to M^*. Technically M^* becomes an *oracle ATM* and $x_1, \ldots, x_m$ become the *oracle queries* made by M^*—and the difference is that the oracle computation can be *adaptive.* "Adaptive" means that the starting position x_i for G_i can depend on the play of lower-indexed games, and so not be a fixed function of the input x. We lose no generality by supposing that "i" is encoded into x_i and all subsequent positions of each game G_i, so that we can picture a single globally-unique process controlling the constituent games.

Doing so lends our main theorem the denotation that the class UAP is *self-low,* and we state and prove it this way in the next section. Our proof still relies on the game formulation embodied in the statement of Theorem 1. A standard strategy for self-lowness in complexity theory is to show the class to be closed under (complements and) polynomial-size conjunctions and under "'one query," but in the adaptive case this strategy makes a step of existentially guessing queries that we do not know UAP a-priori to be closed under (note that NP $\subseteq$ SPP is disbelieved in [5]). Moreover we have not found a great simplification of

our proof even just for conjunctions as posed above—and closure of UAP under intersection was not noted in [10].

2 Main Theorem

For any oracle set A, UAP^A is the class of languages accepted by polynomial-time oracle ATMs that are unambiguous with oracle A. We re-state:

Theorem 1. $\mathrm{UAP}^{\mathrm{UAP}} = \mathrm{UAP}$.

Proof. Let M be a polynomial-time bounded oracle ATM that has the global uniqueness property with oracle $A \in \mathrm{UAP}$, and let M_A be a polynomial-time bounded non-oracle ATM with global uniqueness that accepts A. The following protocol implicitly defines a polynomial-time bounded non-oracle ATM M^* with global uniqueness that accepts $L(M^A)$. The $\exists$-player and $\forall$-player initially and always make their corresponding moves in M until M makes a query. The subgame G_i arises when M makes the i-th query y_i in its computation path, and commences from the position σ_i corresponding to the initial configuration of M_A on input y_i. Although the games G_i may appear uniform and dependent since M_A is the same machine in all instances, the query strings y_i may contain the actual rules of the game G_i with M_A used intuitively only as a universal reader. Thus no generality seems to be lost by our technical setting, compared to the motivational description above.

Before describing the protocol, we need to define its ingredients. First, a global total ordering of legal moves at every possible position of every game G_i is imposed. We picture the lowest legal move at a position as "leftmost." The $\exists$-player and $\forall$-player play each G_i in sequence, except that either has the option at any time to "trump previous play." At every move of a game G_i, the player to move *declares* "Win" or "Lose"—and in the latter case, *must* play the leftmost legal move on pain of otherwise being liable for "trumping." If the plays of each G_i and the intervening simulation of M all finish without either player using the trump option, then the overall winner is determined by the final configuration of M that is reached. When M is an oracle P-machine, it helps to picture M as a neutral referee who determines the overall winner based only on the outcomes of the games (i.e., queries).

Trumping means to backtrack to a previous position π in some game G_j—with either the same or the opposite player to move at π—and change the declaration and/or move that was made there. The essence is that the following three rules avoid the infinite-regress problems described in Section 1.

(a) If the original player to move at π declared "Lose," then either player can trump by declaring "Win" and playing a move.
(b) If the original player to move at π declared "Win," then either player can trump by declaring "Win" and playing a different move.
(c) If the original player to move at π declared "Win," then either player can trump by playing that move, declaring "Win" at the succeeding position π', and making a move from π'.

Once "trump" is declared at π, future "trump" options may return only to games G_i with $i < j$, to positions earlier than π in the original play of G_j, or:

> In any ensuing play after a case-(c) trump move, it is legal to make a "counter-trump" move of type (b) by returning to π, declaring "Win," and making a move—which again must be different from the move that was originally made at π.

In all cases, if the trumping player wins the ensuing play of game G_j, then he wins the entire game G^*. Note, however, that terminal positions of G_j still allow the above (counter-)trumping options. The key idea of the proof is that *whenever a victorious trumping move is legal, it is the unique optimal play.*

These provisos ensure that plays of the expanded game G^* have length polynomial in both m and the maximum length of any play of any G_j. Every position of the game G^* encodes not only the corresponding position of the current game G_j, but also the entire game history of G^* to that point—this in particular determines all possible trumping moves. Moreover, the number of legal (trumping) moves available in any position expands by only a polynomial factor. This completes the description of the game G^*.

To prove that G^* has the global uniqueness property—regardless of the referee's verdict in plays where all m sub-games are completed without trumping—we make the following observations:

1. There is a unique play α_0 of the sequence of games that allows no successful trumping opportunities. In this play, each player makes the correct declaration at every step, and a player declaring "Win" makes the unique winning move. The play α_0 has no trumping moves itself.
2. If a play α deviates from α_0 at a position π by an incorrect move or declaration, then in any play forward from π, either player can win by rewinding to position π and trumping. Here are the possible deviations and trump responses:
   (a) Declaring "Lose" in a winning position. Trump by declaring "Win" and playing a winning move. Trump option (a) is the only one that applies, and by global uniqueness of G_j itself, the winning move is unique. (It may be the leftmost move itself—i.e., the one that was played.)
   (b) Declaring "Win" in a winning position, but not playing the winning move. Trump by declaring "Win" and playing the correct move—which again is unique by global uniqueness of G_j. This option (b) is the unique correct reply.
   (c) Declaring "Win" in a losing position. Trump by playing the move that was played (which need not be the leftmost move), then declaring "Win," and making a winning move. Trump option (a) doesn't apply and (b) is false, so (c) is unique—as is the winning move in the second position.

The other kind of deviation from α_0 that can occur is a trumping move. We distinguish between a "virgin" deviation and one in reply to an earlier deviation at the same position π. We first argue inductively that a virgin deviation (by

trumping) always loses for the trumping player. If it is a trumping move of type (a) or (b), then the resulting position is a win for the responding player. Since (by induction) trumping at a position earlier than π along α_0 would be virgin and thus lose, and since no "counter-trump" at π is available, winning the ensuing play of the game G_j is the only winning option for the responder—and it is unique by global uniqueness of G_j. If the virgin deviation is of type (c), then the position π is winning in G_j for the player P to move there, and P originally made the right move μ—because it was not a deviation. The trumper makes μ and then declares that the resulting position π' is a win for him, making a move μ' there. Here the trumper is wrong about π', so the responder can win the ensuing play of G_j. The responder does have the "counter-trump" option available, but since μ *was* the right move, it would lose. Thus winning the ensuing play of G_j is the unique way to punish a virgin deviation of type (c).

To finish the argument, consider any position ρ in which the player P to move can win. If ρ is along the unique optimal play α_0, then all alternatives to furthering α_0 lose, so *a fortiori* the unique winning move must be the one that furthers α_0. If not, then there is a first position π at which the play α deviates from α_0. If the deviation at π was by a bad move, or by a move accompanying the wrong declaration, and there has been no trumping at π already, then trumping at π is legal and winning for either player. Since all other moves by P either leave trumping at π open to the other player, or are virgin trumps at positions earlier than π along α_0, trumping at π is the unique winning move for P.

Otherwise, a trumping move has occurred at π, and position ρ is immediately afterward or in ensuing play from that move. If the trump move was of type (a) or (b), virgin or not, then since there is no counter-trump at π, and since trumping below π is virgin and hence losing, there is no option other than the ensuing play in game G_j, whose global uniqueness carries through here. If it was a virgin trump of type (c), then it was incorrect, and ensuing play is globally unique as argued above. It remains to cover cases where ρ ensues from a trump of type (c) at π that comes after an incorrect "Win" declaration and/or move at π that represented the original deviation from α_0.

(i) The "Win" declaration was incorrect—i.e., position π is losing in G_j for the player to move. In this case the type-(b) counter-trump option loses. As in the cases above of a type (a) or (b) trump at π, the only way ρ can be winning for the player to move is for it to be winning in G_j, and the unique winning play in G_j is the unique way to win G^*.

(ii) The "Win" declaration was correct, but the original player made the wrong move. Then the counter-trump option at π is winning. Since it is available to both players at all times in the ensuing play of G_j, it is always the unique optimal reply.

This finishes the argument that G^* represents a general protocol for playing a series of globally-unique games in a globally-unique way. Note the symmetry between the two players in all aspects of the argument, and the use of "leftward-ness" *only* to constrain options when a player declares "Lose." This already amounts to a proof that $\mathrm{P}^{\mathrm{UAP}} = \mathrm{UAP}$.

For $\mathrm{UAP}^{\mathrm{UAP}} = \mathrm{UAP}$ we need only describe the machine M^* a little further. If a trumping move occurs during the play, then M^* follows the resolution of this trump—and all possible counter-trumps and trumps at earlier positions—and gives the result as its own final answer (accept iff the $\exists$-player wins). If not, then the play β_i from σ_i ends with a winner of that game, and M^* takes the result as the answer to the oracle query y_i. If the sub-games for all queries reach their conclusion and no trumps are made, then eventually a terminal position of the game G_M represented by M is reached. If it is winning for the player who just moved, then the opposing player still has the option of trumping; if not, then the game ends and the opposing player wins the game G_{M^*} right there. (This last proviso preserves global uniqueness even at the terminal position of G_{M^*}.)

For each oracle query y_i, there is a unique "correct" play α_i from σ_i, being a segment of "α_0" as established above. If β_i deviates from α_i, then in any ensuing configuration of M^*, either side has the winning option of trumping at the (first) point of deviation, and this winning option is unique. If not, then M^* gets the correct answer to the oracle query $y_i \in ?A$. Putting this all together, the global uniqueness that M has with oracle A carries through to global uniqueness of M^*, and $L(M^*) = L(M^A)$. □

We remark that coming into or exiting a sub-game G_i may give two consecutive moves to one of the players, but (i) this does not matter to the above analysis, (ii) it is easy to avoid beforehand by modifying M_A and M to avoid this, and (iii) it is fixable afterwards by padding methods used here and in results of [1] quoted below anyway.

Corollary 1. UAP *is closed under all Boolean operations.*

One consequence of self-lowness is that several hierarchies based on Turing reductions with uniqueness are contained in UAP. Niedermeier and Rossmanith [10] define "$AU\Sigma_k^p$" to be the subclass of UAP of languages defined by games that start with the $\exists$-player and have at most $k-1$ alternations in any play between the $\exists$-player and the $\forall$-player. They credit to "Hemaspaandra [unpublished]" the observation that these coincide with the levels UP,UP[co-UP],UP[co-UP[UP]],... defined by quantifiers on UP and co-UP predicates. Lange and Rossmanith [9] define *"AUPH"* to be the union of these classes, *"UPH"* to be the union of the levels UP, $\mathrm{UP}^{\mathrm{UP}}, \mathrm{UP}^{\mathrm{UP}^{\mathrm{UP}}}, \ldots$, and "$\mathcal{UPH}$" to be the union of the "smart-reduction" levels $\mathrm{UP}, \mathrm{UP}_s^{\mathrm{USAT}}, \mathrm{UP}_s^{\mathrm{USAT}_2}$ (where "USAT_2" characterizing $\mathrm{UP}_s^{\mathrm{USAT}}$ was described above following Theorem 3). They give SPP as an upper bound for these hierarchies, and Theorem 1 allows us immediately to improve this to:

Corollary 2. $\mathcal{UPH} \subseteq \mathrm{UAP}$, *and also* $\mathrm{UAP}_s^{\mathrm{USAT}} = \mathrm{UAP}$.

Our main corollary, with application to the complexity of the Graph Isomorphism problem, needs its own section.

3 Logic Games, Smart Reductions, and Graph Isomorphism

In [1], UAP was characterized as the class of languages that polynomial-time many-one reduce to the following promise problem *GUQBF*:

INSTANCE: A Boolean formula F in variables $x_1, \ldots, x_d$ for some $d \geq 0$ that induces the quantified Boolean formula $\psi = (\exists x_d)(\forall x_{d-1}) \cdots (Qx_1)\, F$.
PROMISE: The logic game on ψ has the global uniqueness property.
QUESTION: Is ψ true?

Here "Q" is $\exists$ if d is odd, else it is $\forall$. In the usual logic game, the $\exists$-player goes first and assigns 0 or 1 to x_d, then the $\forall$-player assigns to x_{d-1}, and the players alternate moves until F is either made true, a win for $\exists$, or false, a win for $\forall$. A family of small formulas (wins for $\exists$) in which the promise holds is typified by

$$F = x_1 \vee x_2(x_3 \vee x_4(x_5 \vee x_6(x_7 \vee x_8))),$$

where we have written AND as multiplication binding tighter than OR to improve visual intuition. The $\exists$-player must assign 1 to even-numbered variables, whereupon the $\forall$-player's next choice is immaterial; but if the $\exists$-player wrongly assigns 0 to (say) x_8, then the $\forall$-player succeeds uniquely by assigning 0 to x_7 and so on, whereupon the $\exists$-player's moves become immaterial and losing.

The promise problem USAT can be regarded as a sub-promise of *GUQBF*:

INSTANCE: A Boolean formula F in variables $x_1, \ldots, x_d$, here viewed as inducing the quantified Boolean formula $\psi = (\exists x_d)(\exists x_{d-1}) \cdots (\exists x_1) F$.
PROMISE: F has zero or one satisfying assignment (which implies that the logic game on ψ has the global uniqueness property).
QUESTION: Is F satisfiable—i.e., is ψ true?

Abstractly, a promise problem (Q, R) has *promise set* Q and *property set* R, and a language S is a *solution* if $S \cap Q \subseteq R$ and $Q \setminus S \subseteq \bar{R}$. A reduction to a promise problem is required to be simultaneously a reduction to every solution. For a many-one reduction f this entails $Ran(f) \subseteq Q$ [11], but for reductions of Turing type it is possible that allowing queries to strings outside of Q makes a difference, as discussed by Grollman and Selman [7]. They called a Turing reduction to a promise problem *smart* if it never queries strings outside the promise set. A many-one reduction is a special case of a *smart* Turing reduction.

Note that a language A belongs to UP if and only if $A \leq_m^p$ USAT, so that USAT characterizes UP by many-one reductions the same way *GUQBF* characterizes UAP. However, the class $\mathrm{P}_s^{\mathrm{USAT}}$ of languages with smart polynomial-time Turing reductions to USAT is apparently larger: it is closed under complements, contains FewP [4],[2] and contains the graph-isomorphism problem:

[2] Here and in [10] the class is written "$\mathrm{P}^{\mathcal{UP}}$," but we write "$\mathrm{P}_s^{\mathrm{USAT}}$" to avoid possible font confusion with P^{UP} and to emphasize the role of USAT.

Theorem 2 (after [2]). $\mathrm{GI} \in \mathrm{P}_s^{\mathrm{USAT}}$.

Proof. The main lemma of [2] creates an oracle reduction to a group-theoretically defined language L such that every query w to L has a unique witnessing answer. Arvind and Kurur remark that the queries are "UP-like." Because their language L belongs to NP, the w can be transformed to queries w' to USAT, and the "UP-like" property is preserved and makes this a "smart" polynomial-time Turing reduction to USAT. Their algorithm computes what they call a "UP-single-valued function with SPP oracle," but the use of this function to decide Graph Isomorphism stays within the bounds of $\mathrm{P}_s^{\mathrm{USAT}}$. □

Corollary 3 (to Theorem 1). $\mathrm{P}_s^{\mathrm{USAT}} \subseteq \mathrm{UAP}$ *(hence* $\mathrm{GI} \in \mathrm{UAP}$*).*

Proof. The oracle P-machine accomplishing a smart reduction to USAT initiates only those plays of the logic game on existential formulas that have (global) uniqueness, so it becomes a non-oracle UAP-machine via Theorem 1. □

4 Globally Unique Formulas

We note first that the promise problem *GUQBF* is invariant under equivalence of Boolean formulas F and F', since the PQBFs induced from F and F' define the same logic game. Thus we can characterize game positions with global uniqueness by selecting representatives from each equivalence class. Let TRUE (resp., FALSE) denote a constant Boolean formula whose value is 1 (resp., 0). We define inductively $A_0 = \{\,\textsc{False}\,\}$, $B_0 = \{\,\textsc{True}\,\}$, and for $d \geq 1$,

$$A_d = \{\,(x_d \wedge \neg F_1) \vee (\bar{x}_d \wedge \neg F_0) : F_0, F_1 \in B_{d-1}\,\},$$
$$B_d = \{\,(x_d \wedge \neg F_1) \vee (\bar{x}_d \wedge \neg F_0) : (F_0 \in A_{d-1} \text{ and } F_1 \in B_{d-1}) \text{ or } (F_0 \in B_{d-1} \text{ and } F_1 \in A_{d-1})\,\}.$$

Here B_d comprises those d-variable Boolean formulas that induce globally-unique logic games with the ∃-player to move that are wins for the ∃-player, while A_d comprises those in which the ∃-player is to move but loses. Note that A_d says that both substitutions for x_d leave Boolean formulas whose *negations* are in B_{d-1}, meaning the negations are unique wins with the ∃-player to move, which implies that the resulting formulas themselves are unique ∀-player wins with the ∀-player to move. The recursion for B_d is interpreted similarly.

For example, $A_1 = \{\,(x_1 \wedge \textsc{False}) \vee (\bar{x}_1 \wedge \textsc{False})\,\}$, which is equivalent to $A_1 = \{\,\textsc{False}\,\}$ and (mentioning x_1) to $A_1 = \{\,x_1 \wedge \bar{x}_1\,\}$. Also

$$B_1 = \{\,(x_1 \wedge \textsc{True}) \vee (\bar{x}_1 \wedge \textsc{False}),\quad (x_1 \wedge \textsc{False}) \vee (\bar{x}_1 \wedge \textsc{True})\,\},$$

which reduces to $B_1 = \{\,x_1, \bar{x}_1\,\}$. The following is shown in [1] by straightforward induction.

Lemma 1 ([1]). *The formulas in $A_d \cup B_d$ are pairwise inequivalent, and when interpreted as logic games with the $\exists$-player to move, those in A_d are $\exists$-player losses with global uniqueness, while those in B_d are $\exists$-player wins with global uniqueness.*

Every formula in A_d has exactly N_d satisfying assignments, where $N_d = (2/3)(2^d - 1)$ if d is even, while $N_d = N_{d-1}$ if d is odd. Moreover, every formula in B_d has exactly $N_d + 1$ satisfying assignments. The journal version [1] shows that UAP $\subseteq$ SPP also follows from this. Counting A_d and B_d shows they contain Boolean functions all of whose formulas have bit-size $\Omega(2^d)$.

Here we study the analogues of A_d and B_d for polynomials in arithmetic mod 2, under the standard correspondence $H(\text{TRUE}) = 1$, $H(\text{FALSE}) = 0$, $H(x_i) = x_i$, $H(\neg f) = 1 - H(f)$, $H(f \wedge g) = H(f)H(g)$, and $H(f \vee g) = H(f) + H(g) - H(f)H(g)$. Define I_d to be the ideal generated by $\{x_1^2 - x_1, \ldots, x_d^2 - x_d\}$ in $\mathcal{F}[x_1, \ldots, x_d]$, where we take the field $\mathcal{F}$ to be the integers mod 2 or any field of characteristic 2. We observe (proofs from here on are in the full paper):

Lemma 2. *Over $\mathcal{F}$, for $d \geq 1$, every polynomial in $H(A_d)$ has degree $d - 1$ modulo $\mathcal{I}_d$, while every polynomial in $H(B_d)$ has degree d modulo $\mathcal{I}_d$.*

It is curious that the cancellation is guaranteed only in characteristic 2. This leads to the following decision problem "DEGREE MOD $\mathcal{I}_n$" about representations of Boolean functions by polynomials, which we have not seen studied in the literature on the "polynomial method" (see [3] for stem references).

INSTANCE: A formula for a polynomial $f \in \mathcal{F}[x_1, \ldots, x_n]$.
QUESTION: Does f have degree n when reduced modulo the ideal $\mathcal{I}_n$?

Theorem 3. DEGREE MOD $\mathcal{I}_n$ *is polynomial-time many-one hard for* UAP.

What is the exact complexity of DEGREE MOD $\mathcal{I}_n$? Note that f is a tautology or unsatisfiable iff $H(f)$ reduces to 1 or 0, so that the related question of whether the degree mod $\mathcal{I}_n$ is *positive* is NP-hard. It seems not to follow simply, however, that our problem of whether the degree is n is NP-hard. The problem of whether a given set of polynomials reduces to 1 under the Gröbner basis algorithm belongs to the second level of the polynomial hierarchy [8], but the same for DEGREE MOD $\mathcal{I}_n$ would put UAP inside PH, which also seems questionable. Nor do we even know whether DEGREE MOD $\mathcal{I}_n$ belongs to polynomial space! This problem deserves further study, and its hardness for UAP provides a context.

Now define $\mathcal{A}_d$ to be the set of d-variable Boolean formulas that are equivalent to a formula in A_d, and $\mathcal{B}_d$ similarly with regard to B_d. Then set $\mathcal{A} = \cup_d \mathcal{A}_d$, $\mathcal{B} = \cup_d \mathcal{B}_d$. Clearly both languages are UAP-hard, and by the recursive definitions, $\mathcal{A} \cup \mathcal{B}$ is polynomial-time self-reducible. We prove something stronger:

Theorem 4. *$\mathcal{A}$ and $\mathcal{B}$ are polynomial-time isomorphic and self-reducible.*

It is not clear whether these equations yield conjunctive or disjunctive self-reductions that are polynomially well-founded. Many other questions pop to mind about the sets $\mathcal{A}$ and $\mathcal{B}$. Are they NP-hard? Is their complexity tied to that of the promise problem *GUQBF*? Are they learnable? We note:

Proposition 1. *For every formula $F \in \mathcal{B}_d$ with d even, there are $2^{d/2}$ satisfying truth assignments α such that flipping the value on that assignment to* FALSE *leaves a formula in $\mathcal{A}_d$.*

The $2^{d/2}$ assignments are those reached by unique winning play by $\exists$ against the $2^{d/2}$ possible different plays by the $\forall$-player. Being able to compute any one of them deterministically is equivalent to telling whether the QBF induced from F is true, i.e. to solving *GUQBF* on that instance. Thus unless UAP = P these ranges must avoid polynomial-time computable functions in some sense. Does that give them any pseudo-random properties?

5 Oracles and the UAP Versus SPP Problem

Since SPP is intuitively just above UAP and is also self-low, the natural next question to attack is whether UAP = SPP. Niedermeier and Rossmanith [10] noted that UAP meets their definition of *locally definable* classes, while SPP is the smallest class thrust out by a notion of *gap-definability* [5] that seemed (to them) inherently non-local. Put simply, the difference is that the gap-counting condition that defines SPP need only count the results of computation paths, while that defining UAP depends on all internal configurations of the computation tree. Thus equality may seem surprising. Inequality, however, would mean that we have found another self-low counting class, one below the level of "gap-definability" (see discussion in [5, 6]).

We have separated UAP by oracle from its lower neighbor $\mathrm{P}_s^{\mathrm{USAT}}$. This follows on constructing an oracle A such that $\mathrm{UP}^{\mathrm{UP}^A} \not\subseteq \mathrm{P}^{\mathrm{NP}^A}$, which is not subsumed by oracle results for UP-based hierarchies in [4] and [10]:

Theorem 5. *There exists an oracle relative to which* $\mathrm{UAP} \neq \mathrm{UP}_s^{\mathrm{USAT}}$.

This extension makes plausible that further levels of the smart-reductions hierarchy "$\mathcal{UPH}$" are also different from UAP. The proof is in the full paper.

6 Conclusions and Open Problems

Our results enhance the value of UAP as a natural complexity class. It resides just below SPP and is likewise self-low. The jump from UAP to SPP, however, may go from a large "locally-definable" class [10] to the smallest gap-definable class [5]. If they are equal, this contrast makes the equality interesting; if they are different, why are they so structurally alike? We have tried to separate them by an oracle, and not found it easy. Even if UAP turns out to equal SPP, our

protocol itself—which appears markedly different from the proof of self-lowness of SPP in [5]—would likely retain independent interest.

The property of global uniqueness itself—of an ATM or of (the QBF induced by) a Boolean formula—also deserves study. Is *GUQBF* a case of a promise problem with a promise more difficult, no more difficult, or incomparable with the problem being solved? Does the promise problem *GUQBF* even have solutions in UAP? The isomorphic languages $\mathcal{A}$ and $\mathcal{B}$ of Boolean formulas seem to have many interesting properties for further exploration.

Last, we look to further applications of the game-playing protocol behind Theorem 1. It remains serial even in the non-adaptive case where the initial game positions $x_1, \ldots, x_m$ are known in advance, since "trumping" relies on a fixed total ordering of the games G_i and temporal order of their plays. Extending it for parallel or "asynchronous" plays of the G_i might impact (unambiguous) computation in classes below P.

Acknowledgments. We thank Steve Fenner for helpful conversations at Complexity'02 and afterward, and several authors of [4,9,10] for answering our precedent-search queries. We thank all referees for helpful comments.

References

1. S. Aida, M. Crasmaru, K. Regan, and O. Watanabe. Games with a uniqueness property. In *Proc. 19th Annual Symposium on Theoretical Aspects of Computer Science,* volume 2285 of *Lect. Notes in Comp. Sci.,* pages 396–407. Springer Verlag, 2002.
2. V. Arvind and P. Kurur. Graph isomorphism is in SPP. In *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science,* pages 743–750, 2002. Also ECCC TR02-037.
3. R. Beigel. The polynomial method in circuit complexity. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory,* pages 82–95, 1993. Revised version, 1995.
4. J.-Y. Cai, L. Hemachandra, and J. Vyskoc. Promise problems and guarded access to unambiguous computation. In *Complexity Theory: Current Research, Edited by Klaus Ambos-Spies, Steven Homer, and Uwe Schöning, Cambridge University Press.* Springer Verlag, 1993.
5. S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *J. Comp. Sys. Sci.,* 48:116–148, 1994.
6. S. Fenner, L. Fortnow, and L. Li. Gap-definability as a closure property. *Inform. and Comp.,* 130:1–17, 1996.
7. J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput,* 17:309–335, 1988.
8. Pascal Koiran. Hilbert's Nullstellensatz is in the polynomial hierarchy. *Journal of Complexity,* 12(4):273–286, December 1996.
9. K.-J. Lange and P. Rossmanith. Unambiguous polynomial hierarchies and exponential size. In *Proc. 9th Annual IEEE Conference on Structure in Complexity Theory,* pages 106–117, 1994.

10. Rolf Niedermeier and Peter Rossmanith. Unambiguous computations and locally definable acceptance types. *Theoretical Computer Science,* 194(1–2):137–161, 1998.
11. A. Selman. Promise problems complete for complexity classes. *Inform. and Comp.,* 78:87–98, 1988.

A Combinatorial Strongly Subexponential Strategy Improvement Algorithm for Mean Payoff Games*

Henrik Björklund, Sven Sandberg, and Sergei Vorobyov

Uppsala University, Information Technology Department,
Box 337, 751 05 Uppsala, Sweden
vorobyov@csd.uu.se

Abstract. We suggest the first strongly subexponential and purely combinatorial algorithm for mean payoff games. It is based on solving a new "controlled" version of the shortest paths problem. By selecting exactly one outgoing edge in each of the controlled vertices we want to maximize the shortest distances to the unique sink. Mean payoff games easily reduce to this problem. To compute the longest shortest paths, player MAX selects a strategy (one edge in each controlled vertex) and player MIN responds by evaluating shortest paths to the sink in the remaining graph. Then MAX locally changes choices in controlled vertices, making attractive switches that seem to increase shortest paths (under the current evaluation). We show that this is a monotonic strategy improvement, and every locally optimal strategy is globally optimal. A careful choice of the next iterate results in a randomized algorithm of complexity $\min(poly(n) \cdot W,\ 2^{O(\sqrt{n \log n})})$, which is simultaneously pseudopolynomial (W is the maximal absolute edge weight) and subexponential in the number of vertices n. All previous algorithms for mean payoff games were either exponential or pseudopolynomial (which is purely exponential for exponentially large edge weights).

1 Introduction

Infinite games on finite graphs play a fundamental role in model checking, automata theory, logic, and complexity theory. We consider the problem of solving *mean payoff games* (MPGs) [18,8,22], also known as *cyclic games* [12,19]. In these games, two players take turns moving a pebble along edges of a directed edge-weighted graph. Player MAX wants to maximize and player MIN to minimize (in the limit) the average edge weight of the infinite path thus formed. Mean payoff games are determined, and every vertex has a *value,* which each player can secure by a uniform positional strategy. Determining whether the value is above (below) a certain threshold belongs to NP∩coNP. The well-known parity games, also in NP∩coNP, polynomial time equivalent to model-checking for μ-calculus [10,9], are polynomial time reducible to MPGs. Other well-known games

* Supported by grants from the Swedish Research Council (VR) and the Swedish Foundation for International Cooperation in Research and Higher Education (STINT).

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 673–685, 2004.
© Springer-Verlag Berlin Heidelberg 2004

with NP∩coNP decision problems, to which MPGs reduce, are *simple stochastic* [6] and *discounted payoff* [20,22] games. At present, despite substantial efforts, there are no known polynomial time algorithms for the games mentioned.

All previous algorithms for mean payoff games are either pseudopolynomial or exponential. These include a potential transformation method by Gurvich, Karzanov, and Khachiyan [12,19], and a dynamic programming algorithm solving k-step games for big enough k by Zwick and Paterson [22]. Both algorithms are *pseudopolynomial* of complexity $O(poly(n) \cdot W)$, where n is the number of vertices and W is the maximal absolute edge weight. For both algorithms there are known game instances on which they show a worst-case $\Omega(poly(n) \cdot W)$ behavior, where W may be exponential in n. Reduction to simple stochastic games [22] and application of the algorithm from [15] gives subexponential complexity only if the game graph has bounded outdegree. The subexponential algorithms we suggested for simple stochastic games of arbitrary outdegree in [3] make $2^{O(\sqrt{n \log n})}$ iterations, but when reducing from mean payoff games, the weights may not allow each iteration (requiring solving a linear program) to run in strongly polynomial time, independent of the weights. This drawback is overcome with the new techniques presented in this paper, which avoid the detour over simple stochastic games altogether.

We suggest a *strongly subexponential strategy improvement* algorithm, which starts with some strategy of the maximizing player and iteratively "improves" it with respect to a strategy evaluation function based on computing shortest paths. Iterative strategy improvement algorithms are known for the related simple stochastic [13,7], discounted payoff [20], and parity games [21,2]. Until the present paper, a direct combinatorial iterative strategy improvement for mean payoff games appeared to be elusive. Reductions to discounted payoff games and simple stochastic games (with known iterative strategy improvement) lead to numerically unstable computations with long rationals and solving linear programs. The algorithms suggested in this paper are free of these drawbacks. Our method is discrete and requires only addition and comparison of integers in the same order of magnitude as occurring in the input. There is also a simple reduction from parity games to MPGs, and thus our method can be used to solve parity games. Contrasted to the strategy improvement algorithms of [21,2], the new method is conceptually much simpler, more efficient, and easier to implement.

We present a simple and discrete randomized subexponential strategy improvement scheme for MPGs, and show that for any integer p, the set of vertices from which MAX can secure a value $> p$ can be found in time

$$\min\left(O(n^2 \cdot |E| \cdot W),\ 2^{O(\sqrt{n \log n})}\right).$$

The first bound matches those from [12,22,19], while the second part is an improvement when $n \log n = o(\log^2 W)$.

The new strategy evaluation for MPGs may be used in several other iterative improvement algorithms, which are also applicable to parity and simple stochastic games [21,13,7]. These include random single switch, all profitable switches, and random multiple switches [1]. They are simplex-type algorithms, very ef-

ficient in practice, but without currently known subexponential upper bounds, and no nontrivial lower bounds.

Outline. Section 2 defines mean payoff games and introduces the associated computational problems. Section 3 describes the longest-shortest paths problem and its relation to mean payoff games. In addition, it gives an intuitive explanation of our algorithm and the particular randomization scheme that achieves subexponential complexity. Section 4 describes the algorithm in detail and Section 5 states the two main theorems guaranteeing correctness. Finally, Section 6 explains how to improve the cost per iteration and provides complexity results, generalizations, and applications. All proofs and extensions may be found in [4].

2 Mean Payoff Games and Associated Problems

A *mean payoff game* (MPG) is played by two adversaries, MAX and MIN, on a finite, directed, edge-weighted, leafless graph $G = (V = V_{\text{MAX}} \uplus V_{\text{MIN}}, E, w)$, where $w : E \to \mathbb{Z}$ is the weight function. The players move a pebble along edges of the graph, starting in some designated initial vertex. If the current vertex belongs to V_{MAX}, MAX chooses the next move, otherwise MIN does. The duration of the game is infinite. The resulting infinite sequence of edges is called a *play*. The *value* of a play $e_1 e_2 e_3 \ldots$ is defined as $\liminf_{k\to\infty} 1/k \cdot \sum_{i=1}^{k} w(e_i)$. The goal of MAX is to maximize the value of the play, while MIN tries to minimize it. In the decision version, the game also has an integer threshold value p. We say that MAX *wins* a play if its value is $> p$, while MIN wins otherwise.

A *positional strategy* for MAX is a function $\sigma : V_{\text{MAX}} \to V$ such that $(v, \sigma(v)) \in E$ for all $v \in V_{\text{MAX}}$. Positional strategies for MIN are defined symmetrically. Every MPG is memoryless determined, which means that for every vertex v there is a value $\nu(v)$ and positional strategies of MAX and MIN that secure them payoffs $\geq \nu(v)$ and $\leq \nu(v)$, respectively, when a play starts in v, against any strategy of the adversary [18,8,12,19,5]. Moreover, both players have *uniform* positional strategies securing them optimal payoffs independently of the starting vertex. Accordingly, we consider positional strategies only. Given a positional strategy σ for MAX, define $G_\sigma = (V, E')$, where $E' = E \setminus \{(v,u) | v \in V_{\text{MAX}} \text{ and } \sigma(v) \neq u\}$, i.e., G_σ results from G by deleting all edges leaving vertices in V_{MAX}, except those selected by σ. Note that if both players use positional strategies, the play will follow a (possibly empty) path to a simple loop, where it will stay forever. The value of the play is the average edge weight on this loop [8,12].

We address the following algorithmic problems for MPGs.

The Decision Problem. Given a distinguished start vertex and a threshold value p, can MAX guarantee a value $> p$?

p-Mean Partition. Given p, partition the vertices of an MPG G into subsets $G_{\leq p}$ and $G_{>p}$ such that MAX (MIN, resp.) can secure a value $> p$ ($\leq p$, resp.) starting from every vertex in $G_{>p}$ (in $G_{\leq p}$, resp.)

Ergodic Partition. Compute the value of each vertex of the game. This gives an *ergodic* [12] partition of the vertices into subsets with the same value.

Our basic algorithm solves the *0-mean partition* problem, which subsumes the p-mean partition. Indeed, subtracting p from the weight of every edge makes the mean value of all loops (in particular, of optimal ones) smaller by p, and the problem reduces to 0-mean partition. The complexity remains the same for integer thresholds p, and changes slightly for rational ones; see Section 6. Clearly, the p-mean partition subsumes the decision problem. Other problems to which our algorithm extends are ergodic partition and finding optimal strategies [4].

3 A High-Level Description of the Algorithm

We start by informally describing the key ingredients of our algorithm.

3.1 The Longest-Shortest Paths Problem (LSP)

The essential step in computing 0-mean partitions can be explained by using a "controlled" version of the well-known *single sink (target) shortest paths problem* on directed graphs. Suppose in a given digraph some set of *controlled* vertices is distinguished, and we can select *exactly one edge* leaving each controlled vertex, deleting all other edges from these vertices. Such a selection is called a *positional strategy*. We want to find a positional strategy that maximizes the shortest paths from all vertices to the distinguished sink (also avoiding negative cycles that make the sink unreachable and the distances equal $-\infty$). For a strategy σ denote by G_σ the graph obtained from G by deleting all edges from controlled vertices except those in σ. Formally, the problem is specified as follows.

THE LONGEST-SHORTEST PATHS PROBLEM (LSP).
Given: (1) a directed weighted graph G with unique sink t,
(2) a set of *controlled* vertices U of G, with $t \notin U$.
Find: a positional strategy σ such that in the graph G_σ the shortest simple path from every vertex to t is as long as possible (over all positional strategies).

If a negative weight loop is reachable in G_σ, the length of the shortest path is $-\infty$, which MAX does not want. If only positive loops are reachable, and t is not, then the shortest path distance is $+\infty$. For our purposes it suffices to consider a version of the problem above with the following additional input data.

Additionally Given: some strategy τ, which guarantees that in the graph G_τ there are no negative weight cycles.

This strategy τ ensures that the longest shortest distance from every vertex to the sink t is not $-\infty$; it is not excluded that τ or the optimal strategy will make some distances equal $+\infty$. We make sure that our algorithm never comes to a strategy that allows for negative cycles. The simplifying additional input strategy τ is easy to provide in the reduction from MPGs, as we show below.

For DAGs, the LSP problem can be solved in polynomial time using dynamic programming. Start by topologically sorting the vertices and proceed backwards from the sink (distance 0), using the known longest-shortest distances for the preceding vertices.

3.2 Relating the 0-Mean Partition and the LSP Problems

To find a 0-mean partition in an MPG G, add a *retreat vertex* t to the game graph with a self-loop edge of weight 0, plus a 0-weight *retreat edge* from every vertex in V_{MAX} to t. From now on, we assume G has undergone this transformation. Clearly, we have the following property.

Proposition 1. *Adding a retreat vertex t does not change the 0-mean partition of the game, except that t is added to the $G_{\leq 0}$ part.* □

This is because we do not introduce any new loops allowing MAX to create positive cycles, or MIN to create new nonpositive cycles. MAX will prefer playing to t only if all other positional strategies lead to negative loops.

The key point is now as follows. Break the self-loop in the retreat t and consider the LSP problem for the resulting graph, with t being the unique sink. Vertices V_{MAX} become controlled, and the initial strategy (the "additionally given" clause on the previous page) selects t in every controlled vertex, guaranteeing that no vertex has distance $-\infty$.[1] We have the following equivalence.

Theorem 1. *The partition $G_{>0}$ in the MPG consists exactly of those vertices for which the longest-shortest path distance to t is $+\infty$.* □

Our algorithm computes the longest shortest paths by iterative improvement to be explained below. Surprisingly (to our knowledge), the longest-shortest path problem was not previously addressed in the literature and its relation to strategy improvement for the MPG problem was not exploited before.[2]

The evaluation of the shortest paths for a fixed positional strategy gives a useful quality measure on strategies that can be used in different iterative improvement schemes, discussed later.

3.3 The Algorithm

Our algorithm computes longest-shortest paths in the graph resulting from a mean payoff game (after adding the retreat vertex and edges, as explained

[1] Actually, there may exist negative loops consisting only of vertices from V_{MIN}. Such loops are easy to identify and eliminate in a preprocessing step, using the Bellman–Ford algorithm. In the sequel we assume that this is already done.

[2] Leonid Khachiyan (personal communication) considered the similar Blocking Nonpositive Cycles problem, polynomial time equivalent to the decision problem for MPGs [4]. The question is whether MAX can stop MIN from reaching a cycle with nonpositive weight in a graph without sinks. The authors would appreciate any further references on the LSP problem.

above), by making iterative strategy improvements. Once a strategy is fixed, all shortest paths are easily computable, using the Bellman-Ford algorithm. Since there are negative weight edges, the Dijkstra algorithm does not apply. However, in Section 6 we briefly discuss an improvement over the straightforward application of the BF-algorithm.

Comparing a current choice made by the strategy with alternative choices, a possible improvement can be decided locally as follows. If changing the choice in a controlled vertex to another successor seems to give a longer distance (seems attractive), we make this change. Such a change is called a *switch.*

We prove two crucial properties (Theorems 2 and 3, respectively):

1. every attractive switch really increases the shortest distances, i.e., attractive is improving;
2. once none of the switches is attractive, the longest shortest paths are found, i.e., stable is optimal.

Although our algorithm proceeds by making just one attractive switch at a time, other algorithms making many choices simultaneously are also possible and fit into our framework; see Theorem 2.

Another interpretation of our algorithm is game-theoretic. MAX chooses in the controlled vertices, and the choices in all other vertices belong to MIN. For every strategy of MAX, MIN responds with an optimal counterstrategy, computing the shortest paths from every vertex to the sink. After that, the algorithm improves MAX's strategy by making an attractive switch, etc.

3.4 Randomization Scheme

The order in which attractive switches are made is essential for the subexponential complexity bound; see [4] for an example of an exponentially long chain of switches. The space of all positional strategies of MAX can be identified with the Cartesian product of sets of edges leaving the controlled vertices. Fixing any edge in this set and letting others vary determines a *facet* in this space. Now the algorithm for computing the longest-shortest paths in G looks as follows, starting from some strategy σ assumed to provide for a shortest distance $> -\infty$ from each vertex to the sink.

1. Randomly and uniformly select some facet F of G not containing σ. Throw this facet away, and recursively find a best strategy σ^* on what remains. This corresponds to deleting an edge not selected by σ and finding the best strategy in the resulting subgame.
2. If σ^* is optimal in G, stop (this is easily checked by testing whether there is an attractive switch from σ^* to F). The resulting strategy is globally optimal, providing for the longest-shortest distances.
3. Otherwise, switch to F, set $G = F$, denote the resulting strategy by σ, and repeat from step 1.

This is the well-known randomization scheme for linear programming due to Matoušek, Sharir, and Welzl [16,17]. When applied to the LSP and MPG problems, it gives a subexponential $2^{O(\sqrt{n \log n})}$ expected running time bound [16,3]. Another possibility would be to use the slightly more complicated randomization scheme of Kalai [14], as we did in [2] for parity games, which leads to the same subexponential complexity bound.

4 Retreats, Admissible Strategies, and Strategy Measure

As explained above, we modify an MPG by allowing MAX to "surrender" in every vertex. Add a retreat vertex t of MIN with a self-loop of weight 0 and a retreat edge of weight 0 from every vertex of MAX to t. Clearly, the same strategy (if any) secures MAX a value > 0 from a vertex in the original and modified games. Assume from now on that the retreat has been added to G. Intuitively, the "add retreats" transformation is useful because MAX can start by a strategy that selects the retreat edge in every vertex, thus "losing only 0" and satisfying the "additionally given" clause of the LSP problem in Section 3.

Definition 1. *A strategy σ of* MAX *in G is* admissible *if all loops in G_σ are positive, except the loop over t.* □

Our algorithm iterates only through admissible strategies of MAX. This guarantees that the only losing (for MAX) loop ever constructible is the one over t.

4.1 Measuring the Quality of Strategies

We now define a measure that evaluates the "quality" of an admissible strategy. It can be computed in strongly polynomial time; see Section 6 and [4].

Given an admissible strategy σ, the best MIN can do is to reach the 0-mean self-loop over t. Any other reachable loop will be positive, by the definition of an admissible strategy. The shortest path from every vertex v to t is well-defined, because there are no negative cycles in G_σ. This is suggestive for defining values.

Definition 2. *For an admissible strategy σ of* MAX, *the value $\mathrm{val}_\sigma(v)$ of vertex v is defined as the shortest path distance from v to t in G_σ, or $+\infty$ if t is not reachable. The value of a strategy is a vector of vertex values.* □

Note that for a fixed admissible strategy of MAX there is a positional counterstrategy of MIN (defined by the shortest paths forest) that guarantees the shortest paths from each vertex to the sink. The relative quality of two admissible strategies is defined componentwise in the strategy value.

Definition 3. *Let σ and σ' be two admissible strategies. Say that σ is* better than *σ', formally $\sigma > \sigma'$, if $\mathrm{val}_\sigma(v) \geq \mathrm{val}_{\sigma'}(v)$ for all vertices $v \in V$, with strict inequality for at least one vertex. Define $\sigma \geq \sigma'$, if $\sigma > \sigma'$ or they have equal values.* □

The following notation will be useful for describing switches.

Notation. If σ is a strategy of MAX, $x \in V_{\text{MAX}}$, and $(x, y) \in E$, then the *switch in x to y* changes σ to the new strategy $\sigma[x \mapsto y]$, defined as

$$\sigma[x \mapsto y](v) \stackrel{\text{def}}{=} \begin{cases} y, & \text{if } v = x; \\ \sigma(v), & \text{otherwise.} \end{cases}$$ □

We distinct between two kinds of switches later shown equivalent.

Definition 4. *Given an admissible strategy σ, a switch in vertex v to u is:*

1. attractive, *if* $w(v,u) + \text{val}_\sigma(u) > \text{val}_\sigma(v)$;
2. profitable, *if* $\sigma[v \mapsto u]$ *is admissible and* $\sigma[v \mapsto u] > \sigma$. □

4.2 Requirements for the Measure

The algorithm relies on the following properties.

1. An admissible strategy that has no better (Definition 3) strategy is winning from all vertices in MAX's winning set $G_{>0}$. This follows from definitions.
2. If a strategy has no profitable switches then it is optimal. This is shown in two steps (the two main theorems of the following section).
   a) Every attractive switch is also profitable (Theorem 2).
   b) If an admissible strategy has no attractive switches, then there is no better strategy (Theorem 3).

Properties (2a) and (2b) give another advantage: to find profitable switches, we only need to test attractivity, which is efficient as soon as the measure has been computed. Testing profitability would otherwise require recomputing the measure for every possible switch. In addition, we prove the following property, allowing an algorithm to change the strategy in more than one vertex at a time.

3. If several switches in an admissible strategy σ are attractive at the same time, then making any subset of them results in a better strategy (Theorem 2).

5 Correctness of the Measure

In this section we state the two major theorems (proved in [4]), guaranteeing that every step is improving and that the final strategy is the best, respectively.

Theorem 2. *If σ is an admissible strategy then any strategy obtained by one or more attractive switches is admissible and better. Formally, if the switches in s_i to t_i are attractive for $1 \leq i \leq r$ and $\sigma' \stackrel{\text{def}}{=} \sigma[s_1 \mapsto t_1][s_2 \mapsto t_2] \cdots [s_r \mapsto t_r]$, then σ' is admissible and $\sigma' > \sigma$.* □

We also show that every strategy without attractive switches is optimal. This guarantees that the strategy computed by the algorithm is indeed optimal.

Theorem 3. *If σ is an admissible strategy with no attractive switches, then $\sigma \geq \sigma'$ for all admissible strategies σ'.* □

The following corollary equates attractiveness and profitability.

Corollary 1. *If σ is an admissible strategy and an admissible strategy σ' is obtained from σ by one or more* non-attractive *switches, then $\sigma' \leq \sigma$. In particular, a single switch is attractive iff it is profitable.* □

6 Complexity

6.1 Efficient Computation of the Measure

The measure can be straightforwardly computed in time $O(n \cdot |E|)$ by using the Bellman–Ford algorithm for single-sink shortest paths in graphs with possibly negative weights. Since every vertex can have its shortest path length improved at most $O(n \cdot W)$ times, and there are n vertices, the number of switches cannot exceed $O(n^2 \cdot W)$. Together with the $O(n \cdot |E|)$ cost per iteration this gives total time $O(n^3 \cdot |E| \cdot W)$. In [4] we show how to reuse the measure computed in a previous iteration to improve this upper bound to $O(n^2 \cdot |E| \cdot W)$. The idea is to first compute which vertices will change their value, by an efficient backward reachability algorithm, and then run the Bellman–Ford algorithm only on the induced subgraph. Single iteration steps may still need $O(n \cdot |E|)$ time but the amortized time over all iterations is improved. There are no known examples for which the algorithm makes many improvement steps, and therefore the time per iteration becomes a practically significant part of the complexity.

6.2 Worst-Case Analysis

Without changing the asymptotic running time, the basic algorithm of Section 3 solving the *0-mean partition* problem can be extended [4] to solve the *p*-mean partition, for integer p, as well as a slightly more general problem of *splitting into three sets* [22] around an integer threshold p, with vertices of value $< p$, $= p$, and $> p$, respectively. By using the randomization scheme of Matoušek, Sharir, and Welzl from Section 3.4 we obtain the simultaneous bound $2^{O(\sqrt{n \log n})}$, *independent* of W [4].

Theorem 4. *The decision, p-mean partition, and splitting into three sets problems for mean payoff games can be solved in time*

$$\min\left(O(n^2 \cdot |E| \cdot W),\ 2^{O(\sqrt{n \log n})}\right),$$

and space linear in the size of the input. □

6.3 Computing the Ergodic Partitions

By using the standard dichotomy and approximation techniques [12,22] together with the p-mean partitioning algorithm above (extended to deal with rational thresholds; see [4]), we construct an algorithm for finding ergodic partitions in mean payoff games. First, we proceed by dichotomy until the value of each vertex is contained within a unit interval. After that, the threshold in calls to the basic algorithm will be non-integral. The minimal difference between two possible vertex values is $1/n(n-1)$ [12,22]. Thus the dichotomy process can stop as soon as every vertex value is contained in an interval of this size. The thresholds considered will never have denominators larger than n^2, and no vertex can improve its value more than $n^2 \cdot W$ times during one call to the basic algorithm [4]. We thus obtain

Theorem 5. *The ergodic partition problem for MPGs can be solved in time*

$$\min\left(O(n^3 \cdot |E| \cdot W \cdot \log(n \cdot W)),\quad (\log W) \cdot 2^{O(\sqrt{n \log n})}\right). \qquad \square$$

Zwick's and Paterson's algorithm for this problem has complexity $O(n^3 \cdot |E| \cdot W)$ [22, Theorem 2.3], which is slightly better for small W, but worse for large W.

6.4 NP∩coNP-Membership

The decision version of the LSP problem is restricted to determine whether the longest shortest path from a distinguished vertex s to the sink is bigger than a given bound. Together with MPGs and related games, this is another example of a problem in NP∩coNP [4].

Proposition 2. *The decision version of the LSP problem is in* NP∩coNP. $\square$

6.5 Exponential Sequences of Attractive Switches

Since it is not so easy to come up with "hard" LSP examples, one might conjecture that any sequence of attractive switches converges fast on any LSP problem instance, and consequently MPGs are quickly solvable by polynomially many iterative improvements. However, in [4] we present a family of instances of the LSP and MPG problems, which together with one specific improvement policy (method for selecting an attractive switch in every step), leads to exponentially long sequences of strategy improvements. This shows that the problems are non-trivial, and the choice of the next attractive switch is crucial for the efficiency.

6.6 Variants of the Algorithm

Theorem 2 shows that any combination of attractive switches improves the strategy value, and thus any policy for selecting switches in each iteration will eventually find an optimal strategy. In particular, all policies that have been suggested for parity and simple stochastic games apply. These include the all profitable, random single, and random multiple switch algorithms [1]. In [4] we suggest two alternative ways of combining policies.

6.7 Application to Parity Games

The algorithm described in this paper immediately applies to parity games, after the usual translation [20]. Parity games are similar to MPGs, but instead of weighted edges they have vertices colored in nonnegative integer colors. Player EVEN (MAX) wants to ensure that in every infinite play the largest color appearing infinitely often is *even,* and player ODD (MIN) tries to make it *odd.* Parity games are determined in positional strategies [11,5]. Our previous algorithm from [2] has complexity

$$\min\left(O(n^4 \cdot |E| \cdot k \cdot (n/k+1)^k),\ 2^{O(\sqrt{n \log n})}\right),$$

where n is the number of vertices, $|E|$ is the number of edges, and k is the number of colors. A direct analysis of the reduction from parity games to MPGs shows that our new algorithm improves on this bound by a factor n. An additional improvement is achieved by assigning smaller, compared to $(n/k+1)^k$, maximal weight translating a parity game to the MPG.

7 Conclusions

We defined the longest shortest paths (LSP) problem and applied it to create a discrete strategy measure and iterative improvement algorithms for mean payoff games. Similar measures were already known for parity [21,2], discounted payoff [20], and simple stochastic games [15], although not discrete for the last two classes of games. We showed that with our discrete measure any strategy iteration policy may be applied to solve mean payoff games, thus avoiding the difficulties of high precision rational arithmetic involved in reductions to discounted payoff and simple stochastic games, and solving associated linear programs.

Combining our strategy evaluation with the algorithm for combinatorial linear programming suggested by Matoušek, Sharir, and Welzl, yields a $2^{O(\sqrt{n \log n})}$ algorithm for the mean payoff game decision problem.

An interesting open question is whether the LSP problem is more general than mean payoff games, and if it has other applications. We showed that it belongs to NP∩coNP, and is solvable in expected subexponential randomized time.

The strategy measure presented does not apply to all strategies, only to admissible ones, which do not allow negative weight loops. This is enough for the algorithm, but it would be interesting to know if the measure can be modified or extended to the whole strategy space, and in this case if it would be completely local-global, like the measures for parity and simple stochastic games [3].

The major open problem is still whether there is a polynomial time strategy improvement scheme for the games discussed.

Acknowledgments. We thank DIMACS for providing a creative working environment. We are grateful to Leonid Khachiyan, Vladimir Gurvich, and Endre Boros for inspiring discussions and illuminating ideas. This paper is based on [4].

References

1. H. Björklund, S. Sandberg, and S. Vorobyov. Complexity of model checking by iterative improvement: the pseudo-Boolean framework. In M. Broy and A. Zamulin, editors, *Andrei Ershov Fifth International Conference "Perspectives of System Informatics",* volume 2890 of *Lecture Notes in Computer Science,* pages 381–394, 2003.
2. H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In H. Alt and M. Habib, editors, *20th International Symposium on Theoretical Aspects of Computer Science, STACS'2003,* volume 2607 of *Lecture Notes in Computer Science,* pages 663–674, Berlin, 2003. Springer-Verlag.
3. H. Björklund, S. Sandberg, and S. Vorobyov. Randomized subexponential algorithms for parity games. Technical Report 2003-019, Department of Information Technology, Uppsala University, April 2003. `http://www.it.uu.se/research/reports/`.
4. H. Bjorklund, S. Sandberg, and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. Technical Report DIMACS-2004-05, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, NJ, March 2004. `http://dimacs.rutgers.edu/TechnicalReports/`.
5. H. Björklund, S. Sandberg, and S. Vorobyov. Memoryless determinacy of parity and mean payoff games: A simple proof. *Theoretical Computer Science,* 310(1-3):365–378, January 2004.
6. A. Condon. The complexity of stochastic games. *Information and Computation,* 96:203–224, 1992.
7. A. Condon. On algorithms for simple stochastic games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 13:51–71, 1993.
8. A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journ. of Game Theory,* 8:109–113, 1979.
9. E. A. Emerson. Model checking and the Mu-calculus. In N. Immerman and Ph. G. Kolaitis, editors, *DIMACS Series in Discrete Mathematics,* volume 31, pages 185–214, 1997.
10. E. A. Emerson, C. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In C. Courcoubetis, editor, *Computer Aided Verification, Proc. 5th Int. Conference,* volume 697, pages 385–396. Lect. Notes Comput. Sci., 1993.
11. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Annual IEEE Symp. on Foundations of Computer Science,* pages 368–377, 1991.
12. V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *U.S.S.R. Computational Mathematics and Mathematical Physics,* 28(5):85–91, 1988.
13. A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Science,* 12(5):359–370, 1966.
14. G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC,* pages 475–482, 1992.
15. W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation,* 117:151–155, 1995.
16. J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. In *8th ACM Symp. on Computational Geometry,* pages 1–8, 1992.
17. J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. *Algorithmica,* 16:498–516, 1996.

18. H. Moulin. Extensions of two person zero sum games. *J. Math. Analysis and Applic.,* 55:490–508, 1976.
19. N. Pisaxuk. Mean cost cyclical games. *Mathematics of Operations Research,* 24(4):817–828, 1999.
20. A. Puri. *Theory of hybrid systems and discrete events systems.* PhD thesis, EECS Univ. Berkeley, 1995.
21. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In E. A. Emerson and A. P. Sistla, editors, *CAV'00: Computer-Aided Verification,* volume 1855 of *Lect. Notes Comput. Sci.,* pages 202–215. Springer-Verlag, 2000.
22. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.,* 158:343–359, 1996.

When Can You Play Positionally?*

Hugo Gimbert and Wiesław Zielonka

LIAFA, Université Paris 7, case 7014
2, Place Jussieu
75251 Paris Cedex 05, France
{hugo,zielonka}@liafa.jussieu.fr

Abstract. We consider infinite antagonistic games over finite graphs. We present conditions that, whenever satisfied by the payoff mapping, assure for both players positional (memoryless) optimal strategies. To verify the robustness of our conditions we show that all popular payoff mappings, such as mean payoff, discounted, parity as well as several other payoffs satisfy them.

1 Introduction

We study antagonistic (zero sum) games played on finite oriented graphs G by two players Max and Min. Each vertex of G belongs to one of the players. If the current game position is a vertex v then the owner of v chooses an outgoing edge e and the target of e becomes a new game position. After an infinite number of moves we obtain an infinite path p in G that we call a play.

We suppose that the edges of G are coloured by elements of some set C. Then each play yields an infinite sequence of colours of the edges traversed during the play. The payoff function indicates for each such infinite sequence of colours a real number: the amount that player Min pays to player Max at the end of the game.

Since the seminal paper of Shapley[12], this type of games is studied in game theory even in a much more general setting of stochastic games.

In general, optimal strategies of both players can depend on the whole past history. However, it turns out that for many games both players have optimal positional strategies, i.e. optimal strategies where the players' moves depend only on the current position. This type of strategies is particularly interesting in computer science since positional strategies allow us an easy and efficient implementation. Moreover, since the number of possible positional strategies is finite it is always possible to find optimal strategies by exhaustive search — note however that for most of the games we have much more efficient algorithms, see [11] for a recent survey of algorithmic problems.

Motivated by economic applications classical game theory studies mainly, but not exclusively, two payoff functions: mean-payoff and discounted [6]. The games

* This research was supported by European Research Training Network: Games and Automata for Synthesis and Validation

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 686–697, 2004.
© Springer-Verlag Berlin Heidelberg 2004

with limsup payment are studied in the framework of gambling systems [8]. Since the paper of Shapley [12] it is known that even for stochastic discounted games both players have positional optimal strategies.

The existence of optimal positional strategies for mean-payoff deterministic games was established by Moulin [10] and Mycielski and Świerczkowski[4].

Let us note that recently discounted and even mean payoff games entered also in computer science, see [3,2] for a nice exposition of the motivations behind discounting system properties.

Parity games have much more recent history. They appear in the work of Emerson and Jutla [5] in relation to the modal μ-calculus and in Mostowski [9] in relation to the problem of determinizing finite automata over infinite trees (Rabin theorem). Again these games admit optimal positional strategies (even over infinite graphs).

As noted by Björklund et al. [1] it is possible to present highly similar proofs of the existence of optimal positional strategies for mean-payoff and parity games. Let us note however that [1] failed to extract explicitly the ingredients of both proofs that make them so similar. One can only guess that there are some common axioms hidden in the proof. Moreover the inductive method presented in [1] fails for discounted games.

All these results give rise to the following general question: what conditions should satisfy the payment mapping to assure the existence of optimal positional strategies for both players? The aim of our paper is to examine this problem in the simplest setting of deterministic perfect information games over finite graphs. Ideally, one would like to have conditions that are both sufficient and necessary.

In our paper we were able to provide a set of three conditions that are only sufficient. However, they seem to be quite robust in practice: virtually all payoff mappings that were discussed previously in the literature and that admit positional optimal strategies turn out to satisfy our conditions. It would be interesting to have more natural examples of "memoryless" payoff mappings to test the robustness of our conditions, we provide one such new mapping that can be of independent interest. We checked also our conditions against numerous trivial payoff mappings.

On the other hand, one may ask when our conditions fail, i.e., when a payoff mapping yielding optimal positional strategies does not satisfy our conditions. We have found one such example, however even in this case we can use our conditions indirectly to establish the existence of optimal positional strategies.

The conditions that we provide look also quite "natural", the examples of payoff mappings that are interesting in the context of the perfect information deterministic games over finite graphs but which do not satisfy our conditions are rare: the payoff mapping mentioned above is rather artificial, another prominent example are the games with Muller condition [7] but these games need memory.

Let us note finally that the inductive method developed in our paper was successfully adapted by the second author to perfect information stochastic games where it allowed us to show in a simpler way the existence of optimal positional pure strategies [13] for parity games.

An intriguing open question is whether our three conditions assure the existence of optimal positional pure strategies for perfect information *stochastic* games.

2 Preliminaries

For any (possibly infinite) set C, we write C^+ to denote the set of all finite non-empty words over C. An infinite word $c_0c_1c_2\ldots$ over C is said to be *finitely generated* if there is a finite subset A of C such that for all i, $c_i \in A$. In our paper C^ω will stand always for a set of all *finitely generated infinite words* over C. Let us note that this is a departure from the standard notation where C^ω stands usually for the set of all infinite sequences (infinite words) over C. The difference appears of course only if C is infinite. However, in our paper while it is useful to allow infinite alphabets, only finitely generated sequences are of interest.

For $x \in C^+$, by $x^\omega = xxx\ldots$ we note the infinite concatenation of words x. We use also the standard mathematical notation: in particular for any sequence $a_n, n \geq 0$, of real numbers, $\limsup_n a_n = \lim_{n\to\infty} \sup_{i\geq n} a_i$.

$\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ will denote the extended set of real numbers.

An *arena* is a tuple $G = (V_{\text{Max}}, V_{\text{Min}}, E, C, \varphi)$, where $(V_{\text{Max}} \cup V_{\text{Min}}, E)$ is a finite oriented graph with the set $V = V_{\text{Max}} \cup V_{\text{Min}}$ of vertices partitioned onto the set V_{Max} of vertices of player Max and the set V_{Min} of vertices belonging to player Min. $E \subseteq V \times V$ is the set of edges. We shall colour edges by means of a mapping $\varphi : E \to C$ which associates with each edge $e \in E$ a colour $\varphi(e) \in C$. Although the set of colours will be often infinite (for example $\mathbb{R}$ or $\mathbb{N}$) actually only finite subsets of C will be used since we restrict our attention to finite arenas.

For any edge $e = (v, w) \in E$ we call v the *source* and w the *target* of e and note $\mathbf{source}(e) = v$ and $\mathbf{target}(e) = w$. For a vertex $v \in V$, by $vE = \{e \in E \mid \mathbf{source}(e) = v\}$ we denote the set of outgoing edges.

We suppose that arenas have no dead-ends, i.e. each vertex has at least one outgoing edge.

A *path* in G is a finite or infinite sequence of edges $p = e_0e_1e_2\ldots$ such that, for all $i \geq 0$, $\mathbf{target}(e_i) = \mathbf{source}(e_{i+1})$. The source of p is the source of the first edge e_0. If p is finite then $\mathbf{target}(p)$ is the target of the last edge in p.

Players Max and Min play on G in the following way. If the current game position is a vertex $v \in V_{\text{Max}}$ then player Max chooses an outgoing edge $e \in vE$ and vertex $\mathbf{target}(e)$ becomes a new game position. Otherwise, if the current game position v belongs to player Min, $v \in V_{\text{Min}}$, then Min chooses an outgoing edge $e \in vE$ and vertex $\mathbf{target}(e)$ becomes a new game position. If the initial position was v then in this way the players construct an infinite path $\mathbf{e} = e_0e_1e_2\ldots$ of visited edges such that $\mathbf{source}(\mathbf{e}) = v$. Such an infinite path will be called sometimes a play in G. The set of all plays (infinite paths) in G will be denoted P_G^ω. The set of finite paths in G will be noted P_G^*. Elements of P_G^* will be sometimes

called *histories* or finite plays, especially when they are used to encode the history of all movements of both players up to a current moment. It is convenient to assume that for each vertex v there exists an empty path $\lambda_v \in P_G^*$ with no edges and such that $\mathrm{source}(\lambda_v) = \mathrm{target}(\lambda_v) = v$.

With any play $\mathbf{e} = e_0e_1e_2\ldots$ we associate the payoff sequence $\varphi(\mathbf{e}) = \varphi(e_0)\varphi(e_1)\varphi(e_2)\ldots$ of visited colours. Note that we have extended in this way the colouring mapping to $\varphi : P_G^\omega \to C^\omega$. In a similar way, we set for a finite path $p = e_0 \ldots e_k$, $\varphi(p) = \varphi(e_0)\ldots\varphi(e_k)$.

A *payoff function* is a mapping

$$u : C^\omega \to \overline{\mathbb{R}}.$$

Intuitively, after an infinite play p player Min pays to player Max the amount $u(\varphi(p))$ (with the natural interpretation that if $u(\varphi(p)) < 0$ then it is rather player Max that pays to player Min the amount $|u(\varphi(p))|$). Let us note that since our arenas are finite $\varphi(p)$ will be always finitely generated for any infinite path p. This is sometimes important since the definitions of some of the payoff mappings u are meaningless for infinite colour words that are not finitely generated.

A *game* is a couple $\mathbf{G} = (G, u)$ made of an arena G and a payoff function.

Let G be an arena and $\mathfrak{p} \in \{\mathrm{Min}, \mathrm{Max}\}$ a player. A *strategy* for player $\mathfrak{p}$ in G is a mapping $\sigma_\mathfrak{p}$ which indicates for each finite history p such that $\mathrm{target}(p) \in V_\mathfrak{p}$ an edge outgoing from $\mathrm{target}(p)$ that player $\mathfrak{p}$ should choose. Thus formally

$$\sigma_\mathfrak{p} : \{p \in P_G^* \mid \mathrm{target}(p) \in V_\mathfrak{p}\} \to E,$$

where $\sigma_\mathfrak{p}(p) \in vE$ whenever $v = \mathrm{target}(p)$.

A finite or infinite path $\mathbf{e} = e_0e_1e_2\ldots$ is said to be *consistent* with the strategy $\sigma_\mathfrak{p}$ of player $\mathfrak{p}$ if, for all i, whenever $\mathrm{source}(e_i) \in V_\mathfrak{p}$ then $e_i = \sigma_\mathfrak{p}(e_0 \ldots e_{i-1})$.

$\Sigma_\mathfrak{p}$ denotes the set of strategies of a player $\mathfrak{p}$.

In this paper we are especially interested in the class of positional strategies.

A *positional* strategy for player $\mathfrak{p}$ is a mapping $\sigma_\mathfrak{p} : V_\mathfrak{p} \to E$ such that for all $v \in V_\mathfrak{p}$, $\sigma_\mathfrak{p}(v) \in vE$. Intuitively, when $\mathfrak{p}$ uses a positional strategy then his choice of the outgoing edge depends only on the current game position and not on the previous history.

Given a vertex v and strategies σ and τ for player Max and player Min respectively, there exists a unique play starting in v and consistent with σ and τ. This play is denoted by $p^G(v, \sigma, \tau)$.

Strategies $\sigma^\sharp \in \Sigma_{\mathrm{Max}}$ and $\tau^\sharp \in \Sigma_{\mathrm{Min}}$ are said to be *optimal* if for all vertices $v \in V$ and all strategies $\sigma \in \Sigma_{\mathrm{Max}}$ and $\tau \in \Sigma_{\mathrm{Min}}$

$$u(\varphi(p^G(v, \sigma, \tau^\sharp))) \leq u(\varphi(p^G(v, \sigma^\sharp, \tau^\sharp))) \leq u(\varphi(p^G(v, \sigma^\sharp, \tau))). \tag{1}$$

If $\sigma^\sharp$ and $\tau^\sharp$ are optimal then the quantity $u(\varphi(p^G(v, \sigma^\sharp, \tau^\sharp)))$ is called the *value* of the game $\mathbf{G}$ at v and is noted $\mathrm{val}(\mathbf{G}(v))$. Let us note that in our definition we require for positional strategies to be optimal independently of the starting vertex.

3 Fairly Mixing Payoffs Yield Positional Strategies

The aim of this section is to present sufficient conditions for the payoff mapping u assuring the existence of optimal positional strategies for both players.

Definition 1. *A payoff mapping $u : C^\omega \to \overline{\mathbb{R}}$ is said to be* fairly mixing *if the following conditions hold*

(C1) for all $x \in C^+, y_0, y_1 \in C^\omega$, if $u(y_0) \leq u(y_1)$ then $u(xy_0) \leq u(xy_1)$,
(C2) for all $x \in C^+, y \in C^\omega$, $\min\{u(x^\omega), u(y)\} \leq u(xy) \leq \max\{u(x^\omega), u(y)\}$,
(C3) Suppose that (x_i), $i \in \mathbb{N}$, is an infinite sequence of non-empty colour words $x_i \in C^+$ such that the infinite word $x_0x_1x_2\ldots$ is finitely generated. Let $I \cup J = \mathbb{N}$ be any partition of $\mathbb{N}$ onto two infinite sets I and J ($I \cap J = \emptyset$). Let U_I be the set of all payoffs $u(x_{i_0}x_{i_1}x_{i_2}\ldots)$, where $(i_k)_{k=0}^\infty$ ranges over all sequences of elements of I. Similarly, let U_J be the set of all payoffs $u(x_{j_0}x_{j_1}x_{j_2}\ldots)$, where $(j_k)_{k=0}^\infty$ ranges over all infinite sequences of elements of J. Then the following condition holds:

$$\inf(U_I \cup U_J) \leq u(x_0x_1x_2x_3\ldots) \leq \sup(U_I \cup U_J). \tag{2}$$

Let us emphasize that in (C3) we suppose that (2) holds for any partition of $\mathbb{N}$ onto infinite sets I and J. Moreover, in the definition of the set U_I we use really *all* possible infinite sequences of elements of I, we do not limit ourselves to, for example, increasing sequences, neither we assume that the sequence $(i_k)_{k=0}^\infty$ enumerates all elements of I. The same remark is valid for the definition of U_J.

Theorem 1. *Let $u : C^\omega \to \overline{\mathbb{R}}$ be a fairly mixing payoff function. Then both players have optimal positional strategies in any game $\mathbf{G} = (G, u)$ over a finite arena G coloured by C..*

For the sake of simplicity the following lemma is formulated only for player Max, however it should be clear that analogous characterisation holds also for player Min. This point will be discussed briefly in the sequel.

Lemma 1. *Let $\mathbf{G} = (G, u)$ be a game with fairly mixing payoff function u. Suppose that*

(1) there exists a vertex $v \in V_{\mathrm{Max}}$ such that $|vE| > 1$, where $vE = \{e \in E \mid \mathrm{source}(e) = v\}$ is the set of all edges with the source v,
(2) $vE = A_1 \cup A_2$ is a partition of the set vE onto two non-empty sets A_1 and A_2,
(3) $E' = \{e \in E \mid \mathrm{source}(e) \neq v\}$, $E_i = E' \cup A_i$ and $G_i = (V_{\mathrm{Max}}, V_{\mathrm{Min}}, E_i, C, \varphi)$, $i = 1, 2$, are arenas obtained from G by keeping only the edges of E_i (i.e., by removing the edges of A_{3-i}) and keeping all vertices V,
(4) players Max *and* Min *have optimal positional strategies in the games $\mathbf{G}_1 = (G_1, u)$ and $\mathbf{G}_2 = (G_2, u)$.*

Then Max *and* Min *have optimal strategies* $\sigma^\sharp$ *and* $\tau^\sharp$ *in the game* $\mathbf{G} = (G, u)$. *More exactly, we can assume without loss of generality that*

$$\mathrm{val}(\mathbf{G}_2(v)) \leq \mathrm{val}(\mathbf{G}_1(v)) \tag{3}$$

and then

(i) the optimal positional strategy of player Max *in* $\mathbf{G}_1$ *is also optimal for the same player in the game* $\mathbf{G}$,

(ii) for all $w \in V$, $\mathrm{val}(\mathbf{G}(w)) = \mathrm{val}(\mathbf{G}_1(w))$.

Proof. Let $\sigma_i^\sharp$, $\tau_i^\sharp$ be optimal positional strategies of players Max and Min respectively in the games $\mathbf{G}_i = (G_i, u)$, $i = 1, 2$. We then set

$$\sigma^\sharp = \sigma_1^\sharp . \tag{4}$$

It is clear that $\sigma^\sharp$ is a positional strategy for Max not only in the game $\mathbf{G}_1$ but also in the game $\mathbf{G}$. We claim that $\sigma^\sharp$ is optimal for Max in the game $\mathbf{G} = (G, u)$ and that for all vertices $w \in V$, $\mathrm{val}(\mathbf{G}(w)) = \mathrm{val}(\mathbf{G}_1(w))$.

The following remark holds under the hypotheses of Lemma 1 and conditions (3) and (4):

Remark 1. Let τ be any strategy of Min in the game $\mathbf{G}$ and w an initial vertex. Then $u(\varphi(p^G(w, \sigma^\sharp, \tau))) \geq \mathrm{val}(\mathbf{G}_1(w))$, i.e., for the games starting at w the strategy $\sigma^\sharp$ can assure to player Max the payoff of at least $\mathrm{val}(\mathbf{G}_1(w))$.

Indeed, if we restrict the strategy τ to finite paths in G_1 then we obtain a strategy τ_1 of Min in the game $\mathbf{G}_1$. Now note that if player Max plays according to $\sigma^\sharp = \sigma_1^\sharp$ then he never uses the edges of A_2 and the resulting game looks for his adversary like the game on G_1. In particular, the play in $\mathbf{G}$ that is consistent with $\sigma^\sharp$ and τ is the same as the play in $\mathbf{G}_1$ consistent with $\sigma_1^\sharp = \sigma^\sharp$ and τ_1, $p^G(w, \sigma^\sharp, \tau) = p^{G_1}(w, \sigma_1^\sharp, \tau_1) \geq \mathrm{val}(\mathbf{G}_1(w))$, which proves Remark 1.

To finish the proof of Lemma 1 we should construct a strategy $\tau^\sharp$ for player Min assuring that for any strategy σ of player Max in $\mathbf{G}$ and any initial vertex w:

$$u(\varphi(p^G(w, \sigma, \tau^\sharp))) \leq \mathrm{val}(\mathbf{G}_1(w)), \tag{5}$$

i.e., player Max cannot win more that $\mathrm{val}(\mathbf{G}_1(w))$ in the game $\mathbf{G}$ against the strategy $\tau^\sharp$.

We define first a mapping $b : P_G^* \to \{1, 2\}$. For a finite path $p \in P_G^*$ in G we set

$$b(p) = \begin{cases} 1 & \text{if either } p \text{ does not contain any edge with the source } v \text{ or} \\ & \text{the last edge of } p \text{ with the source } v \text{ belongs to } G_1, \\ 2 & \text{if the last edge of } p \text{ with the source } v \text{ belongs to } G_2. \end{cases} \tag{6}$$

Then the strategy $\tau^\sharp$ of Min in $\mathbf{G}$ is defined as:

$$\tau^\sharp(p) = \begin{cases} \tau_1^\sharp(\mathrm{target}(p)) & \text{if } b(p) = 1 \\ \tau_2^\sharp(\mathrm{target}(p)) & \text{if } b(p) = 2 \end{cases} \tag{7}$$

for any finite path p with $\mathrm{target}(p) \in V_{\mathrm{Min}}$. In other words, playing in $\mathbf{G}$ player Min applies either his optimal strategy $\tau_1^\sharp$ from the game $\mathbf{G}_1$ or his optimal strategy $\tau_2^\sharp$ from the game $\mathbf{G}_2$. Initially, up to the first visit to v, player Min uses the strategy $\tau_1^\sharp$. After the first visit at v the choice between $\tau_1^\sharp$ and $\tau_2^\sharp$ depends on whether the last time when visiting v his adversary Max chose an outgoing edge in E_1 or an edge in E_2. Intuitively, if the last time at v player Max chose an outgoing edge from E_1 then it means that the play from this moment is like a play in $\mathbf{G}_1$ thus player Min tries to respond with his optimal strategy from $\mathbf{G}_1$. Symmetrically, if during the last visit at v player Max chose an outgoing edge from E_2 then from this moment onward the play is like a play in $\mathbf{G}_2$ and player Min tries to counter with his optimal strategy from $\mathbf{G}_2$.

It should be clear that the strategy $\tau^\sharp$ needs in fact just two valued memory $\{1, 2\}$ for player Min to remember if during the last visit to v an edge of E_1 or an edge of E_2 was chosen by his adversary. This memory is initialised to 1 and updated only when the vertex v is visited.

In the sequel we shall say that a finite or infinite path p in G is *homogeneous* if one of the following three conditions folds: (1) p never visits v or (2) each edge e of p with source v belongs to E_1 or (3) each edge e of p with source v belongs to E_2.

The proof of (5) is divided on four cases.

Case 1: $w = v$ and the memory state of player Min **is ultimately constant during the play $p = p^G(v, \sigma, \tau^\sharp)$.**
This means that p can be factorised as $p = p_0 p_1 \ldots p_n q$, where p_i are finite non-empty homogeneous paths such that $\mathrm{source}(p_i) = \mathrm{target}(p_i) = v$ and q is an infinite homogeneous path with source v.

Since p is consistent with $\tau^\sharp$ and p_i are homogeneous, each infinite play $p_i^\omega = p_i p_i p_i \ldots$ is consistent either with $\tau_1^\sharp$ (if p_i contains only the edges of G_1) or with $\tau_2^\sharp$ (if p_i contains only edges of G_2).

By optimality of strategies $\tau_1^\sharp$ and $\tau_2^\sharp$, we get that either $u(\varphi(p_i^\omega)) \leq \mathrm{val}(\mathbf{G}_1(v))$ or $u(\varphi(p_i^\omega)) \leq \mathrm{val}(\mathbf{G}_2(v))$. Thus, by (3),

$$\text{for all } i,\ 0 \leq i \leq n, \quad u(\varphi(p_i^\omega)) \leq \mathrm{val}(\mathbf{G}_1(v)). \tag{8}$$

Similarly, the infinite path q is consistent either with $\tau_1^\sharp$ or with $\tau_2^\sharp$ implying that $u(\varphi(q))$ cannot be greater than either $\mathrm{val}(\mathbf{G}_1(v))$ or $\mathrm{val}(\mathbf{G}_2(v))$. Thus, again by (3),

$$u(\varphi(q)) \leq \mathrm{val}(\mathbf{G}_1(v)) \tag{9}$$

From (C2) of Definition 1, by a trivial induction on n, we can deduce that $u(x_0 \ldots x_n y) \leq \max\{u(x_0^\omega), \ldots, u(x_n^\omega), u(y))\}$ for any $x_i \in C^+$ and $y \in C^\omega$.

This inequality, and (8) and (9) imply

$$u(\varphi(p)) = u(\varphi(p_0) \ldots \varphi(p_n)\varphi(q)) \leq \max\{u(\varphi(p_0)^\omega), \ldots, u(\varphi(p_n)^\omega), u(\varphi(q))\} \leq \mathrm{val}(\mathbf{G}_1(v)).$$

Case 2: $w = v$ and the memory state of player Min changes infinitely often during the play $p = p^G(v, \sigma, \tau^\sharp)$.

Thus, in particular, p visits infinitely often the vertex v. Let $p = p_0 p_1 p_2 \ldots$ the the unique factorization of p such that for all $i \in \mathbb{N}$: (1) each finite path p_i is non-empty and (2) $\mathrm{source}(p_i) = \mathrm{target}(p_i) = v$ and the first edge of p_i is the only edge of p_i with the source v.

We set $I = \{i \in \mathbb{N} \mid p_i \text{ begins with an edge belonging to } A_1\}$ and $J = \{j \in \mathbb{N} \mid p_j \text{ begins with an edge belonging to } A_2\}$. Thus $I \cup J = \mathbb{N}$ forms a partition of $\mathbb{N}$.

For any sequence $(i_k)_{k=0}^{\infty}$ of elements of I the infinite concatenation $p_{i_0} p_{i_1} p_{i_2} \ldots$ of finite paths is in fact an infinite path in the arena G_1. Moreover, directly from the definition of $\tau^\sharp$ it follows that during the passage through p_{i_k} in p player Min played using the strategy $\tau_1^\sharp$ and therefore $p_{i_0} p_{i_1} p_{i_2} \ldots$ is consistent with $\tau_1^\sharp$ implying that $u(\varphi(p_{i_0} p_{i_1} p_{i_2} \ldots)) \leq \mathrm{val}(\mathbf{G}_1(v))$ by optimality of $\tau_1^\sharp$ in $\mathbf{G}_1$.

A similar reasoning shows that for any sequence $(j_k)_{k=0}^{\infty}$ of elements of J we have $u(\varphi(p_{j_0} p_{j_1} p_{j_2} \ldots)) \leq \mathrm{val}(\mathbf{G}_2(v))$.

These two bounds imply, by (C3) and (3), that
$u(\varphi(p)) \leq \max\{\mathrm{val}(\mathbf{G}_1(v)), \mathrm{val}(\mathbf{G}_2(v))\} = \mathrm{val}(\mathbf{G}_1(v))$.

Case 3: $w \neq v$ and the play $p = p^G(w, \sigma, \tau^\sharp)$ never visits the vertex v.

If we define σ_1 to be the restriction of the strategy σ to the paths in G_1 then σ_1 is a strategy of Max in G_1. Moreover, since v is never visited, player Min using $\tau^\sharp$ in fact applies always the strategy $\tau_1^\sharp$, which is optimal for him in $\mathbf{G}_1$. Thus p can be seen as a play in G_1 consistent with σ_1 and with $\tau_1^\sharp$ and by optimality of $\tau_1^\sharp$ we get $u(\varphi(p)) \leq \mathrm{val}(\mathbf{G}_1(w))$.

Case 4: $w \neq v$ and the play $p = p^G(w, \sigma, \tau^\sharp)$ visits at least once the vertex v.

Let us factorise p: $p = rq$, where r is the finite prefix of p until the first visit to v, i.e. r is the shortest prefix of p with $\mathrm{target}(r) = v$.

Let $q^\sharp = p^{G_1}(v, \sigma_1^\sharp, \tau_1^\sharp)$, where $\sigma_1^\sharp$ is the optimal positional strategy of Max in $\mathbf{G}_1$. Thus, since both $\sigma_1^\sharp$ and $\tau_1^\sharp$ are optimal in $\mathbf{G}_1$, we have $u(\varphi(q^\sharp)) = \mathrm{val}(\mathbf{G}_1(v))$.

Now note that q is in fact a play in G starting at v and consistent with $\tau^\sharp$. This situation was already examined above (Cases 1 and 2) and we have learned there that $u(\varphi(q)) \leq \mathrm{val}(\mathbf{G}_1(v))$.

In this way we get $u(\varphi(q)) \leq \mathrm{val}(\mathbf{G}_1(v)) = u(\varphi(q^\sharp))$ which, by condition (C1) of fairly mixing payoffs, yields

$$u(\varphi(r)\varphi(q)) \leq u(\varphi(r)\varphi(q^\sharp)) = u(\varphi(rq^\sharp)). \tag{10}$$

However, $rq^\sharp$ is an infinite play in G_1 starting at w and consistent with $\tau_1^\sharp$ (r is consistent with $\tau_1^\sharp$ since until the first visit to v in $p = rq$ player Min plays according to $\tau_1^\sharp$ while $q^\sharp$ is consistent with $\tau_1^\sharp$ just by definition). Thus, by optimality of $\tau_1^\sharp$ in $\mathbf{G}_1$, we get $u(\varphi(rq^\sharp)) \leq \mathrm{val}(\mathbf{G}_1(w))$. This and (10) imply $u(\varphi(p)) = u(\varphi(rq)) \leq \mathrm{val}(\mathbf{G}_1(w))$.

This concludes the proof of Lemma 1.

□

Before applying Lemma 1 let us note what happens if all hypotheses of Lemma 1 are satisfied except that the vertex v belongs rather to player Min. Suppose also that, as in Lemma 1, $\mathrm{val}(\mathbf{G}_2(v)) \leq \mathrm{val}(\mathbf{G}_1(v))$. Then it is the optimal strategy $\tau_2^\sharp$ of Min from $\mathbf{G}_2$ that is optimal in $\mathbf{G}$. This can be deduced immediately from Lemma 1 since player Min becomes the maximizing player for the payoff $-u$ (and $-u$ is fairly mixing iff u is).

Proof of Theorem 1

Let $G = (V_{\mathrm{Max}}, V_{\mathrm{Min}}, E, C, \varphi)$ be a finite arena and $\mathbf{G} = (G, u)$ with fairly mixing payoff u.

We prove the theorem by induction on $n_G = |E| - |V|$.

If $n_G = 0$ then, since G has no dead ends, each vertex of V has only one outgoing edge. Thus the players have no choice and there is only one possible strategy for each of them and such a strategy is positional and, obviously, also optimal.

Let $n_G > 0$ and suppose that the thesis holds for each game $\mathbf{G}'$ over an arena such that $n_{G'} < n_G$.

If all vertices $v \in vM$ of player Max have only one outgoing edge then Max has only one possible strategy and this strategy is positional. Obviously, this unique strategy is also optimal for Max.

Now suppose that there exists $v \in vM$ having at least two outgoing edges. We decompose G onto two subarenas G_1 and G_2 exactly as in Lemma 1. Since G_1 and G_2 have the same number of vertices as G but their number of edges is strictly less than $|E|$ we can apply to $\mathbf{G}_i = (G_i, u)$, $i = 1, 2$, the induction hypothesis to deduce the existence of optimal positional strategies $\sigma_i^\sharp$ for Max in $\mathbf{G}_i$, $i = 1, 2$. Again by Lemma 1, either $\sigma_1^\sharp$ or $\sigma_2^\sharp$ is an optimal positional strategy of Max in $\mathbf{G}$, depending on whether $\mathrm{val}(\mathbf{G}_2(v)) \leq \mathrm{val}(\mathbf{G}_1(v))$ or the inverse inequality holds.

The existence of an optimal positional strategy for player Min follows by a symmetric argument.

□

4 Applications

In this section we show that virtually all popular (as well as some less popular) payoff mappings satisfy conditions (C1)-(C3). This implies immediately

the existence of positional optimal strategies due to Theorem 1. Due to space restrictions the proofs, in most cases rather straightforward, are omitted.

If not stated explicitly otherwise, in the examples examined below we suppose that $C = \mathbb{R}$, i.e. the edges are labelled by real numbers. In particular, $\mathbb{R}^+$ will stand always for the set of non-empty finite sequences of real numbers and $\mathbb{R}^\omega$ is the set of finitely generated infinite sequences of reals.

Sup Game. Max wins the highest value seen during the play, i.e. the payoff is

$$u_s(c_0c_1\ldots) = \sup\{c_0, c_1, \ldots\}, \quad \text{where } c_i \in C.$$

Limsup Game. Now we suppose that Max wins the highest value seen infinitely often during the play, i.e. the payoff is given by

$$u_l(c_0c_1\ldots) = \limsup_i c_i.$$

This payoff is used for example in gambling systems [8].

Total Reward Game. In the total reward game player Max accumulates the one day payoffs:

$$u_t(c_0c_1\ldots) = \limsup_n \sum_{i=0}^{n} c_i.$$

Note that in this case the payoff can take infinite values $\pm\infty$. This type of payoffs is classical in game theory [6].

Parity Game. $C = \mathbb{N}$ is the set of non-negative integers. The payoff is defined as

$$u_p(c_0c_1\ldots) = (\limsup_{i\in\mathbb{N}} c_i) \mod 2$$

In other words, player Max wins 1 if the highest colour visited infinitely often is odd, otherwise his payoff is 0. This is the most relevant type of payoff for computer science, [5].

Weighted Reward Game. $C = \mathbb{R}$ is again the set of real numbers. The payoff is given by

$$u_\lambda(c_0c_1\ldots) = \lambda \cdot \liminf_{i\in\mathbb{N}} c_i + (1-\lambda) \cdot \limsup_{i\in\mathbb{N}} c_i,$$

where $\lambda \in [0,1]$ is any fixed constant from the closed interval [0,1].

The interpretation for u_λ is the following. If c_i is the capital of player Max on the day i then using the coefficient λ he can weight relatively his good and bad fortune. For example, for payoffs with λ close to 1 the bad fortune counts more than the good one. This type of payoff is a natural extension of Limsup payoff but it seems not be considered before.

Note also that the payoff $u = 2 \cdot u_{\frac{1}{2}}$ can be seen as a generalisation of the parity payoff. To see this let us take a parity game $\mathbf{G}$ with an underlying arena

G. Let us replace in G each odd label c by $-c$ and now consider over this modified arena the game $\mathbf{G}' = (G', u)$ with the payoff u defined above. Now it suffices to note that if the game value of $\mathbf{G}'(s)$ is non-negative then in the original parity game $\mathbf{G}(s)$ it is the player 0 that wins. On the other hand, if the game value of $\mathbf{G}'(s)$ is negative then in the original parity game $\mathbf{G}(s)$ it is the player 1 that wins. The game $\mathbf{G}'$ can be seen as quantitative version of the parity game. In the parity game we examine only if the maximal infinitely often visited colour is even or odd, in the game $\mathbf{G}'$ we measure more precisely the "distance" between the greatest even and odd colours visited infinitely often.

Mean Payoff Game. Again $C = \mathbb{R}$. With any finite sequence $x \in \mathbb{R}^+$ of elements of $\mathbb{R}$ we associate their mean value

$$\mathrm{mean}(x) = \frac{1}{|x|}\mathfrak{S}(x),$$

where $\mathfrak{S}(x)$ denotes the sum of all elements of x while $|x|$ stands for the length of x. The mean payoff mapping is defined by

$$u_m(c_0c_1\ldots) = \limsup_{n\in\mathbb{N}}(\mathrm{mean}(c_0\ldots c_{n-1}))$$

Let us note that the apparently similar game with the payoff $g(c_0c_1\ldots) = \sup_{n\in\mathbb{N}}(\mathrm{mean}(c_0\ldots c_{n-1}))$ does not satisfy (C1) and in fact there are finite graphs for which both players need a memory to play optimally for the payoff g.

Discounted Game. The set of colours is $C = [0,1)\times\mathbb{R}$. For any finitely generated infinite word $(\lambda_0, a_0)(\lambda_1, a_1)\ldots \in C^\omega$ we set

$$u_d((\lambda_0, a_0)(\lambda_1, a_1)\ldots) = \lambda_0a_0 + \lambda_0\lambda_1a_1 + \lambda_0\lambda_1\lambda_2a_2 + \cdots$$

Usually in discounted games there is one discount factor λ and then $u_d(a_0a_1\ldots) = \sum_{i\geq 0}\lambda^i a_i$. Allowing different discount factors is in the spirit of the original paper of Shapley [12].

Theorem 2. *All payoff mappings listed above satisfy conditions (C1)-(C3) and therefore both players have optimal positional strategies.*

Let us note finally that the conditions of Definition 1 are not necessary. Let $(c_i)_{i=0}^{\infty}$ be a finitely generated sequence of real numbers. Consider the payoff $f(c_0c_1c_2\ldots) = \mathrm{sgn}(\inf_{n\geq 0}\sum_{i=0}^{n} c_i)$, where $\mathrm{sgn}(x)$ is equal either -1, or 0, or 1 depending on whether x is negative, zero or positive. This payoff mapping admits optimal positional strategies but does not satisfy (C1). Nevertheless, even in this case we can use indirectly Theorem 1. Indeed, the payoff $g(c_0c_1\ldots) = \inf_{n\geq 0}\sum_{i=0}^{n} c_i$ is fairly mixing. Now note the following trivial observation:

Remark 2. If a payoff u admits optimal positional strategies and and $h : \mathbb{R}\to\mathbb{R}$ is a non-decreasing mapping then the composition $h\circ u$ admits optimal positional strategies, in fact the strategies optimal for u remain optimal for $h\circ u$.

Thus from the fact that g is fairly mixing and that sgn is non-decreasing we can conclude from Remark 2 that $f = \mathrm{sgn} \circ g$ admits optimal positional strategies.

This example leads immediately to the following question:

Let u be a payoff mappings admitting optimal positional strategies for both players for all finite arenas. Is it true that there exist always a fairly mixing mapping g and a non-decreasing mapping $h : \mathbb{R} \to \mathbb{R}$ such that $u = h \circ g$?

We ignore the answer to this problem.

References

1. Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. Memoryless determinacy of parity and mean payoff games: a simple proof. *Theor. Computer Science,* 310:365–378, 2004.
2. Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. Model checking discounted temporal properties. In *TACAS 2004,* volume 2988 of *LNCS,* pages 77–92. Springer, 2004.
3. Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Discounting the future in systems theory. In *ICALP 2003,* volume 2719 of *LNCS,* pages 1022–1037. Springer, 2003.
4. Andrzej Ehrenfeucht and Jerzy Mycielski. Positional strategies for mean payoff games. *Intern. J. of Game Theory,* 8:109–113, 1979.
5. E.A. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *FOCS'91,* pages 368–377. IEEE Computer Society Press, 1991.
6. Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes.* Springer, 1997.
7. Y. Gurevich and L. Harrington. Trees, automata and games. In *Proc. 17th Symp. on the Theory of Comp.,* pages 60–65. IEEE Computer Society Press, 1982.
8. Ashok P. Maitra and William D. Sudderth. *Discrete Gambling and Stochastic Games.* Springer.
9. A.W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, Instytut Matematyki, 1991.
10. Hervé Moulin. Prolongements des jeux á deux joueurs de somme nulle. Une théorie abstraite des duels. *Bull. Soc. math. France,* pages 1–111, 1976. Mémoire 45.
11. T.E.S Raghavan. Finite-step algorithms for single-controller and perfect information stochastic games. In A. Neyman and S. Sorin, editors, *Stohastic Games and Applications,* volume 570 of *NATO Science Series C, Mathematical and Physical Sciences,* pages 227–251. Kluwer Academic Publishers, 2003.
12. L. S. Shapley. Stochastic games. *Proceedings Nat. Acad. of Science USA,* 39:1095–1100, 1953.
13. Wiesław Zielonka. Perfect-information stochastic parity games. In *FOSSACS 2004,* volume 2987 of *LNCS,* pages 499–513. Springer, 2004.

The Dual of Concatenation

Alexander Okhotin

School of Computing, Queen's University, Kingston, Ontario, Canada K7L3N6
okhotin@cs.queensu.ca

Abstract. A binary language-theoretic operation is proposed, which is dual to the concatenation of languages in the same sense as the universal quantifier in logic is dual to the existential quantifier; the dual of Kleene star is defined accordingly. These operations arise whenever concatenation or star appears in the scope of negation. The basic properties of the new operations axe determined in the paper. Their use in regular expressions and in language equations is considered, and it is shown that they often eliminate the need of using negation, at the same time having an important technical advantage of being monotone. A generalization of context-free grammars featuring dual concatenation is introduced and proved to be equivalent to the recently studied Boolean grammars.

1 Introduction

The dual of a logical proposition $f(x_1, \ldots, x_n)$ is its transform under negation, $\neg f(\neg x_1, \ldots, \neg x_n)$. For instance, conjunction is the dual of disjunction, as stated by de Morgan law. The existential and the universal quantifier are dual to each other, since $\neg(\exists x)P(x)$ is equivalent to $(\forall x)\neg P(x)$. In temporal logic, the operators F ("eventually") and G ("always") demonstrate the same kind of duality. The possibility of saying "eventually X" instead of the cumbersome "not always not X" is certainly helpful both for intuitive clarity and due to the technical difficulties associated with handling nonmonotone negation.

Among the commonly used language-theoretic operations, union and intersection are dual to each other, while concatenation is apparently without a dual. As a result, reasoning about language constructs like $\overline{L_1 \cdot L_2}$ becomes as uncomfortable as it would be to deal with the statement "not always not X", being forbidden to use the notion "eventually". This makes a significant contribution to the general attitude to complement in formal language theory as a "hard" operation.

It is not hard to define the operation dual to concatenation: recalling that concatenation of two languages is

$$L_1 \cdot L_2 = \{w \mid \text{there exists a factorization } w = uv, \text{ s.t. } u \in L_1 \text{ and } v \in L_2\},$$

one can formally invert it by replacing the existential quantifier with the universal quantifier and conjunction with disjunction:

$$L_1 \odot L_2 = \{w \mid \text{for every factorization } w = uv \text{ it holds that } u \in L_1 \text{ or } v \in L_2\}.$$

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 698–710, 2004.
© Springer-Verlag Berlin Heidelberg 2004

By definition, this is just a binary operation on languages. Although it might look artificial at the first glance, one should bear in mind that such an operation invariably arises whenever concatenation occurs in the scope of negation. This gives a clear motivation for conducting a study of this operation, tentatively named *dual concatenation.*

In Sections 2 and 3, dual concatenation and its iterative counterpart, *dual star,* are formally introduced and their basic properties are established. Section 4 obtains closure/nonclosure results with respect to these operations for the most common families of languages. Sections 5 and 6 examine the use of dual concatenation together with or instead of concatenation in regular expressions and in language equations. A generalization of context-free grammars featuring explicit dual concatenation is defined in Section 7; it is shown to be computationally equivalent to the recently studied Boolean grammars [9], which feature explicit negation and are notable for being able to express the complete syntax of a simple programming language [11]. The final Section 8 summarizes the contributions of dual concatenation to these areas and argues for its importance.

2 The Dual Concatenation

Let us start from giving two equivalent definitions of the new operation.

Definition 1. *The dual concatenation of two languages* $L_1, L_2 \subseteq \Sigma^*$ *is*

$$L_1 \odot L_2 = \{w \mid \textit{for every factorization } w = uv \textit{ it holds that } u \in L_1 \textit{ or } v \in L_2\}$$

Definition 2. *The dual concatenation of two languages* $L_1, L_2 \subseteq \Sigma^*$ *is*

$$L_1 \odot L_2 = \overline{\overline{L_1} \cdot \overline{L_2}} \tag{1}$$

Theorem 1. *Definitions 1 and 2 are equivalent.*

Proof. Writing down a formal negation of Definition 1, a string w is *not* in the dual concatenation of L_1 and L_2 according to that definition if and only if there exists a factorization $w = uv$, such that $u \notin L_1$ and $v \notin L_2$. This is in turn equivalent to $w \in \overline{L_1} \cdot \overline{L_2}$, which holds iff w is *not* in the language (1). □

Theorem 2 (Algebraic properties of dual concatenation).

1. *Dual concatenation is associative, i.e.,* $L_1 \odot (L_2 \odot L_3) = (L_1 \odot L_2) \odot L_3$.
2. *Dual concatenation is not commutative, i.e.,* $L_1 \odot L_2 \neq L_2 \odot L_1$ *for some* L_1, L_2.
3. Σ^+ *is a two-sided identity for dual concatenation, i.e.,* $L \odot \Sigma^+ = \Sigma^+ \odot L = L$.
4. Σ^* *is a two-sided zero for dual concatenation, i.e.,* $L \odot \Sigma^* = \Sigma^* \odot L = \Sigma^*$.
5. *Dual concatenation is distributive over intersection, i.e.,* $L_1 \odot (L_2 \cap L_3) = (L_1 \odot L_2) \cap (L_1 \odot L_3)$.

Corollary 1. *Formal languages over an alphabet Σ form a semiring over $\cap$ as sum and $\odot$ as product, with zero Σ^* and identity Σ^+.*

Theorem 3 (Analytic properties of dual concatenation).

1. *Dual concatenation is monotone with respect to inclusion, i.e., if $L_1 \subseteq L_1'$ and $L_2 \subseteq L_2'$, then $L_1 \odot L_2 \subseteq L_1' \odot L_2'$.*
2. *Dual concatenation is $\cup$- and $\cap$-continuous, i.e., for every two sequences of languages $\{L_n'\}_{n=1}^{\infty}$ and $\{L_n''\}_{n=1}^{\infty}$ that are both increasing (both decreasing, resp.), the sequence $\{L_n' \odot L_n''\}_{n=1}^{\infty}$ is also increasing (decreasing, resp.) and its limit equals $\sup_{n\geqslant 1}\{L_n'\} \odot \sup_{n\geqslant 1}\{L_n''\}$ ($\inf_{n\geqslant 1}\{L_n'\} \odot \inf_{n\geqslant 1}\{L_n''\}$, resp.).*

An important case of concatenation is *linear concatenation*, where a singleton $\{a\}$ ($a \in \Sigma$) is left- or right- concatenated to a language: $a \cdot L$ and $L \cdot a$. What happens if the dual of concatenation is similarly restricted? Consider *linear dual concatenation* defined as $\overline{a} \odot L$ and $L \odot \overline{a}$. It turns out that linear concatenation and linear dual concatenation can be expressed through each other:

Theorem 4. *For every language L and for every $a \in \Sigma$, $\overline{a} \odot L = aL \cup \overline{a \cdot \Sigma^*}$ and $a \cdot L = \overline{a} \odot L \cap a \cdot \Sigma^*$.*

Example 1. Define the following three languages: $L_a = \{xay | x, y \in \{a,b\}^*,\ |x| = |y|\}$, $L_b = \{xby \mid x, y \in \{a,b\}^*,\ |x| = |y|\}$, $L_{even} = \{aa, ab, ba, bb\}^*$. Then $L = L_a \odot (L_b \cup L_{even}) \cap L_b \odot (L_a \cup L_{even}) = \{ww \mid w \in \{a,b\}^*\}$.

Let us informally explain how this construct ensures that $x \in L$ if and only if $x = ww$ for some w. Consider a string x of even length. For every factorization of x into two strings of odd length, $x = uv$, let $s, t \in \{a,b\}$ be the middle symbols in u and v respectively. It is easy to see that s and t occupy the same relative position in the two halves of x, and hence it should be checked that $s = t$. The requirement that $x \in L_a \odot (L_b \cup L_{even})$ states that, for every such factorization, $s = a$ or $t = b$, effectively forbidding the mismatch $s = b$ and $t = a$. Similarly, $x \in (L_a \cup L_{even}) \odot L_b$ forbids the case $s = a$ and $t = b$. The use of L_{even} allows to handle factorizations of x into even-length strings, as well as to enforce that $|x|$ is even.

3 The Dual Star

Kleene closure, $L^* = \bigcup_{n=0}^{\infty} L^n$, admits an equivalent representation as the set of all strings w, such that there exist $n \geqslant 0$ and a factorization $w = w_1 \dots w_n$, such that for all i $(1 \leqslant i \leqslant n)$ $w_i \in L$. Let us take a formal dual of this representation.

Definition 3. *Let $L \subseteq \Sigma^*$ be a language. The dual star of L is defined as*

$$L^{\circledast} = \{w \mid \textit{for every } n \geqslant 0 \textit{ and for every factorization } w = w_1 \dots w_n \textit{ there exists } i\ (1 \leqslant i \leqslant n), \textit{ such that } w_i \in L\} \tag{2}$$

Definition 4. *Let $L \subseteq \Sigma^*$ be a language. Define the dual star of L as $L^{⊛} = \overline{\overline{L}^*}$.*

Consider the third way of defining the dual star, by an iterative application of dual concatenation:

Definition 5. *The n-th dual concatenation power of a language L is defined as $L^{ⓝ} = L \odot \ldots \odot L$, where L is repeated n times; if $n = 0$, assume $L^{ⓝ} = \Sigma^+$.*

Definition 6. *The n-th dual concatenation power of L is defined as $L^{ⓝ} = \{w |$ for every factorization $w = w_1 \ldots w_n$ there exists i $(1 \leqslant i \leqslant n)$, s.t. $w_i \in L\}$.*

Theorem 5. *Definitions 5 and 6 are equivalent.*

Definition 7. *Define the dual star of $L \subseteq \Sigma^*$ as $L^{⊛} = \bigcap_{n=0}^{\infty} L^{ⓝ}$.*

Theorem 6. *Definitions 3, 4 and 7 are equivalent.*

Theorem 7. *For every language L, (1) $\varepsilon \notin L^{⊛}$ (cf.: $\varepsilon \in L^*$); (2) $L^{⊛} \subseteq L \subseteq L^*$; (3) $X = L^{⊛}$ is the greatest solution of equation $X = X \odot L \cap \Sigma^+$.*

Example 2. Consider the languages L_a, L_b, L_{even} from Example 1 and define $L_1 = c \cdot L_a \cup L_b \cdot d \cup L_{even} d \cup c \cdot L_{even} \cdot d \cup \{a, b\}^*$ and $L_2 = c \cdot L_b \cup L_a \cdot d \cup L_{even} d \cup c \cdot L_{even} \cdot d \cup \{a, b\}^*$. Then $L_1^{⊛} \cap L_2^{⊛} \cap c \cdot L_{even} \cdot d = \{cwwd \mid w \in \{a, b\}^*\}$.

4 Closure Properties

Let $\mathcal{L} \subseteq 2^{\Sigma^*}$ be a class of languages. Denote co-$\mathcal{L} = \{\overline{L} \mid L \in \mathcal{L}\}$.

Theorem 8. *A class of languages $\mathcal{L} \subseteq 2^{\Sigma^*}$ is closed under dual concatenation (dual star) if and only if co-$\mathcal{L}$ is closed under concatenation (star, resp.).*

Proof. Let $\mathcal{L}$ be closed under concatenation, and consider an arbitrary pair of languages $L_1, L_2 \in$ co-$\mathcal{L}$. The languages $\overline{L_1}$ and $\overline{L_2}$ are in $\mathcal{L}$ according to the definition of co-$\mathcal{L}$, and then, by the closure of $\mathcal{L}$ under concatenation, $\overline{L_1} \cdot \overline{L_2}$ is in $\mathcal{L}$. Hence, its complement, $\overline{\overline{L_1} \cdot \overline{L_2}} = L_1 \odot L_2$, is in co-$\mathcal{L}$. The case of dual star is similar. □

Corollary 2. *Every family of languages closed under complement is (i) either closed or not closed under both concatenation and dual concatenation; (ii) either closed or not closed under both star and dual star.*

Theorem 9. *Regular, deterministic context-sensitive, context-sensitive, recursive, and the languages generated by Boolean grammars [9] are closed under both dual concatenation and dual star. Deterministic context-free and linear conjunctive languages [8] are closed under neither dual concatenation nor dual star.*

Lemma 1. *Consider the languages L_a, L_b, L_{even} from Example 1. The language $L_a \odot (L_b \cup L_{even})$ is not a finite intersection of context-free languages ($\cap CF$). Similarly, the languages $L_1^{\circledast}$ and $L_2^{\circledast}$ from Example 2 are not in $\cap CF$.*

Proof. Suppose $L_a \odot (L_b \cup L_{even})$ is in $\cap CF$. By the symmetry, so is $L_b \odot (L_a \cup L_{even})$. Hence, their intersection is also in $\cap CF$. However, this intersection equals $\{ww \mid w \in \{a,b\}^*\}$, which is known not to be in $\cap CF$ [14], a contradiction.

If $L_1^{\circledast}$ from Example 2 (and, by the symmetry, $L_2^{\circledast}$) can be represented by such an intersection, then $L_1^{\circledast} \cap L_2^{\circledast} \cap c \cdot L_{even} \cdot d = \{cwwd \mid w \in \{a,b\}^*\}$ is in $\cap CF$, which similarly yields a contradiction. □

Since L_a, $L_b \cup L_{even}$ and L_1 are linear context-free, Lemma 1, partially with the help of Theorem 8, leads to the following nonclosure results:

Theorem 10. *Context-free and linear context-free languages are not closed under dual concatenation and dual star. Co-context-free and co-linear-context-free languages are not closed under concatenation and star.*

Theorem 11. *Finite languages are closed under both dual concatenation and dual star.*

Proof. **The case of $\odot$.** Let L_1 and L_2 be arbitrary finite languages. Let m_i be the length of the longest string in L_i (take $m_i = 0$ if $L_i = \varnothing$) and let us prove that $L_1 \odot L_2$ does not contain strings of more than $m_1 + m_2 + 1$ symbols long.

Suppose there is $w \in L_1 \odot L_2$, such that $|w| \geqslant m_1 + m_2 + 2$. By the definition of dual concatenation, for every factorization $w = uv$ it should hold that $u \in L_1$ or $v \in L_2$. Consider the factorization $w = uv$, such that $|u| = m_1 + 1$ and thus $|v| \geqslant m_2 + 1$. By the choice of m_i, $u \notin L_1$ and $v \notin L_2$, which is a contradiction.

So, $L_1 \odot L_2 \subseteq \Sigma^{\leqslant m_1 + m_2 + 1}$ and is therefore finite.

The case of $\circledast$. By Theorem 7, $L^{\circledast}$ is a subset of L, and hence is finite. □

Finally, the following theorem can easily be proved by an explicit construction of a Turing machine. The results of this section are put together in Table 1.

Theorem 12. *Recursively enumerable and co-recursively enumerable languages are closed under both dual concatenation and dual star.*

Table 1. Closure properties of common families of languages

	$\sim$	$\cup$	$\cap$	$\cdot$	$\odot$	$*$	$\circledast$
Finite	−	+	+	+	+	−	+
co-Finite	−	+	+	+	+	+	−
Regular	+	+		+		+	
LinCF	−	+	−	−	−	−	−
co-LinCF	−	−	+	−	−	−	−
DetCF	+	−		−		−	
CF	−	+	−	+	−	+	−
co-CF	−	−	+	−	+	−	+
LinConj [5,8]	+	+		−		−	
Conjunctive [5]	?	+	+	+	?	+	?
co-Conjunctive	?	+	+	?	+	?	+
Boolean [9]	+	+		+		+	
DetCS	+	+		+		+	
CS	+	+		+		+	
Recursive	+	+		+		+	
RE	−	+	+	+	+	+	+
co-RE	−	+	+	+	+	+	+

5 Dual Concatenation in Regular Expressions

A fundamental theorem due to Kleene states that a set is recognized by a finite automaton if and only if it can be represented as a *regular expression,* which is a formula over the operations $\cup$, $\cdot$ and $*$ and the constants a, ε and $\varnothing$. More expressive formalisms, *semi-extended regular expressions* with the operations $\{\cup, \cap, \cdot, *\}$, and *extended regular expressions* with $\cup$, $\cap$, $\sim$, $\cdot$ and $*$ [2,15], have been subsequently studied. Consider the following related formalism:

Definition 8 (Extended dual regular expression). *(i)* $\varnothing$, $\overline{\varnothing}$, ε, $\overline{\varepsilon}$, a *and* $\overline{a}$ *(for every* $a \in \Sigma$*) are extended dual regular expressions. (ii) If* α *is an extended dual regular expression, then so are* α^* *and* $\alpha^{\circledast}$ *(iii) If* α, β *are extended dual regular expressions, then so are* $(\alpha \cdot \beta)$, $(\alpha \odot \beta)$, $(\alpha \mid \beta)$, $(\alpha \& \beta)$. *A language* $L(\alpha)$ *is associated with every extended dual regular expression* α *in an obvious way:* $\mid$ *means union,* & *denotes intersection, other symbols specify themselves.*

As compared with extended regular expressions [15], extended dual regular expressions do not have negation, but have dual concatenation and dual star instead. It turns out that the lack of negation does not increase the descriptional complexity, as any given extended dual regular expression can be negated by simply changing all operations to their duals:

Definition 9. The dual $d(\alpha)$ *of an extended dual regular expression* α *is defined as* $d(\varnothing) = \overline{\varnothing}$, $d(\overline{\varnothing}) = \varnothing$, $d(\varepsilon) = \overline{\varepsilon}$, $d(\overline{\varepsilon}) = \varepsilon$, $d(a) = \overline{a}$, $d(\overline{a}) = a$, $d(\alpha^*) = d(\alpha)^{\circledast}$, $d(\alpha^{\circledast}) = d(\alpha)^*$, $d(\alpha \cdot \beta) = d(\alpha) \odot d(\beta)$, $d(\alpha \odot \beta) = d(\alpha) \cdot d(\beta)$, $d(\alpha \mid \beta) = d(\alpha) \& d(\beta)$, $d(\alpha \& \beta) = d(\alpha) \mid d(\beta)$.

Lemma 2. *For every α as in Definition 8, $d(d(\alpha)) = \alpha$ and $L(d(\alpha)) = \overline{L(\alpha)}$.*

So the succinctness of description of languages by extended regular expressions and by extended dual regular expressions is the same. Along with this, the latter inherit many other amazing properties of the former, such as the nonelementary complexity of the emptiness problem [2], and nonelementary succinctness tradeoff with the standard regular expressions [2]. It is important to emphasize that in our case this is being achieved without using the complement: the operations $\cdot$, $\odot$, $*$, $\circledast$, $\cup$ and $\cap$ are all monotone.

Let us now consider some restricted classes of regular expressions with dual concatenation: *semi-extended dual regular expressions* forbid the operations of $\cdot$ and $*$ and the constants $\varnothing$, ε and a (retaining $\odot$, $\circledast$, $\cup$, $\cap$ and the constants $\overline{\varnothing}$, $\overline{\varepsilon}$, $\overline{a}$), while *dual regular expressions* also exclude union (thus featuring $\odot$, $\circledast$, $\cap$ and the same constants).

Proposition 1. *The dual of every regular expression is a dual regular expression. The dual of a semi-extended regular expression is a semi-extended dual regular expression.*

Generally, dual regular expressions have the same expressive power as the standard regular expressions: for every regular L, $\overline{L}$ is regular and hence is generated by some regular expression α – consequently, the dual regular expression $d(\alpha)$ generates L. Thus dual regular expressions, semi-extended dual regular expressions and extended dual regular expressions form three new equivalent representations of regular sets. They vary in descriptional complexity:

Example 3. It is known that the language $L_n = (0|1)^* \cdot 0 \cdot (0|1)^n \cdot 0 \cdot (0|1)^*$ is specified by a regular expression of size $O(n)$, while $\overline{L_n}$ requires a regular expression of size at least $C \cdot 2^n$ [2]. On the other hand, $\overline{L_n}$ is specified by the dual regular expression $(\overline{0}\&\overline{1})^{\circledast} \odot \overline{0} \odot (\overline{0}\&\overline{1})^{\circledast} \odot \overline{0} \odot (\overline{0}\&\overline{1})^{\circledast}$ of linear size, but L_n requires an exponential-size dual regular expression.

6 Dual Concatenation in Language Equations

Language equations with concatenation and union were studied in the early days of formal language theory as an algebraic semantics for context-free grammars [1,3]; their extension, additionally equipped with intersection, has been shown to characterize *conjunctive grammars* [5,6] in a similar way.

Complement proved to be a problematic operation for language equations: a system of equations with $\cdot$, $\cup$, $\cap$ and $\sim$ can have no solutions at all or multiple pairwise incomparable solutions, these properties are undecidable, and the expressive power of unique solutions amounts to all recursive languages [7]. Using such equations typically forces one to impose heavy syntactical [4] or semantical [9] constraints. In this section, language equations with dual concatenation instead of complement, i.e., with the operations $\cdot$, $\odot$, $\cup$ and $\cap$, will be studied.

Definition 10. *Let φ_i $(1 \leqslant i \leqslant n)$ be expressions that contain symbols from Σ and variables $(X_1, \ldots, X_n)$ that assume values of languages over Σ -connected with language-theoretic operations from a certain fixed set (e.g., $\{\cdot, \odot, \cup, \cap\}$). Then $X_i = \varphi_i(X_1, \ldots, X_n)$ $(X = \varphi(X)$ is vector form) is called a resolved system of language equations. $L = (L_1, \ldots, L_n)$ is its solution if $L_i = \varphi_i(L)$ for all i. L is called a solution modulo a language $M \subseteq \Sigma^*$, if $L_i \cap M = \varphi_i(L) \cap M$ for all i. A system is said to have a strongly unique solution, if it has a unique solution modulo every finite language closed under substring.*

In light of Theorem 3, it is easy to generalize the well-known properties of language equations over $\{\cdot, \cup\}$ [1,3] for the case of language equations over $\{\cdot, \odot, \cup, \cap\}$:

Theorem 13. *Every system of language equations over $\{\cdot, \odot, \cup, \cap\}$ has least and greatest solutions given by the limits of the increasing sequence $\{\varphi^i(\varnothing, \ldots, \varnothing)\}_{i=0}^{\infty}$ and of the decreasing sequence $\{\varphi^i(\Sigma^*, \ldots, \Sigma^*)\}_{i=0}^{\infty}$ respectively.*

Theorem 14 (Criterion of solution uniqueness for monotone systems). *A system over $\{\cdot, \odot, \cup, \cap\}$ has a unique solution if and only if it has a strongly unique solution.*

Theorem 14 expresses the uniqueness of solution by a first-order formula with a single universal quantifier; this shows that the problem is co-recursively-enumerable. It can be proved to be hard for this class by reducing the context-free universality problem.

Theorem 15. *The set of systems over $\{\cdot, \odot, \cup, \cap\}$ that have a unique solution is co-RE-complete.*

Theorem 16. *For every system of language equations over $\{\cdot, \odot, \cup, \cap\}$ there exists and can be effectively constructed a system of language equations over the same set of operations, which has a unique solution, such that the first component of this unique solution coincides with the first component of the original system's least (greatest) solution.*

Therefore, least, greatest, unique and strongly unique solutions of language equations over $\{\cdot, \odot, \cup, \cap\}$ denote a common class of languages. Let us show that this is the class defined by strongly unique solutions of language equations over $\{\cdot, \cup, \cap, \sim\}$. First, define the dual $d(\varphi)$ of a formula φ over $\{\cdot, \odot, \cup, \cap\}$ in variables $(X_1, \ldots, X_n)$ similarly to the dual of a regular expression (Definition 9); the only thing to add is that $d(X_i) = X_i$. Then:

Lemma 3. *For every formula φ, $d(d(\varphi)) = \varphi$, and for every vector of languages $(L_1, \ldots, L_n)$, $d(\varphi)(L_1, \ldots, L_n) = \overline{\varphi(\overline{L_1}, \ldots, \overline{L_n})}$ (cf. Lemma 2).*

Corollary 3. *$(L_1, \ldots, L_n)$ is a solution of $X_i = \varphi_i(X)$ $(i = 1 \ldots n)$ if and only if $(\overline{L_1}, \ldots, \overline{L_n})$ is a solution of $X_i = d(\varphi_i)(X)$ $(i = 1 \ldots n)$.*

Table 2. Negation vs. dual concatenation in language equations

	$\cup, \cap, \cdot, \sim$	$\cup, \cap, \cdot, \odot$
Decision problems		
Existence of solution	co-RE-complete [7]	trivial (Th. 13)
Uniqueness of solution	Π_2-complete [7]	co-RE-complete (Th. 15)
Strong uniqueness of solution	co-RE-complete [9]	co-RE-complete (Th. 15)
Existence of minimal solution	Π_2-complete [7]	trivial (Th. 13)
Existence of maximal solution	Π_2-complete [7]	trivial (Th. 13)
Expressive power		
Unique solutions	Recursive [7]	Boolean (Th. 18)
Least solutions	RE [7]	Boolean (Th. 18)
Greatest solutions	co-RE [7]	Boolean (Th. 18)
Strongly unique solutions	Boolean [9]	Boolean (Th. 18)

Corollary 4. *If a system* $X_i = \varphi_i(X)$ $(i = 1 \ldots n)$ *has a unique (a least, a greatest) solution* $(L_1, \ldots, L_n)$, *then the system* $X_i = d(\varphi_i)(X)$ $(i = 1 \ldots n)$ *has the unique (the greatest, the least, resp.) solution* $(\overline{L_1}, \ldots, \overline{L_n})$.

Theorem 17. *For every system of language equations over* $\{\cdot, \cup, \cap, \sim\}$, *which has a strongly unique solution with the first component L, there exists and can be effectively constructed a system of language equations over* $\{\cdot, \odot, \cup, \cap\}$, *which has a unique solution with the same first component.*

Theorem 18. *The set of languages specified by unique (least, greatest) solutions of system of language equations over* $\{\cdot, \odot, \cup, \cap\}$ *coincides with the set of languages specified by strongly unique solutions of systems over* $\{\cdot, \cup, \cap, \sim\}$.

Strongly unique solutions of systems over $\{\cdot, \cup, \cap, \sim\}$ were originally used to define *Boolean grammars* [9], which are context-free grammars with added conjunction and negation. The same class of languages has now been obtained using only monotone operations. The differences between these two representations are summarized in Table 2. Although negation is intuitively more clear than dual concatenation, the formal properties of language equations with dual concatenation are definitely more attractive.

Let us now determine the expressive power of restricted types of language equations, which may contain dual concatenation but not concatenation. First, using Corollary 4, a statement akin to Proposition 1 can be obtained.

Proposition 2. *If unique (least, greatest) solutions of language equations over some fixed set of operations denote a language family* $\mathcal{L}$, *then unique (greatest, least, resp.) solutions of language equations over the set of duals of these operations denote co-*$\mathcal{L}$.

Theorem 19. *The languages denoted by least, unique and greatest solutions of language equations with* $\cap$ *and* LIN$\odot$ *are the* ***co-linear-context-free*** *languages; language equations with* $\cap$ *and* $\odot$ *denote* ***co-context-free*** *languages; language equations with* $\cup$, $\cap$ *and* LIN$\odot$ *denote* ***linear conjunctive*** *languages [5,8]; language equations with* $\cup$, $\cap$ *and* $\odot$ *denote* ***co-conjunctive*** *languages [5].*

The known results on the expressive power of language equations are presented in Table 3; those established in this paper are labeled with an asterisk.

7 Dual Concatenation in Formal Grammars

The principle of *parsing as deduction* [12], brought to a formal perfection in the theory of *parsing schemata* [13], means representing a parsing method as a formal deduction system. For instance, the Cocke–Kasami–Younger algorithm would be formally described using elementary propositions of the form $[A, w]$ $(A \in N,\ w \in \Sigma^+)$ and employing axioms like $\vdash [A, a]$ $(A \to a \in P)$ and inference rules like $[B, u], [C, v] \vdash [A, uv]$ $(A \to BC \in P)$. This approach can also serve as an alternative definition of context-free grammars. Let us use it as the primary definition of grammars with explicit dual concatenation.

Definition 11. *A dual concatenation grammar is a quadruple* $G = (\Sigma, N, P, S)$, *in which* Σ *and* N *are finite nonempty disjoint sets of terminal and nonterminal systems,* $S \in N$ *and every production in* P *is of the form:*

$$A \to s_{11} \cdot \ldots \cdot s_{1k_1} \,\&\, \ldots \,\&\, s_{m1} \cdot \ldots \cdot s_{mk_m} \,\&\, t_{11} \odot \ldots \odot t_{1l_1} \,\&\, \ldots \,\&\, t_{n1} \odot \ldots \odot t_{nl_n}, \quad (3)$$

where $m + n \geqslant 1$, $k_i, l_i \geqslant 0$, $s_{ij} \in \Sigma \cup N$ *and* $t_{ij} \in \{\overline{a} \mid a \in \Sigma\} \cup N$.

The language generated by a grammar is defined by a formal deduction system:

Table 3. Classes of languages defined by solutions of language equations

Operations	Strongly unique	Unique	Least	Greatest
$\cup$, LIN$\cdot$ $\cap$, LIN$\odot$	LinCF [9] co-LinCF*	LinCF [1,3] co-LinCF*	LinCF [1,3] co-LinCF*	LinCF [10] co-LinCF*
$\cup$, $\cdot$ $\cap$, $\odot$	CF [9] co-CF*	CF [1,3] co-CF*	CF [1,3] co-CF*	CF [10] co-CF*
$\cup$, $\cap$, LIN$\cdot$ $\cup$, $\cap$, LIN$\odot$	Lin.Conj. [9] Lin.Conj.*	Lin.Conj. [10] Lin.Conj.*	Lin.Conj. [6] Lin.Conj.*	Lin.Conj. [10] Lin.Conj.*
$\cup$, $\cap$, $\cdot$ $\cup$, $\cap$, $\odot$	Conj. [9] co-Conj.*	Conj. [10] co-Conj.*	Conj. [6] co-Conj.*	Conj. [10] co-Conj.*
$\cup$, $\cap$, $\cdot$, $\odot$	Bool.*	Bool.*	Bool.*	Bool.*
$\cup$, $\cap$, $\sim$, LIN$\cdot$	Lin.Conj. [9]	Rec. [7]	RE [7]	co-RE [7]
$\cup$, $\cap$, $\sim$, $\cdot$	Bool. [9]	Rec. [7]	RE [7]	co-RE [7]

Definition 12. *Let* $\{[\varphi, w] \mid \varphi$ *is* $s_1 \cdot \ldots \cdot s_k$ *or* $s_1 \odot \ldots \odot s_k$ *for some* $s_i \in \Sigma \cup N$, $w \in \Sigma^*\}$ *be the set of elementary propositions (items). The axioms are:* $\boldsymbol{A1}$*:* $\vdash [\varepsilon, \varepsilon]$, $\boldsymbol{A2}$*:* $\vdash [a, a]$ *(for all* $a \in \Sigma$*),* $\boldsymbol{A3}$*:* $\vdash [\overline{\varepsilon}, w]$ *(for all* $w \in \Sigma^+$*)* $\boldsymbol{A4}$*:* $\vdash [\overline{a}, w]$ *(for all* $a \in \Sigma$ *and* $w \in \Sigma^* \setminus \{a\}$*). The deduction rules are:* $\boldsymbol{C}$*:* $[\varphi, u], [\psi, v] \vdash [\varphi \cdot \psi, uv]$ *for all* φ, ψ *and* $u, v \in \Sigma^*$. $\boldsymbol{D}$*:* $[\varphi_{i_0}, u_0], \ldots, [\varphi_{i_{|w|}}, u_{|w|}] \vdash [\varphi_1 \odot \varphi_2, w]$ *for all* φ_1, φ_2, $w \in \Sigma^*$, $i_j \in \{1, 2\}$ *and* $u_j \in \Sigma^*$ $(0 \leqslant i_j \leqslant |w|)$*, such that each* u_j *is a* j*-symbol prefix of* w *if* $i_j = 1$*, or a* $(|w| - j)$*-symbol suffix of* w *if* $i_j = 2$. $\boldsymbol{P}$*:* $[s_{11} \cdot \ldots \cdot s_{1k_1}, w], \ldots, [s_{m1} \cdot \ldots \cdot s_{mk_m}, w], [t_{11} \odot \ldots \odot t_{1l_1}, w], \ldots, [t_{n1} \odot \ldots \odot t_{nl_n}, w] \vdash [A, w]$ *for every rule (3).*

Define $L_G(\varphi) = \{w \mid G \vdash [\varphi, w]\}$ *and* $L(G) = L_G(S) = \{w \mid G \vdash [S, w]\}$.

Example 4. Consider the following dual concatenation grammar, which generates $\{ww \mid w \in \{a, b\}^*\}$ as in Example 1. $G \vdash [S, abab]$ is proved below.

$$S \to A \odot D \,\&\, B \odot C \qquad C \to A \mid E$$
$$A \to a \cdot A \cdot a \mid a \cdot A \cdot b \mid b \cdot A \cdot a \mid b \cdot A \cdot b \mid a \qquad D \to B \mid E$$
$$B \to a \cdot B \cdot a \mid a \cdot B \cdot b \mid b \cdot B \cdot a \mid b \cdot B \cdot b \mid b \qquad E \to a \cdot a \cdot E \mid a \cdot b \cdot E \mid b \cdot a \cdot E \mid b \cdot b \cdot E \mid \varepsilon$$

$\vdash [a, a]$	$[a, a], [b \cdot E, b] \vdash [a \cdot b \cdot E, ab]$
$[a, a] \vdash [A, a]$	$[a \cdot b \cdot E, ab] \vdash [E, ab]$
$\vdash [b, b]$	$[b, b], [E, ab] \vdash [b \cdot E, bab]$
$[b, b] \vdash [B, b]$	$[a, a], [b \cdot E, bab] \vdash [a \cdot b \cdot E, abab]$
$[a, a], [B, b] \vdash [a \cdot B, ab]$	$[a \cdot b \cdot E, abab] \vdash [E, abab]$
$[a \cdot B, ab], [a, a] \vdash [a \cdot B \cdot a, aba]$	$[E, \varepsilon] \vdash [D, \varepsilon]$
$[a \cdot B \cdot a, aba] \vdash [B, aba]$	$[E, ab] \vdash [D, ab]$
$[B, aba] \vdash [D, aba]$	$[E, abab] \vdash [D, abab]$
$\vdash [\varepsilon, \varepsilon]$	$[D, abab], [A, a], [D, ab], [D, aba], [D, \varepsilon] \vdash [A \odot D, abab]$
$[\varepsilon, \varepsilon] \vdash [E, \varepsilon]$	In the same way one deduces: $\vdash [B \odot C, abab]$
$[b, b], [E, \varepsilon] \vdash [b \cdot E, b]$	$[A \odot D, abab], [B \odot C, abab] \vdash [S, abab]$

For dual concatenation grammars there exists a direct analog of the classical characterization of context-free grammars due to Ginsburg and Rice [3]:

Definition 13. *Let* $G = (\Sigma, N, P, S)$ *be a dual concatenation grammar. The system of language equations corresponding to* G *is a system over* Σ *in variables* N*, in which the equation for every* $A \in N$ *is*

$$A = \bigcup_{\substack{\text{There is a rule (3)} \\ \text{for } A \text{ in } P}} \left(\bigcap_{i=1}^{m} (s_{i1} \cdot \ldots \cdot s_{ik_i}) \cap \bigcap_{i=1}^{n} (t_{i1} \odot \ldots \odot s_{il_i}) \right) \tag{4}$$

Now it can be proved that if $(L_1, \ldots, L_n)$ is the least solution of the system corresponding to a dual concatenation grammar G, then for every $A_i \in N$ and $w \in \Sigma^*$, $G \vdash [A, w]$ if and only if $w \in L_i$. This yields the following result:

Theorem 20. *For every grammar* $G = (\Sigma, \{A_1, \ldots, A_n\}, P, S)$*, the least solution of the corresponding system of equations is* $(L_G(A_1), \ldots, L_G(A_n))$.

This result establishes a correspondence between the classes of grammars and language equations. If $n = 0$ for every rule (3) of some grammar, the grammar is *conjunctive;* if additionally $m = 1$, the grammar is *context-free.* If $n = 0$ and there is no more than one nonterminal symbol in every conjunct $s_{i1} \cdot \ldots \cdot s_{ik_i}$, the grammar is *linear conjunctive;* if additionally $m = 1$, the grammar is *linear context-free.* On the other hand, if $m = 0$ for every rule, the grammar can be named *co-conjunctive;* if at the same time there is exactly one rule for every nonterminal, the resulting object, being the first "positive" grammatical characterization of the co-context-free languages, deserves the name of a *co-context-free* grammar. If $m = 0$ and there is no more than one nonterminal symbol in every conjunct $t_{i1} \odot \ldots \odot t_{il_i}$, the grammar is *co-linear-conjunctive* (such grammars will generate exactly the linear conjunctive languages); if additionally the rule for each nonterminal is unique, *co-linear-context-free* grammars are obtained.

8 Conclusion

Joint use of concatenation and complement yields dual concatenation whether we want it or not. This paper attempted to consider this operation as a self-contained entity and to use it in some contexts where both concatenation and complement are (at least *would be*) naturally used: in regular expressions, in language equations and in formal grammars.

When concatenation and dual concatenation are used *together,* the use of the complement is considerably facilitated. One can reduce the scope of negation to minimal terms, and in many cases get rid of it entirely. The latter was the case with extended dual regular expressions. As for language equations, it turned out that the equations with $\{\cdot, \odot, \cup, \cap\}$ denote the same class of languages as a specifically restricted class of equations over $\{\cdot, \cup, \cap, \sim\}$ studied under the name of Boolean grammars [9] and capable of specifying the set of well-formed programs in a simple programming language [11]. This results also allowed to characterize Boolean grammars by a simple formal deductive system, contributing to the study of this noteworthy language family.

When dual concatenation is used *instead* of the standard concatenation, the resulting constructs tend to specify complements of what could be originally denoted. Dual regular expressions form a simple example. The proposed co-context-free grammars give the first "positive", negation-free specification for this long known family of languages.

Generally, it can be concluded that dual concatenation often allows to avoid the use of logical negation while reasoning about concatenation of languages, or at least facilitate such reasoning. Hence, whenever concatenation and complement of languages are being used together, it makes sense to recognize the dual concatenation and take benefit of its properties.

References

1. N. Chomsky, M. P. Schützenberger, "The algebraic theory of context-free languages", in: Braffort, Hirschberg (Eds.), *Computer Programming and Formal Systems,* North-Holland Publishing Company, Amsterdam, 1963, 118–161.
2. K. Ellul, J. Shallit, M.-w. Wang, "Regular expressions: new results and open problems", *Pre-proc. of DCFS 2002* (London, Canada, 21–24 August, 2002), 17–34.
3. S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM,* 9 (1962), 350–371.
4. E. L. Leiss, "Unrestricted complementation in language equations over a one-letter alphabet", *Theoretical Computer Science,* 132 (1994), 71–93.
5. A. Okhotin, "Conjunctive grammars", *Journal of Automata, Languages and Combinatorics,* 6:4 (2001), 519–535.
6. A. Okhotin, "Conjunctive grammars and systems of language equations", *Programming and Computer Software,* 28:5 (2002) 243–249.
7. A. Okhotin, "Decision problems for language equations with Boolean operations", *Automata, Languages and Programming* (ICALP 2003, Eindhoven, The Netherlands, June 30–July 4, 2003), LNCS 2719, 239–251; journal version submitted.
8. A. Okhotin, "On the equivalence of linear conjunctive grammars to trellis automata", *Informatique Théorique et Applications,* 38:1 (2004), 69–88.
9. A. Okhotin, "Boolean grammars", *Information and Computation,* to appear; earlier version in: *Developments in Language Theory* (DLT 2003), LNCS 2710, 398–410.
10. A. Okhotin, "Greatest solutions of language equations", submitted.
11. A. Okhotin, "A Boolean grammar for a simple programming language", Tech. Rep. 2004–478, School of Computing, Queen's University, Kingston, Ontario, Canada.
12. S. M. Shieber, Y. Schabes, F. C. N. Pereira, "Principles and implementation of deductive parsing", *Journal of Logic Programming,* 24 (1995), 3–36.
13. K. Sikkel, *Parsing Schemata,* Springer-Verlag, 1997.
14. D. Wotschke, "The Boolean closures of deterministic and nondeterministic context-free languages", In: W. Brauer (ed.), *Gesellschaft für Informatik e. V., 3. Jahrestagung 1973,* LNCS 1, Springer-Verlag, Berlin, 1973, 113–121.
15. S. Yu, "Regular Languages", in: Rozenberg, Salomaa (Eds.), *Handbook of Formal Languages,* Vol. 1, 41–110, Springer-Verlag, Berlin, 1997.

Computational Aspects of Disjunctive Sequences

Klaus Ambos-Spies and Edgar Busse

Institut für Informatik, Ruprecht-Karls-Universität Heidelberg,
Im Neuenheimer Feld 294, D-69120 Heidelberg, Germany

Abstract. An infinite binary sequence is disjunctive if every binary word occurs as a subword in the sequence. For a computational analysis of disjunctive sequences we can identify an infinite 0-1-sequence either with its prefix set or with its corresponding set, where a set A of binary words corresponds to a sequence α if α is the characteristic sequence of A. Most of the previous investigations of disjunctive sequences have dealt with prefix sets. Here, following the more common point of view in computability theory, we focus our investigations on the sets corresponding to disjunctive sequences.

We analyze the computational complexity and the Chomsky complexity of sets corresponding to disjunctive sequences. In particular, we show that no such set is regular but that there are linear languages with disjunctive characteristic sequences. Moreover, we discuss decidability questions. Here our main results are that the disjunctivity problem for Chomsky-0-grammars is Π_3^0-complete while the corresponding problem for linear or context free or context sensitive grammars is Π_2^0-complete. The latter implies that, for any language class C which contains the linear languages, the class of the languages in C corresponding to disjunctive sequences is not recursively presentable.

1 Introduction

An infinite binary sequence is disjunctive if every binary word occurs as a subword in the sequence. We obtain simple examples of disjunctive sequences by concatenating all nonempty binary words in length lexicographical ordering (0 1 00 01 10 11 000 ...) or by concatenating all binary numbers (0 1 10 11 100 ...), the latter sequence being known as Champernowne's sequence (for base 2). A special case of disjunctive sequences are the Borel normal sequences in which all words of a given length occur with the same frequency. Disjunctive sequences are abundant. Typical sequences in the sense of Baire category or Lebesgue measure are disjunctive. To be more precise, the class of disjunctive sequences is co-meager and has measure 1 (see Staiger [9], [11], [12]). Complementing the simple examples of disjunctive sequences above, this abundance implies that there are arbitrarily complex disjunctive sequences (see e.g. Calude and Yu [4]).

For a computational analysis of disjunctive sequences it is convenient to appropriately identify sequences with sets or languages (i.e., sets of natural numbers or binary words which we will identify in the following). This will allow us to use

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 711–722, 2004.
© Springer-Verlag Berlin Heidelberg 2004

the standard concepts of computability theory, computational complexity theory and the theory of formal languages which in general are phrased in terms of sets or languages. Most commonly this is done by either identifying a sequence α with the set $Prefix(\alpha)$ of its finite initial segments or by identifying α with its corresponding set $S(\alpha)$ which has the property that the nth word (with respect to the length-lexicographical ordering) is an element of $S(\alpha)$ if and only if the nth bit of the sequence α is 1 (in other words, the sequence α is the characteristic sequence of $S(\alpha)$).

While the previous investigations of computational aspects of disjunctive sequences focussed on their prefix sets, here we will deal with the sets corresponding to disjunctive sequences. The identification of a sequence with its corresponding set has the advantage that it yields a one-to-one correspondence between sequences and sets, whereas prefix sets have a very particular syntactic structure (e.g. contain just one word of each length). As a consequence, some complexity classes do not contain any prefix sets whence the computational analysis of disjunctive sequences via their prefix sets might be coarser than that via corresponding sets.

By analyzing the time complexity of disjunctive sequences we obtain a first example of this phenomenon (see Section 3 below): While we can easily show that any linear-time many-one degree contains a set corresponding to a disjunctive sequence (i.e., intuitively, for any set A we can find a set with disjunctive characteristic sequence which has the same complexity as A), there are such degrees (in fact polynomial-time Turing degrees) which do not contain any prefix set at all.

This deficiency of the prefix set approach becomes still more apparent if we locate the disjunctive sequences in the Chomsky hierarchy. Intuitively, it is clear that a finite automaton cannot recognize a disjunctive sequence but, in view of the simple examples of disjunctive sequences above, one might expect that there is a context free disjunctive sequence. As Calude and Yu [4] have shown, however, there is no disjunctive sequence with context free prefix set. Their proof is based on the observation that, for prefix sets, the lower part of the Chomsky hierarchy collapses, namely any context free prefix set is already regular. (See Section 4 below for a presentation of these and some new related results.)

For corresponding sets we contrast this result by showing that sets corresponding to disjunctive sequences are not regular but that there is a disjunctive sequence with context free - in fact linear - corresponding language. We have obtained these results already implicitly in [1] where we have shown that disjunctive sequences may be viewed as sequences which appear generic to a finite automaton. The proofs we present here, however, are more direct and somewhat simpler (see Section 5).

We then turn to decidability and presentability questions. For powerful concepts like context sensitive grammars or even more general concepts like general Chomsky grammars the undecidability of the disjunctivity problem, in fact Π_2^0-hardness of the problem, can be easily derived from some general results. For context free or even linear grammars, however, corresponding undecidability results require a more individual approach. In Section 7 we show that the disjunctivity problem for these grammars is undecidable too, in fact Π_2^0-complete.

For general Chomsky grammars we prove Π_3^0-completeness of the disjunctivity problem, thereby completing the characterization of the degrees of unsolvability of the disjunctivity problems for the basic language classes in the Chomsky hierarchy. Moreover, the Π_2^0-hardness results imply some negative results on the recursive presentability of the disjunctive sequences inside the standard language and complexity classes. These representability questions are addressed in some more detail in Section 6.

2 Basic Definitions and Facts

Let $\Sigma = \{0,1\}$ be the binary alphabet and let Σ^* and Σ^ω denote the set of (finite) words or strings and (infinite) sequences over Σ, respectively. We let z_n denote the $(n+1)$th string with respect to the length-lexicographical ordering of Σ^*, and let z_n^k denote the $(n+1)$th string of length k $(0 \leq n < 2^k)$. A subset of Σ^* is called a language or simply a set. Sometimes we identify the word z_n with the number n, i.e., identify languages with sets of natural numbers. Moreover, we identify a set with its characteristic function. I.e., we write $A(x) = 1$ if $x \in A$ and $A(x) = 0$ if $x \notin A$. The sequence $\alpha = \alpha(0)\alpha(1)\alpha(2)...$ defined by $\alpha(n) = A(z_n)$ is called the characteristic sequence of A and is denoted by $\chi(A)$. Conversely, the set A defined by $A(z_n) = \alpha(n)$ is called the set or language corresponding to α and is denoted by $S(\alpha)$. Note that $\chi(S(\alpha)) = \alpha$ and $S(\chi(A)) = A$. The prefix set $Prefix(\alpha)$ of a sequence α is defined by $\{x \in \Sigma^* : x \sqsubset \alpha\}$. Here $x \sqsubset \alpha$ denotes that x is a proper initial segment of α. We say that a word $x = x(0)...x(n-1)$ occurs in a sequence α (as a subword) if $\alpha(m)\alpha(m+1)...\alpha(m+n-1) = x(0)...x(n-1)$ for some number $m \geq 0$.

Definition 1. *(Jürgensen and Thierrin [5]) An infinite sequence α is* disjunctive *if every word $x \in \Sigma^*$ occurs in α.*

For a recent survey on disjunctive sequences see [3]. Disjunctive sequences are known in the literature also under some other names like rich (see e.g. [11]) or saturated (see e.g. [1]). Note that in a disjunctive sequence α any word x actually occurs infinitely often since besides x also the words x^2, x^3, ... occur. Since we will mainly look at the sets corresponding to disjunctive sequences, the following technical lemma which we state without proof will be very useful. Given a disjunctive sequence α and the corresponding set $A = S(\alpha)$, the first part of the lemma shows that for any word x there are infinitely many numbers n such that membership of the strings of length n in A guarantees an occurence of x in α. Namely x occurs as a subword in the subword $A(z_0^n)...A(z_{2^n-1}^n)$ of α. Part (b) further refines this observation: Given a word x of length 2^m there are infinitely many words w such that the membership relation in A of the words extending w by m bits ensures occurence of x in α, namely $x = A(wz_0^m)...A(wz_{2^m-1}^m)$.

Lemma 1. *Let α be a disjunctive sequence and let A be the set corresponding to α. Then the following holds.*

(a) For any word $x \in \Sigma^*$ there are infinitely many numbers $n \geq 0$ such that

$$\exists i(0 \leq i \leq 2^n - (|x| - 1) \text{ and } A(z_i^n)...A(z_{i+|x|-1}^n) = x) \qquad (1)$$

(b) For any word $x \in \Sigma^*$ *such that* $|x| = 2^m$ *for some number* $m > 0$ *there are infinitely many words* w *such that*

$$A(wz_0^m)...A(wz_{2^m-1}^m) = x. \tag{2}$$

(c) For any word $w \in \Sigma^*$, $wA = \{wv : v \in A\}$ *is disjunctive. In fact, any set* B *such that* $B \cap w\Sigma^* = wA$ *is disjunctive.*

We should mention that there is the concept of a disjunctive language in the literature (see e.g. [8]) which is related to the concept of a disjunctive sequence. A language A is called *disjunctive* if any two words x and y can be distinguished by the context of their occurences in words in the language A, i.e., if

$$\forall z, z' \in \Sigma^* \ (z \neq z' \Rightarrow \exists\, u, v \in \Sigma^* \ (uzv \in A \Leftrightarrow uz'v \notin A)) \tag{3}$$

holds. It has been shown that a sequence is disjunctive if and only if its prefix set is a disjunctive language. Moreover, one can easily show that for any disjunctive sequence α the corresponding language $S(\alpha)$ is a disjunctive language. There are disjunctive languages, however, which have nondisjunctive characteristic sequences. The linear language $A = \{ww^R : w \in \Sigma^*\}$ (where w^R is the reversal of w) is an example of such a language. So the notion of a language (or set) corresponding to a disjunctive sequence is stronger than the notion of a disjunctive language. In the following we will only look at disjunctive sequences and the languages corresponding to these sequences, not at disjunctive languages.

3 Computational Complexity

We start our investigation of computational properties of disjunctive sequences by shortly discussing their time complexity. We show that for any set A there is a disjunctive sequence α such that the set $S(\alpha)$ corresponding to α is equivalent to A under linear-time many-one reducibility. Roughly speaking, this says that the sets corresponding to disjunctive sequences may have any given time complexity. As we will also note, the corresponding fact for prefix sets fails.

We start with the observation that there is a disjunctive sequence such that the corresponding set is linear time computable.

Lemma 2. *Let* $D = \{z_i^n : z_n(i) = 1\}$. *Then* $\chi(D)$ *is disjunctive and* $D \in \mathrm{DTIME}(O(n))$.

Note that for the nth word z_n occurrence of z_n in $\chi(D)$ is guaranteed by the first $|z_n|$ words of length n, namely $D(z_0^n)...D(z_{|z_n|-1}^n) = z_n$. We omit the straightforward proof for $D \in \mathrm{DTIME}(O(n))$.

Recall that $A \leq_m^{lin} B$ *(A is many-one reducible to B in linear time)* if there is a linear-time computable function $f : \Sigma^* \to \Sigma^*$ such that $x \in A$ if and only if $f(x) \in B$ (for all $x \in \Sigma^*$); and that $A =_m^{lin} B$ *(A is many-one equivalent to B in linear time)* if $A \leq_m^{lin} B$ and $B \leq_m^{lin} A$.

Theorem 1. *For any set* A *there is a set* B *such that* $A =_m^{lin} B$ *and* $\chi(B)$ *is disjunctive.*

Proof. Fix A, choose D as in Lemma 2, and let $B = A \oplus D = \{0x : x \in A\} \cup \{1x : x \in B\}$. Then, by $D \in \mathrm{DTIME}(O(n))$, $A \leq_m^{lin} B$ and, by disjunctivity of $\chi(D)$ and by Lemma 1(c), $\chi(B)$ is disjunctive. □

For space complexity we easily obtain the corresponding results for logarithmic space in place of linear time. In fact in Theorem 1 we may replace *lin-m* reducibility by simultaneously linear-time and logarithmic-space bounded many-one reducibility.

Calude and Yu [4] have shown that there is a disjunctive sequence with prefix set in $\mathrm{DSPACE}(O(\log n))$. In fact, as one can easily check, for D as in Lemma 2, $Prefix(\chi(D)) \in \mathrm{DTIME}(O(n)) \cap \mathrm{DSPACE}(O(\log n))$.

Lemma 3. *There is a disjunctive sequence α such that*

$$Prefix(\alpha) \in \mathrm{DTIME}(O(n)) \cap \mathrm{DSPACE}(O(\log n)).$$

In contrast to Theorem 1, however, there are *lin-m* equivalence classes - in fact polynomial-time Turing equivalence classes - which do not contain the prefix set of any disjunctive sequence (in fact no prefix set at all). This follows from some general results on sets of low nonuniform complexity. Note that any prefix set A contains just one word of each length whence A is *sparse.* Any sparse set A possesses polynomial-time circuits, i.e., is a member of the class $\mathrm{P}/poly$ and it has been shown that there are polynomial-time Turing equivalence classes which do not intersect $\mathrm{P}/poly$ (see e.g. [2], Chapter 5). So, in order to obtain an analog of Theorem 1 for prefix sets we have to work with some weaker reducibilities. E.g., we can show that, for any set A, there is the prefix set $Prefix(\alpha)$ of a disjunctive sequence α which is exponential-time Turing equivalent to A.

4 Chomsky Hierarchy and Prefix Sets

We now will measure the complexity of disjunctive sequences in terms of the Chomsky hierarchy. We first review the results of Calude and Yu [4] on the prefix sets of disjunctive sequences. The key to these results is the collapse of the lower part of the Chomsky hierarchy for prefix sets.

Lemma 4. *(Calude and Yu [4]) For an infinite sequence $\alpha \in \Sigma^\omega$ the following are equivalent.*

(i) *α is almost periodic, i.e., there are words $v, w \in \Sigma^*$ such that $\alpha = vw^\omega$.*
(ii) *The prefix set $Prefix(\alpha)$ of α is regular.*
(iii) *The prefix set $Prefix(\alpha)$ of α is context free.*
(iv) *The prefix set $Prefix(\alpha)$ of α contains an infinite context free subset.*

Proof. (Following [3]) The implications *(i)* $\Rightarrow$ *(ii)* $\Rightarrow$ *(iii)* $\Rightarrow$ *(iv)* are straightforward. So it suffices to prove the implication *(iv)* $\Rightarrow$ *(i).* Assume that A is an infinite context free subset of $Prefix(\alpha)$. Then, by the pumping lemma

for context free languages, there are words u, v, w, x, y such that $vx \neq \lambda$ and $z_n = uv^n wx^n y \in A$ for all $n \geq 0$. It follows that

$$uv^0 wx^0 y \sqsubset uv^1 wx^1 y \sqsubset \ldots \; uv^n wx^n y \sqsubset uv^{n+1} wx^{n+1} y \; \ldots \sqsubset \alpha.$$

So, if $v \neq \lambda$ then $\alpha = uv^\omega$, and if $v = \lambda$ then $\alpha = uwx^\omega$. □

Theorem 2. *(Calude and Yu [4]) There is a disjunctive sequence α such that $Prefix(\alpha)$ is context sensitive but there is no disjunctive sequence with context free prefix set.*

Proof. The first part of the theorem is immediate by Lemma 3, the second part by Lemma 4. □

Before we will compare these results with the corresponding results for corresponding sets in place of prefix sets in the next section, we will shortly consider a further alternative approach for describing the Chomsky complexity of sequences. In [1] a sequence α is called *regular* if there is a deterministic finite automaton which *predicts* α, i.e., which on input $\alpha \upharpoonright n = \alpha(0)\ldots\alpha(n-1)$ computes $\alpha(n)$ (i.e., the computation ends in a terminal state if $\alpha(n) = 1$ and in a nonterminal state otherwise). Correspondingly we call a sequence α *context free* if there is a nondeterministic pushdown automaton which *predicts* α, i.e., which on input $\alpha \upharpoonright n$ has a run ending in a terminal state if and only if $\alpha(n) = 1$. In [1] we have shown that α is regular if and only if $Prefix(\alpha)$ is regular and, similarly, one can easily show that α is context free if and only if $Prefix(\alpha)$ is context free. Hence, by Lemma 4, regularity and context freeness of a sequence coincide.

Lemma 5. *For an infinite sequence $\alpha \in \Sigma^\omega$ the following are equivalent.*

(i) The sequence α is almost periodic.
(ii) The sequence α is regular, i.e., α is predicted by some deterministic finite automaton.
(iii) The sequence α is context free, i.e., α is predicted by some nondeterministic pushdown automaton.

Though, by the preceding lemmas, this approach turns out to coincide with the prefix set approach and to be too coarse for analysing disjunctive sequences of low complexity, by considering *partial* predictions in place of complete predictions we can characterize the disjunctive sequences as shown in [1].

We say that a deterministic finite automaton M with three types of states (+-states, −-states, and ?-states) *partially predicts* α if there are infinitely many n such that M on input $\alpha \upharpoonright n$ reaches a +-state or a −-state; and, whenever M reaches a +-state (−-state), then $\alpha(n) = 1$ ($\alpha(n) = 0$). Intuitively, M makes infinitely many predictions on α, and all of these predictions are correct.

Theorem 3. *(Ambos-Spies and Busse [1]) A sequence α is disjunctive if and only if α is not partially predictable by deterministic finite automata.*

Interestingly, in contrast to Lemma 5, partial predictability by finite automata and by pushdown automata differs. In fact there is a deterministic 1-counter automaton which partially predicts a disjunctive sequence.

Theorem 4. *(Merkle and Reimann [6]) There is a disjunctive sequence α which can be partially predicted by a deterministic 1-counter automaton.*

Proof. (Idea) The following example of a sequence α with the required properties is somewhat simpler than the examples given in [6]. Let α be the disjunctive sequence obtained by concatenating all binary words in order, i.e., $\alpha = z_0 z_1 z_2 \ldots$. Then, for any number n, the initial segment $\alpha \upharpoonright n$ contains at least as many occurrences of the bit 0 as of the bit 1. Moreover, the number of occurences is equal if and only if $\alpha \upharpoonright n$ consists of all words up to a given length, i.e., $\alpha \upharpoonright n = z_0^0 \ldots z_{2^m-1}^m$ for some number m. So, for such an n, $\alpha(n)$ will be the first bit of the word z_0^{m+1} which is a 0. A deterministic 1-counter automaton M which correctly predicts these occurences of zeroes in α, pushes a 0 on his stack when it reads a 0, pops a 0 from the stack when it reads a 1, and predicts the next bit to be a 0 if the stack is empty. □

5 Chomsky Hierarchy and Corresponding Sets

In the preceding section we have seen that, due to the collapse of the lower part of the Chomsky hierarchy for prefix sets, we obtain only a very coarse classification of disjunctive sequences in terms of the location of their prefix sets in the Chomsky hierarchy. Since every binary language corresponds to a binary sequence, the identification of a sequence with its correspondent set rather than its prefix set will avoid such collapses and will provide a finer scale for measuring the complexity of disjunctive sequences. As we will show next, this approach actually yields a better lower and upper bound on the minimum complexity of disjunctive sequences: No regular set corresponds to a disjunctive sequence but there are disjunctive sequences with corresponding context free - in fact linear - sets. These results were already implicitly obtained in [1] were we investigated some automatic forcing notions closely related to disjunctivity. In the following we give some more direct and simplified proofs of these results.

Theorem 5. *Let α be a disjunctive sequence and let A be the set corresponding to α. Then A is not regular.*

Proof. For a contradiction assume that A is regular. Fix a deterministic finite automaton M which accepts A. Let p be the number of states of M and fix m such that $p < 2^m$. Then, for any set S,

$$|S| = 2^m \Rightarrow \exists u, u' \in S(u \neq u' \text{ and } \forall v(uv \in A \Leftrightarrow u'v \in A)). \tag{4}$$

(Namely, by $p < 2^m$, there are strings $u, u' \in S$, $u \neq u'$, such that M is in the same state after reading u and u'.) Now let

$$x_i = 0^i 1 0^{2^m-(i+1)} (0 \leq i < 2^m) \;\; \& \;\; x = x_0 \ldots x_{2^m-1}. \tag{5}$$

Then $|x| = |x_0| + ... + |x_{2^m-1}| = 2^m 2^m = 2^{2m}$. So, by disjunctivity of α and by Lemma 1 (b), there is a word w such that

$$A(wz_0^{2m})...A(wz_{2^{2m}-1}^{2m}) = x. \tag{6}$$

By choice of x this implies

$$A(wz_i^m z_0^m)...A(wz_i^m z_{2^m-1}^m) = x_i \ (0 \leq i < 2^m)$$

whence, by choice of the strings x_i,

$$A(wz_i^m z_j^m) = 1 \Leftrightarrow i = j \ (0 \leq i, j < 2^m). \tag{7}$$

So, for $S = \{wz_i^m : i < 2^m\}$, $|S| = 2^m$ and for any $u \neq u' \in S$ there is a word v (namely, for $u = wz_i^m$, $v = z_i^m$) such that $uv \in A$ but $u'v \notin A$. But this contradicts (4). □

Theorem 6. *There is a disjunctive sequence α such that the language A corresponding to α is linear.*

Proof. For a nonempty word $x = a_1 \dots a_n$ let $x^D = a_1 a_1 \dots a_n a_n$ be the *duplication* of x and let $x^R = a_n \dots a_1$ be the *reversal* of x ($\lambda^D = \lambda^R = \lambda$). Then the required set A is defined by

$$A = \{x_1^D 01 x_2^D 01 \dots x_n^D 10 x_m^R : 1 \leq m \leq n \ \& \ x_1, \dots, x_n \in \Sigma^+\}.$$

Intuitively, A is a set of coded sequences $\langle x_1, \dots, x_n, x_{n+1}\rangle$ of strings where the final component x_{n+1} coincides with one of the previous components $x_1, \dots, x_n$. The coding is chosen in such a way that one can easily give a linear grammar generating A.

It remains to show that the characteristic sequence α of A is disjunctive. I.e., given a string x, we have to show that x occurs in α. W.l.o.g. we may assume that $|x| = 2^n$ for some $n \geq 1$ and that at least one 1 occurs in x (otherwise consider an appropriate extension of x). I.e. $x = a_0 \dots a_{2^n-1}$ and $P = \{j : 0 \leq j < 2^n \ \& \ a_j = 1\} \neq \emptyset$. Fix $0 \leq j_1 < j_2 < \dots < j_l < 2^n$ such that $P = \{j_1, \dots, j_l\}$ and let $w = ((z_{j_1}^n)^R)^D 01((z_{j_2}^n)^R)^D 01 \dots ((z_{j_l}^n)^R)^D$ and $w_j = w10z_j^n$ $(0 \leq j < 2^n)$. Then $w_0, \dots, w_{2^n-1}$ are consecutive words, whence the sequence $A(w_0) \dots A(w_{2^n-1})$ occurs in α. On the other hand, by definition, $w_j \in A$ iff $j \in P$ whence $x = A(w_0) \dots A(w_{2^n-1})$. So x occurs in α which completes the proof. □

6 Recursive Presentability

Recursive presentability is a central notion in computational complexity theory and computability theory. Intuitively, a class C is recursively presentable (r.p.) if it only contains recursive languages and its members are effectively given in

a uniform way. The standard computational complexity classes and the common recursive language classes studied in formal language theory are recursively presentable. An arbitrary class C (which may also contain nonrecursive sets) is called locally recursively presentable (l.r.p.) if its intersection with any recursively presentable class which is closed under finite variants (c.f.v.) is recursively presentable or empty.

Here we look at presentability questions for the class DIS of languages corresponding to disjunctive sequences. In particular, we are interested in the question whether, for any of the common complexity and Chomsky classes C, $\mathrm{DIS} \cap \mathrm{C}$ is recursively presentable. We first formally introduce the basic notions.

Definition 2. *A class* C *of recursive sets is* recursively presentable (r.p.) *if there is a binary recursive set U such that* $\mathrm{C} = \{U_e : e \geq 0\}$ *(where* $U_e = \{x : (e, x) \in U\}$*). U is called a* universal set *for C or an* indexing *of C.*

A class C *is* closed under finite variants (c.f.v.) *if any finite variant of a set in C is a member of C too.*

A class C *is* locally recursively presentable (l.r.p.) *if, for any r.p. class* D *which is closed under finite variants, the intersection of* C *and* D *is empty or recursively presentable.*

Recursively presentable classes are also called *uniformly recursive.* Note that the class DIS (hence its complement $\overline{\mathrm{DIS}}$) is c.f.v.

We first observe that the class of sets corresponding to non-disjunctive sequences is locally recursively presentable.

Theorem 7. *The class* $\overline{\mathrm{DIS}}$ *is locally recursively presentable.*

Proof. (Idea) Let C be any r.p. and c.f.v. class such that $\overline{\mathrm{DIS}} \cap \mathrm{C}$ is nonempty, say $A \in \overline{\mathrm{DIS}} \cap \mathrm{C}$. We have to show that $\overline{\mathrm{DIS}} \cap \mathrm{C}$ is r.p. again. Let U be a recursive universal set for C. Then we obtain a recursive universal set V for $\overline{\mathrm{DIS}} \cap \mathrm{C}$ by letting

$$V_{\langle e,x\rangle}(z) = \begin{cases} U_e(z) & \text{if } x \text{ does not occur in } \chi(U_e) \upharpoonright z \\ A(z) & \text{otherwise.} \end{cases}$$

□

Since the characteristic sequence of Σ^* is not disjunctive, it follows from Theorem 7 that, for any r.p. and c.f.v. class C such that $\Sigma^* \in \mathrm{C}$, $\mathrm{C} \cap \overline{\mathrm{DIS}}$ is recursively presentable. So, for instance $\mathrm{REG} \cap \overline{\mathrm{DIS}}$, $\mathrm{LIN} \cap \overline{\mathrm{DIS}}$, $\mathrm{CF} \cap \overline{\mathrm{DIS}}$ and $\mathrm{CS} \cap \overline{\mathrm{DIS}}$ are recursively presentable.

Since Schmidt [7] has observed that any sufficiently closed r.p. class cannot be split into two r.p. and c.f.v. classes, Theorem 7 implies that the class of sets corresponding to disjunctive sequences is not l.r.p.

Theorem 8. *(Schmidt [7]) Let* C *be a nonempty c.f.v. class which is downward closed under* $\leq_m^{lin}$*. Then there is no splitting of* C *into two r.p. and c.f.v. classes* C_0 *and* C_1 *(i.e., there are no r.p. and c.f.v. classes* C_0 *and* C_1 *such that* $\mathrm{C}_0 \cup \mathrm{C}_1 = \mathrm{C}$ *and* $\mathrm{C}_0 \cap \mathrm{C}_1 = \emptyset$*).*

Corollary 1. *Let C be any class which contains* $\mathrm{DTIME}(O(n))$. *Then* $\mathrm{C} \cap \mathrm{DIS}$ *is nonempty and not recursively presentable.*

Proof. (Sketch) As one can easily observe, it suffices to show that the class $\mathrm{DTIME}(O(n)) \cap \mathrm{DIS}$ is not recursively presentable. Since $\mathrm{DTIME}(O(n)) \cap \mathrm{DIS}$ and $\mathrm{DTIME}(O(n)) \cap \overline{\mathrm{DIS}}$ are c.f.v. and, by Theorem 1, nonempty, and since, by Theorem 7, $\mathrm{DTIME}(O(n)) \cap \overline{\mathrm{DIS}}$ is r.p. this follows from Theorem 8. □

The following two corollaries are immediate.

Corollary 2. *The class* DIS *is not locally recursively presentable.*

Corollary 3. *The class* $\mathrm{CS} \cap \mathrm{DIS}$ *of the context sensitive sets corresponding to disjunctive sequences is not recursively presentable.*

Schmidt's nonsplitting theorem, however, does not help us in deciding whether for the low Chomsky classes C intersecting DIS, like CF or LIN, the class $\mathrm{C} \cap \mathrm{DIS}$ is recursively presentable. As observed by Schmidt, the nonsplitting theorem fails for C = CF (and similarly for C = LIN): The classes $\mathrm{CF} \cap \mathrm{FIN}$ and $\mathrm{CF} \cap \mathrm{INF}$ where FIN and INF denote the classes of finite and infinite languages, respectively, provide a counter example. So answering the question whether $\mathrm{CF} \cap \mathrm{DIS}$ and $\mathrm{LIN} \cap \mathrm{DIS}$ are recursively presentable will require a different approach. By determining the degree of unsolvability of the disjunctivity problem for linear grammars in the next section, we will show that the classes $\mathrm{CF} \cap \mathrm{DIS}$ and $\mathrm{LIN} \cap \mathrm{DIS}$ are not recursively presentable too.

7 Undecidability and Degrees of Unsolvability

We conclude our investigations of computational aspects of disjunctive sequences with some basic undecidability results and some results on degrees of unsolvability.

In the setting of computability theory it is natural to look at the complexity of the index sets of the recursive (computable) disjunctive sequences and the recursively enumerable (computably enumerable) disjunctive sequences. Fix φ such that $\{\varphi_e : e \geq 0\}$ is a standard enumeration of the 0-1-valued partial recursive functions, and let W be the domain of φ; i.e., $\{W_e : e \geq 0\}$ is a standard enumeration of the recursively enumerable (r.e.) sets. Then we are interested in the index sets of the classes

$$\mathrm{DIS}_{rec} = \{f : f \text{ total recursive and the sequence } f(0)f(1)f(2)\ldots \text{ is disjunctive}\}$$

and

$$\mathrm{DIS}_{r.e.} = \{A : A \text{ r.e } \&\ A \text{ corresponds to a disjunctive sequence}\},$$

i.e.. in the index sets

$$I(\mathrm{DIS}_{rec}) = \{e : \varphi_e \text{ total and } \varphi_e(0)\varphi_e(1)\varphi_e(2)\ldots \text{ disjunctive}\}$$

and

$$I(\mathrm{DIS}_{r.e.}) = \{e : W_e \text{ corresponds to a disjunctive sequence}\}.$$

By Rice's theorem these index sets are nonrecursive. For locating these problems in the arithmetical hierarchy, a straightforward analysis of the definitions of these index sets shows that $I(\mathrm{DIS}_{rec})$ is Π_2^0 while $I(\mathrm{DIS}_{r.e.})$ is Π_3^0. On the other hand, since $I(\mathrm{DIS}_{rec})$ only contains indices of total functions, by a well-known easy folklore theorem of recursion theory, $I(\mathrm{DIS}_{rec})$ is Π_2^0-hard, whence this index set is Π_2^0-complete. For the index set of the r.e. disjunctive sequences we can show that the Σ_3^0-complete index set REC of the recursive r.e. sets is many-one reducible to the complement of $I(\mathrm{DIS}_{r.e.})$ whence $I(\mathrm{DIS}_{r.e.})$ is Π_3^0-complete. The proof of this fact is considerably more involved and due to lack of space omitted.

Theorem 9. *The index set $I(\mathrm{DIS}_{rec})$ is Π_2^0-complete, the index set $I(\mathrm{DIS}_{r.e.})$ is Π_3^0-complete.*

By coincidence of the Chomsky-0-languages with the r.e. languages, the second part of Theorem 9 implies that the disjunctivity problem for Chomsky-0-grammars is Π_3^0-complete. For the other Chomsky classes the first part of the theorem gives an upper bound on the complexity of the corresponding disjunctivity problems: since these classes contain only recursive languages, in fact are r.p., their disjunctivity problems are in Π_2^0. Moreover, the proof of the first part of the theorem can be easily modified to show Π_2^0-completeness of the disjunctivity problem for context sensitive grammars. On the other hand, since no regular language corresponds to a disjunctive sequence, the disjunctivity problem for regular grammars is trivially decidable. The interesting instances of the disjunctivity problem are the ones for context free grammars and linear grammars. In the following we show that the disjunctivity problem for linear grammars has the highest possible degree of unsolvability, namely is Π_2^0-complete. A fortiori this also implies Π_2^0-completeness of the disjunctivity problem for context free grammars.

Theorem 10. *Let $\{G_e : e \geq 0\}$ be an effective enumeration of the linear grammars with terminal alphabet $\Sigma = \{0,1\}$. Then $\{e : \chi(L(G_e))$ disjunctive$\}$ is Π_2^0-complete.*

Due to lack of space we omit the proof. We only remark that the proof is based on a reduction of the Π_2^0-complete co-infinity problem for linear grammars to the disjunctivity problem for these grammars. This reduction works with variants of the disjunctive linear language introduced in the proof of Theorem 6.

Corollary 4. *For any r.p. and c.f.v. class* C *which contains* LIN as *a subclass, the class* $\mathrm{C} \cap \mathrm{DIS}$ *is not recursively presentable. In particular,* $\mathrm{LIN} \cap \mathrm{DIS}$ *and* $\mathrm{CF} \cap \mathrm{DIS}$ *are not recursively presentable.*

Proof. It suffices to show that $\mathrm{LIN} \cap \mathrm{DIS}$ is not r.p. For a contradiction assume that $\mathrm{LIN} \cap \mathrm{DIS}$ is r.p. and fix a recursive universal set U. Then, for $\{G_e : e \geq 0\}$ as in Theorem 10, $L(G_e)$ corresponds to a disjunctive sequence if and only if $\exists e' \forall x (x \in L(G_e) \Leftrightarrow x \in U_{e'})$. It follows that $\{e : \chi(L(G_e))$ disjunctive$\}$ is a Σ_2^0-set hence not Π_2^0-hard. But this contradicts Theorem 10. □

8 Open Problems

We have shown that no regular language corresponds to a disjunctive sequence but that there are linear languages corresponding to such sequences. It is natural to ask whether there are other interesting subclasses of the context free languages which contain languages with disjunctive characteristic sequence. In particular, is there a determinstic context free language corresponding to a disjunctive sequence? Another interesting question is the analysis of the Chomsky complexity of normal sequences. In particular, is there a context free language corresponding to a normal sequence?

References

1. K. Ambos-Spies and E. Busse, Automatic forcing and genericity: on the diagonalization strength of finite automata, DMTCS 2003, Lecture Notes in Computer Science 2731 (2003) 97-108, Springer.
2. J. L. Balcazar, J. Diaz and J. Gabarro, Structural Complexity I, Second Ed., 1995, Springer.
3. C. S. Calude, L. Priese and L. Staiger, Disjunctive sequences: an overview, CDMTCS Research Report 63, 1997.
4. C. Calude and S. Yu, Language-theoretic complexity of disjunctive sequences, Discrete Applied Mathematics 80 (1997) 203-209.
5. H. Jürgensen and G. Thierrin, Some structural properties of ω-languages, 13th Nat. School with Internat. Participation "Applications of Mathematics in Technology", Sofia, 1988, 56-63.
6. W. Merkle and J. Reimann, On selection functions that do not preserve normality, Mathematical Foundations of Computer Science 2003, Lecture Notes in Computer Science 2747 (2003) 602–611, Springer.
7. D. Schmidt, The recursion-theoretic structure of complexity classes, Theor. Comput. Sci. 38 (1985) 143-156.
8. H. J. Shyr, Disjunctive languages on a free monoid, Information and Control 34 (1977) 123-129.
9. L. Staiger, Reguläre Nullmengen, Elektron. Informationsverarb. Kybernetik EIK 12 (1976) 307-311.
10. L. Staiger, Finite-state ω-languages, J. Comput. System Sci. 27 (1983) 434-448.
11. L. Staiger, Rich ω-words and monadic second-order arithmetic, Computer Science Logic, 11th Int. Workshop, CSL'97, Lecture Notes in Computer Science 1414 (1997) 478-490, Springer.
12. L. Staiger, How large is the set of disjunctive sequences?, J. Universal Comput. Sci. 8 (2002) 348 - 362.

Decidability of Trajectory-Based Equations

Michael Domaratzki and Kai Salomaa

School of Computing, Queen's University,
Kingston, ON Canada K7L 3N6
{domaratz,ksalomaa}@cs.queensu.ca

Abstract. We consider the decidability of existence of solutions to language equations involving the operations of shuffle and deletion along trajectories. These operations generalize the operations of catenation, insertion, shuffle, quotient, sequential and scattered deletion, as well as many others. Our results are constructive in the sense that if a solution exists, it can be effectively represented. We show both positive and negative decidability results.

1 Introduction

Work on language equations is one of the core areas of formal language theory [12]. Much of the classical work deals with equations over the Boolean operations, catenation and Kleene closure. Recent research [1,2,6,8,10] has investigated the question of the decidability of existence of solutions to equations of the form $X_1 \diamond X_2 = X_3$, where $\diamond$ is a binary operation on languages, and some of X_1, X_2, X_3 are fixed languages and some are unknowns.

As a particular case of the above type of equations we get the shuffle decomposition problem for regular languages, that is, the question whether a given regular language can be written as a shuffle of two languages in a non-trivial way. In spite of its apparent simplicity the question remains open for general regular languages [1,6]. The decomposition of regular languages with respect to catenation is known to be decidable [11,15].

In this paper, we focus on operations $\diamond$ which are taken from the class of operations defined by shuffle on trajectories [13]. Shuffle on trajectories provides a unifying framework for studying various language composition operations. Introduced independently by the first author [3] and Kari and Sosík [9], the complementary notion of *deletion along trajectories* provides, in the sense of Kari [7], the inverse of shuffle on trajectories, and makes it possible to attack in a systematic way questions of decidability of existence of solutions to equations involving shuffle on trajectories. Some decidability results have already been completed by the first author [3] and Kari and Sosík [9].

We establish for certain classes of trajectories the decidability of the existence of a decomposition for a given regular language. However, our results leave open the question for the set of trajectories $(0+1)^*$ corresponding to ordinary shuffle [1,6]. We also show that for given regular languages L_1, L_2, and R we can decide whether or not there exists a set of trajectories T such that $L_1 \sqcup_T L_2 = R$,

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 723–734, 2004.
© Springer-Verlag Berlin Heidelberg 2004

where $\sqcup_T$ denotes shuffle along the set of trajectories T. To conclude we provide undecidability results for equations involving one or two variables. Proofs of the results given in this paper will be given in a future, final version of this paper.

2 Definitions and Preliminary Results

For additional background in formal languages and automata theory, please see Yu [16]. Let Σ be a finite set of symbols, called *letters*. Then Σ^* is the set of all finite sequences of letters from Σ, which are called words. The empty word ϵ is the empty sequence of letters. The *length* of a word $w = w_1 w_2 \cdots w_n \in \Sigma^*$, where $w_i \in \Sigma$, is n, and is denoted $|w|$. Note that ϵ is the unique word of length 0. A *language* L is any subset of Σ^*. By $\overline{L}$, we mean $\Sigma^* - L$, the complement of L. If L is a language over alphabet Σ, we denote by alph(L) the set of all letters of Σ occurring in words of L ($\mathrm{alph}(L) \subseteq \Sigma$).

A *deterministic finite automaton* (DFA) is a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where Q is the finite set of states, Σ is the alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the distinguished start state, and $F \subseteq Q$ is the set of final states. We extend δ to $Q \times \Sigma^*$ in the usual way. A word $w \in \Sigma^*$ is accepted by M if $\delta(q_0, w) \in F$. The *language accepted* by M, denoted *L(M),* is the set of all words accepted by M. A language is called *regular* if it is accepted by some DFA. A DFA $M = (Q, \Sigma, \delta, q_0, F)$ is *complete* if $\delta(q, a)$ is defined for all $(q, a) \in Q \times \Sigma$.

A *nondeterministic finite automaton* (NFA) is a five-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where Q, Σ, q_0 and F are as in the deterministic case, while $\delta : Q \times (\Sigma \cup \epsilon) \to 2^Q$ is the nondeterministic transition function. Again, δ is extended to $Q \times \Sigma^*$ in the natural way. A word w is accepted by M if $\delta(q_0, w) \cap F \neq \emptyset$. It is known that the language accepted by an NFA is regular.

The shuffle on trajectories operation is a method for specifying the ways in which two input words may be combined to form a result. Each trajectory $t \in \{0,1\}^*$ with $|t|_0 = n$ and $|t|_1 = m$ specifies the manner in which we can form the shuffle on trajectories of two words of length n (as the left input word) and m (as the right input word). The word resulting from the shuffle along t will have a letter from the left input word in position i if the i-th symbol of t is 0, and a letter from the right input word in position i if the i-th symbol of t is 1.

We now recall the formal definition of shuffle on trajectories, originally given by Mateescu *et al.* [13]. Let $x, y \in \Sigma^*$ and $t \in \{0,1\}^*$. We denote the shuffle of x and y along the trajectory t by $x \sqcup_t y$.

If $x = ax'$, $y = by'$ (with $a, b \in \Sigma$) and $t = et'$ (with $e \in \{0,1\}$), we have that

$$x \sqcup_{et'} y = \begin{cases} a(x' \sqcup_{t'} by') \text{ if } e = 0; \\ b(ax' \sqcup_{t'} y') \text{ if } e = 1. \end{cases}$$

If $x = ax'$ ($a \in \Sigma$) and $y = \epsilon$, then

$$x \sqcup_{et'} \epsilon = \begin{cases} a(x' \sqcup_{t'} \epsilon) \text{ if } e = 0; \\ \emptyset \qquad\qquad \text{otherwise.} \end{cases}$$

If $x = \epsilon$ and $y = by'$ $(b \in \Sigma)$, then

$$\epsilon \sqcup\!\sqcup_{et'} y = \begin{cases} b(\epsilon \sqcup\!\sqcup_{t'} y') & \text{if } e = 1; \\ \emptyset & \text{otherwise.} \end{cases}$$

If $x = y = \epsilon$, then $\epsilon \sqcup\!\sqcup_t \epsilon = \epsilon$ if $t = \epsilon$ and $\emptyset$ otherwise. Finally, $x \sqcup\!\sqcup_\epsilon y = \emptyset$ if $\{x, y\} \neq \{\epsilon\}$.

We extend shuffle on trajectories to a set $T \subseteq \{0,1\}^*$ of trajectories as follows:

$$x \sqcup\!\sqcup_T y = \bigcup_{t \in T} x \sqcup\!\sqcup_t y.$$

Further, for $L_1, L_2 \subseteq \Sigma^*$, we define

$$L_1 \sqcup\!\sqcup_T L_2 = \bigcup_{\substack{x \in L_1 \\ y \in L_2}} x \sqcup\!\sqcup_T y.$$

We now give the definition of *deletion along trajectories* [3,9], which models deletion operations controlled by a set of trajectories. Let $x, y \in \Sigma^*$ be words with $x = ax'$, $y = by'$ $(a, b \in \Sigma)$. Let t be a word over $\{i, d\}$ such that $t = et'$ with $e \in \{i, d\}$. Then we define $x \rightsquigarrow_t y$ as follows:

$$x \rightsquigarrow_t y = \begin{cases} a(x' \rightsquigarrow_{t'} by') & \text{if } e = i; \\ x' \rightsquigarrow_{t'} y' & \text{if } e = d \text{ and } a = b; \\ \emptyset & \text{otherwise.} \end{cases}$$

Also,

$$x \rightsquigarrow_t \epsilon = \begin{cases} a(x' \rightsquigarrow_{t'} \epsilon) & \text{if } e = i; \\ \emptyset & \text{otherwise.} \end{cases}$$

Further, $\epsilon \rightsquigarrow_t y = \epsilon$ if $t = y = \epsilon$. Otherwise, $\epsilon \rightsquigarrow_t y = \emptyset$. Finally, $x \rightsquigarrow_\epsilon y = \emptyset$ if $x \neq \epsilon$.

Let $T \subseteq \{i, d\}^*$. Then

$$x \rightsquigarrow_T y = \bigcup_{t \in T} x \rightsquigarrow_t y.$$

We extend this to languages as expected: Let $L_1, L_2 \subseteq \Sigma^*$ and $T \subseteq \{i, d\}^*$. Then

$$L_1 \rightsquigarrow_T L_2 = \bigcup_{\substack{x \in L_1 \\ y \in L_2}} x \rightsquigarrow_T y.$$

Note that $\rightsquigarrow_T$ is neither an associative nor a commutative operation on languages, in general. For the closure properties of $\rightsquigarrow_T$, please see the work of the first author [3] or Kari and Sosík [9].

Given two binary word operations $\diamond, \star : (\Sigma^*)^2 \to 2^{\Sigma^*}$, we say that $\diamond$ is a *left-inverse* of $\star$ [7, Defn. 4.5] if, for all $u, v, w \in \Sigma^*$, $w \in u \star v \iff u \in w \diamond v$. We say that $\diamond$ is a *right-inverse* of $\star$ [7, Defn. 4.1] if, for all $u, v, w \in \Sigma^*$, $w \in u \star v \iff v \in u \diamond w$.

Let $\tau : \{0,1\}^* \to \{i, d\}^*$ be the morphism given by $\tau(0) = i$ and $\tau(1) = d$. The following result will prove useful [3,9]:

Theorem 1. *Let $T \subseteq \{0,1\}^*$ be a set of trajectories. Then $\sqcup\!\sqcup_T$ and $\rightsquigarrow_{\tau(T)}$ are left-inverses of each other.*

Similarly, let $\pi : \{0,1\}^* \to \{i,d\}^*$ be the morphism given by $\pi(0) = d$ and $\pi(1) = i$. Given an operation $\diamond$, let $\diamond^R$ be the operation defined by $x \diamond^R y = y \diamond x$ for all $x, y \in \Sigma^*$.

Theorem 2. *Let $T \subseteq \{0,1\}^*$ be a set of trajectories. Then $\sqcup\!\sqcup_T$ and $(\rightsquigarrow_{\pi(T)})^R$ are right-inverses of each other.*

The following result of Kari [7, Thms. 4.2 and 4.6] allows us to find solutions to equations involving shuffle on trajectories.

Theorem 3. *Let L, R be languages over Σ and $\diamond, \star$ be two binary word operations, which are right-inverses (resp., left-inverses) to each other. If the equation $L \diamond X = R$ (resp., $X \diamond L = R$) has a solution $X \subseteq \Sigma^*$, then the language $R' = \overline{L \star \overline{R}}$ (resp., $R' = \overline{\overline{R} \star L}$) is also a solution of the equation. Moreover, R' is a superset of all other solutions of the equation.*

3 Decidability of Shuffle Decompositions

Say that a language L has a *non-trivial shuffle decomposition* with respect to a set of trajectories $T \subseteq \{0,1\}^*$ if there exist $X_1, X_2 \neq \{\epsilon\}$ such that $L = X_1 \sqcup\!\sqcup_T X_2$.

In this section, we are concerned with giving a class of sets of trajectories $T \subseteq \{0,1\}^*$ such that it is decidable, given a regular language R, whether R has a non-trivial shuffle decomposition with respect to T. For $T = (0 + 1)^*$, this is an open problem [1,6]. While we do not settle this open problem, we establish a non-trivial generalization of the results of Kari and Kari and Thierrin [7,8, 10,11], which leads to a large class of examples of trajectories where the shuffle problem can be proven to be decidable.

A language $L \subseteq \Sigma^*$ is bounded if there exist $w_1, w_2, \ldots, w_n \in \Sigma^*$ such that $L \subseteq w_1^* w_2^* \cdots w_n^*$. Say that L is *letter-bounded* if $w_i \in \Sigma$ for all $1 \leq i \leq n$.

We now define a class of letter-bounded sets of trajectories, called *i-regular* sets of trajectories, which will have strong closure properties. In particular, we can delete, along an *i*-regular letter-bounded set of trajectories, any language from a regular language and the resulting language will be regular. This will allow us to solve the corresponding shuffle decomposition decidability problems.

Let Δ_m be the alphabet $\Delta_m = \{\#_1, \#_2, \ldots, \#_m\}$ for any $m \geq 1$. We define a class of regular substitutions from $(d + \Delta_m)^*$ to $2^{(i+d)^*}$, denoted $\mathfrak{S}_m$, as follows: a regular substitution $\varphi : (d + \Delta_m)^* \to 2^{(i+d)^*}$ is in $\mathfrak{S}_m$ if both (a) $\varphi(d) = \{d\}$; and (b) for all $1 \leq j \leq m$, there exist $a_j, b_j \in \mathbb{N}$ such that $\varphi(\#_j) = i^{a_j}(i^{b_j})^*$.

For all $m \geq 1$, we also define a class of languages over the alphabet $d + \Delta_m$, denoted $\mathfrak{T}_m$, as the set of all languages $T \subseteq \#_1 d^* \#_2 d^* \cdots \#_{m-1} d^* \#_m$. Define the class of trajectories $\mathfrak{I}$ as follows:

$$\mathfrak{I} = \{T \subseteq \{i,d\}^* \ : \ \exists m \geq 1, T_m \in \mathfrak{T}_m, \varphi \in \mathfrak{S}_m \text{ such that } T = \varphi(T_m)\}.$$

If $T \in \mathfrak{I}$, we say that T is *i-regular*. Note that *i*-regular languages are not necessarily regular languages. The following result states that the condition that T is *i*-regular is sufficient for showing that $R \rightsquigarrow_T L$ is regular for all regular languages R and all languages L.

Theorem 4. *Let $T \in \mathfrak{I}$. Then for all regular languages R and all languages L, $R \rightsquigarrow_T L$ is a regular language.*

We note that we can establish that if T is a finite union of *i*-regular languages, or if we relax condition (b) of membership in $\mathfrak{S}_m$ to $\varphi(\#_j) \subseteq i^*$, then the conclusion of Theorem 4 still holds. We also note that if T is not letter-bounded, it may define an operation which does not preserve regularity in the sense of Theorem 4. In particular, we note that for $T = (di)^*$,

$$(a^2)^*(b^2)^* \rightsquigarrow_T \{a^n b^n \ : \ n \geq 0\} = \{a^n b^n \ : \ n \geq 0\},$$

a non-regular context-free language (CFL). For $T = (i + d)^*$, we have that

$$((ab)^*\#(ab)^* \rightsquigarrow_T \{a^n \# b^n \ : \ n \geq 0\}) \cap b^*a^* = \{b^n a^n \ : \ n \geq 0\}.$$

Further, if T is letter-bounded but not *i*-regular, then T may not preserve regularity. For example, let $T = \{i^n d i^n \ : \ n \geq 0\}$. Then $a^*bc^* \rightsquigarrow_T \{b\} = \{a^n c^n \ : \ n \geq 0\}$.

As an example of Theorem 4, consider $T = \{d^n i^m d^n \ : \ n, m \geq 0\}$. It is easily verified that $T \in \mathfrak{I}$. Thus, the language $R \rightsquigarrow_T L$ is regular for all regular languages R and all languages L. For any language $L \subseteq \Sigma^*$, define $sq(L) = \{x^2 \ : \ x \in L\}$. Consider then that

$$R \rightsquigarrow_T sq(L) = \{w \ : vwv \in R, v \in L\}.$$

This precisely defines the *middle-quotient* operation, which has been investigated by Meduna [14] for linear CFLs (LCFLs). Let $R \mid L$ denote the middle quotient of R by L, i.e., $R \mid L = R \rightsquigarrow_T sq(L)$. Thus, we can immediately conclude the following result, which was not considered by Meduna:

Theorem 5. *Given a regular language R and arbitrary language L, the language $R \mid L$ is regular.*

We now return to letter-bounded sets of trajectories. Using a result of Ginsburg and Spanier [4], it is not hard to show that every letter-bounded set of trajectories which is regular is also a finite union of *i*-regular sets of trajectories. In particular, we have the following corollary of Theorem 4.

Corollary 1. *Let $T \subseteq \{i, d\}^*$ be a letter-bounded regular set of trajectories. Then for all regular languages R, there are only finitely many regular languages L' such that $L' = R \rightsquigarrow_T L$ for some language L. Furthermore, given effective constructions for T and R, we can effectively construct a finite set $\mathcal{S}$ of regular languages such that if $L' = R \rightsquigarrow_T L$ for some $L \subseteq \Sigma^*$, then $L' \in \mathcal{S}$.*

Theorem 6. *Let $T \subseteq \{0,1\}^*$ be a letter-bounded regular set of trajectories. Let R be a regular language over an alphabet Σ. Then there exists a natural number $n \geq 1$ such that there are n distinct regular languages Y_i with $1 \leq i \leq n$ such that for any $L \subseteq \Sigma^*$ the following are equivalent:*

(a) there exists a solution $Y \subseteq \Sigma^*$ to the equation $L \sqcup_T Y = R$;
(b) there exists an index i with $1 \leq i \leq n$ such that $L \sqcup_T Y_i = R$.

The languages Y_i can be effectively constructed, given effective constructions for T and R. Further, if Y is a solution to $L \sqcup_T Y = R$, then there is some $1 \leq i \leq n$ such that $Y \subseteq Y_i$.

Proof. Consider the finite set $\mathcal{S}_1(T,R)$ satisfying $\overline{R} \rightsquigarrow_{\pi(T)} L \in \mathcal{S}(T,R)$ for all $L \subseteq \Sigma^*$. This set is finite and effectively constructible by Corollary 1. Let $\mathcal{S}(T,R) = \text{co-}\mathcal{S}_1(T,R)$.

Let L be arbitrary. Thus, if $L \sqcup_T Y = R$, then $Y \subseteq X$ for some $X \in \mathcal{S}(T,R)$ by Theorems 2 and 3, and $L \sqcup_T X = R$. Further, each language in $\mathcal{S}(T,R)$ is regular, by Corollary 1. Thus, (a) implies (b). The implication (b) implies (a) is trivial. □

The symmetric result also holds:

Theorem 7. *Let $T \subseteq \{0,1\}^*$ be a letter-bounded regular set of trajectories. Let R be a regular language over an alphabet Σ. Then there exists a natural number $n \geq 1$ such that there are n distinct regular languages Z_i with $1 \leq i \leq n$ such that for any $L \subseteq \Sigma^*$ the following are equivalent:*

(a) there exists a solution $Z \subseteq \Sigma^*$ to the equation $Z \sqcup_T L = R$;
(b) there exists an index i with $1 \leq i \leq n$ such that $Z_i \sqcup_T L = R$.

The languages Z_i can be effectively constructed, given effective constructions for T and R. Further, if Z is a solution to $Z \sqcup_T L = R$, then there is some $1 \leq i \leq n$ such that $Z \subseteq Z_i$.

We can now give our decomposition decidability result:

Theorem 8. *Let $T \subseteq \{0,1\}^*$ be a letter-bounded regular set of trajectories. Then given a regular language R, it is decidable whether there exist X_1, X_2 such that $X_1 \sqcup_T X_2 = R$.*

Proof. Let $\mathcal{S}(T,R)$ be the set of languages described by Theorem 6 and, analogously, let $\mathcal{T}(T,R)$ be the set of languages described by Theorem 7.

We now note the result follows since if $X_1 \sqcup_T X_2 = R$ has a solution, it also has a solution in $\mathcal{S}(T,R) \times \mathcal{T}(T,R)$, since $\sqcup_T$ is monotone. Thus, we simply test all the finite (non-trivial) pairs in $\mathcal{S}(T,R) \times \mathcal{T}(T,R)$ for the desired equality. □

Theorem 8 was known for catenation, $T = 0^*1^*$ (see, e.g., Kari and Thierrin [11]). However, it also holds for, e.g., the following operations: insertion ($0^*1^*0^*$), k-insertion ($0^*1^*0^{\leq k}$ for fixed $k \geq 0$), and bi-catenation ($1^*0^* + 0^*1^*$).

In a forthcoming paper, we show that if T is 1-thin (i.e., only contains only one trajectory of each length) then we can decide if a regular language R has a non-trivial decomposition along T. With this result and Theorem 8, we still note that there exist operations for which we have not solved the shuffle decomposition problem. To our knowledge, the question of the decidability of the existence of solutions to $R = X_1 \sqcup_T X_2$ for a given regular language R is still open in the following cases:

(a) arbitrary shuffle: $T = (0 + 1)^*$;
(b) literal shuffle: $T = (0^* + 1^*)(01)^*(0^* + 1^*)$;
(c) initial literal shuffle: $T = (01)^*(0^* + 1^*)$.

3.1 Solving Quadratic Equations

Let $T \subseteq \{0,1\}^*$ be a letter-bounded regular set of trajectories. We can also consider solutions X to the equation $X \sqcup_T X = R$, for regular languages R. This is a generalization of a result due to Kari and Thierrin [10].

Theorem 9. *Fix a letter-bounded regular set of trajectories T. Then it is decidable whether there exists a solution X to the equation $X \sqcup_T X = R$ for a given regular language R.*

Proof. Let $\mathcal{S}(T, R)$ be the set of languages described by Theorem 6, and, analogously, let $\mathcal{T}(T, R)$ be the set of languages described by Theorem 7.

Assume the equation $X \sqcup_T X = R$ has a solution. Then we claim that it also has a regular solution. Let X be a language such that $X \sqcup_T X = R$. Then, in particular, X is a solution to the equation $X \sqcup_T Y = R$, where X is fixed and Y is a variable. Thus, by Theorem 6, there is some regular language $Y_i \in \mathcal{S}(T, R)$ such that $X \sqcup_T Y_i = R$. Further, $X \subseteq Y_i$. Analogously, considering the equation $X \sqcup_T Y_i = R$, $X \subseteq Z_j$ for some regular language $Z_j \in \mathcal{T}(T, R)$. Thus, $X \subseteq Y_i \cap Z_j$, and $Z_j \sqcup_T Y_i = R$.

Let $X_0 = Y_i \cap Z_j$. Then note that $R = X \sqcup_T X \subseteq X_0 \sqcup_T X_0 \subseteq Z_j \sqcup_T Y_i = R$. The inclusion follows by the monotonicity of $\sqcup_T$. Thus, $X_0 \sqcup_T X_0 = R$. By construction, X_0 is regular.

Thus, to decide whether there exists X such that $X \sqcup_T X = R$, we construct the set

$$\mathcal{U}(T, R) = \{Y_i \cap Z_j \ : \ Y_i \in \mathcal{S}(T, R), Z_j \in \mathcal{T}(T, R)\},$$

and test each language for equality. If a solution exists, we answer yes. Otherwise, we answer no. □

4 Existence of Trajectories

In this section, we consider the following problem: given languages L_1, L_2 and R, does there exist a set of trajectories T such that $L_1 \sqcup_T L_2 = R$? We prove this to be decidable when L_1, L_2, R are regular languages.

Theorem 10. *Let $L_1, L_2, R \subseteq \Sigma^*$ be regular languages. Then it is decidable whether there exists a set $T \subseteq \{0,1\}^*$ of trajectories such that $L_1 \sqcup_T L_2 = R$.*

Proof. Let

$$T_0 = \{t \in \{0,1\}^* \ : \ \forall x \in L_1, y \in L_2, x \sqcup_t y \subseteq R\}. \tag{1}$$

Note that the following is an equivalent definition of T_0:

$$T_0 = \{t \in \{0,1\}^* \ : \ \forall x \in L_1 \cap \Sigma^{|t|_0}, y \in L_2 \cap \Sigma^{|t|_1}, (x \sqcup_t y \subseteq R)\}. \tag{2}$$

Then we claim that

$$\exists T \subseteq \{0,1\}^* \text{ such that } (L_1 \sqcup_T L_2 = R) \iff L_1 \sqcup_{T_0} L_2 = R.$$

The right-to-left implication is trivial. Assume that there is some $T \subseteq \{0,1\}^*$ such that $L_1 \sqcup_T L_2 = R$. Let $t \in T$. Then for all $x \in L_1$ and $y \in L_2$, $x \sqcup_t y \subseteq L_1 \sqcup_T L_2 = R$. Thus, $t \in T_0$ by definition, and $T_0 \supseteq T$.

Thus, note that $R = L_1 \sqcup_T L_2 \subseteq L_1 \sqcup_{T_0} L_2$. The fact that $L_1 \sqcup_{T_0} L_2 \subseteq R$ is clear from the definition of T_0. Thus $L_1 \sqcup_{T_0} L_2 = R$ and the claim holds.

We now establish that T_0 is regular and effectively constructible; to do this, we instead show that $\overline{T_0} = \{0,1\}^* - T_0$ is regular.

Let $M_j = (Q_j, \Sigma, \delta_j, q_j, F_j)$ be a complete DFA accepting L_j for $j = 1, 2$. Let $M_r = (Q_r, \Sigma, \delta_r, q_r, F_r)$ be a complete DFA accepting R. Define an NFA $M = (Q, \{0,1\}, \delta, q_0, F)$ where $Q = Q_1 \times Q_2 \times Q_r$, $q_0 = [q_1, q_2, q_r]$, $F = F_1 \times F_2 \times (Q_r - F_r)$, and δ is defined as follows:

$$\delta([q_j, q_k, q_\ell], 0) = \{[\delta_1(q_j, a), q_k, \delta_r(q_\ell, a)] \ : \ a \in \Sigma\} \ \forall [q_j, q_k, q_\ell] \in Q_1 \times Q_2 \times Q_r,$$
$$\delta([q_j, q_k, q_\ell], 1) = \{[q_j, \delta_2(q_k, a), \delta_r(q_\ell, a)] \ : \ a \in \Sigma\} \ \forall [q_j, q_k, q_\ell] \in Q_1 \times Q_2 \times Q_r.$$

Then we note that δ has the following property: for all $t \in \{0,1\}^*$,

$$\delta([q_1, q_2, q_r], t) = \{[\delta(q_1, x), \delta(q_2, y), \delta(q_r, x \sqcup_t y)] : x, y \in \Sigma^*, |x| = |t|_0, |y| = |t|_1\}.$$

By (2), if $t \in \overline{T_0}$ there is some $x, y \in \Sigma^*$ such that $x \in L_1$, $y \in L_2$, $|x| = |t|_0$, $|y| = |t|_1$ but $x \sqcup_t y \cap \overline{R} \neq \emptyset$. This is exactly what is reflected by the choice of F. Thus, $L(M) = \overline{T_0}$.

Thus, as T_0 is effectively regular, to determine whether there exists T such that $L_1 \sqcup_T L_2 = R$, we construct T_0 and test $L_1 \sqcup_{T_0} L_2 = R$. □

Note that the proof of Theorem 10 is similar in theme to the proofs of, e.g., Kari [7, Thm. 4.2, Thm. 4.6]: we construct a maximal solution to an equation, and that solution is regular. The maximal solution is tested as a possible solution to the equation to determine if any solutions exist. However, unlike the results of Kari, Theorem 10 does not use the concept of an inverse operation.

We can also repeat Theorem 10 for the case of deletion along trajectories. The results are identical, with the proof following by the substitution of $T_0 = \{t \in \{i,d\}^* \ : \ \forall x \in L_1, y \in L_2, x \rightsquigarrow_t y \subseteq R\}$. The proof that T_0 is regular differs slightly from that above; we leave the construction to the reader. Thus, we have the following result:

Theorem 11. *Let $L_1, L_2, R \subseteq \Sigma^*$ be regular languages. Then it is decidable whether there exists a set $T \subseteq \{i,d\}^*$ of trajectories such that $L_1 \rightsquigarrow_T L_2 = R$.*

5 Undecidability Results

We now demonstrate some undecidability results relating to equations involving shuffle on trajectories.

5.1 Undecidability of One-Variable Equations

Recall that a set $T \subseteq \{0,1\}^*$ is said to be *complete* if $\alpha \sqcup_T \beta \neq \emptyset$ for all $\alpha, \beta \in \Sigma^*$. Say that a set $T \subseteq \{0,1\}^*$ of trajectories if *left-preserving* (resp., *right-preserving*) if $T \supseteq 0^*$ (resp., $T \supseteq 1^*$). Note that if T is complete, then it is both left- and right-preserving.

Let $\Pi_0, \Pi_1 : \{0,1\}^* \to \{0,1\}^*$ be the projections given by $\Pi_0(0) = 0, \Pi_0(1) = \epsilon$ and $\Pi_1(1) = 1, \Pi_1(0) = \epsilon$. We say that $T \subseteq \{0,1\}^*$ is *left-enabling* (resp., *right-enabling*) if $\Pi_0(T) = 0^*$ (resp., $\Pi_1(T) = 1^*$).

Define the right-useful solutions to $L \sqcup_T X = R$ as

$$use_T^{(r)}(X; L) = \{x \in X \ : \ L \sqcup_T x \neq \emptyset\}. \tag{3}$$

The left-useful solutions, denoted $use_T^{(\ell)}(X; L)$, axe defined similarly for the equation $X \sqcup_T L = R$.

In this section, we examine undecidability of the existence of solutions of equations involving context-free languages. Namely, we show that:

Theorem 12. *Fix $T \subseteq \{0,1\}^*$ to be a regular set of left-enabling (resp., right-enabling) trajectories. For a given CFL L and regular language R, it is undecidable whether or not $L \sqcup_T X = R$ (resp., $X \sqcup_T L = R$) has a solution X.*

Proof. Let T be left-enabling. Let Σ be an alphabet of size at least two and let $\#, \$ \notin \Sigma$. Let $R = (\Sigma^+ + \#^+) \sqcup_T \$^*$. By the closure properties of $\sqcup_T$, and the fact that T is regular, R is a regular language. Let $L \subseteq \Sigma^+$ be an arbitrary CFL and $L_\# = L + \#^+$. We claim that

$$L_\# \sqcup_T X = R \text{ has a solution } \iff L = \Sigma^+. \tag{4}$$

This will establish the result, since it is undecidable whether an arbitrary CFL $L \subseteq \Sigma^+$ satisfies $L = \Sigma^+$.

First, if $L = \Sigma^+$, then note that $X = \$^*$ is a solution for (4). Second, assume that X is a solution for (4). It is clear that for all X,

$$L_\# \sqcup_T X = R \iff L_\# \sqcup_T use_T^{(r)}(X; L_\#) = R. \tag{5}$$

Thus, we will focus on useful solutions to the equation $L_\# \sqcup_T X = R$.

Now, we note that, assuming that $use_T^{(r)}(X, L_\#)$ is a solution to (4), words in $use_T^{(r)}(X, L_\#)$ cannot contain letters from Σ, because words in R do not contain words with both $\#$ and letters from Σ. In particular, let $x \in use_T^{(r)}(X, L_\#)$. Then there exists a $y \in L_\#$ (in particular, $y \neq \epsilon$) such that $y \sqcup_T x \neq \emptyset$. Consider the word $\#^{|y|}$. As y and $\#^{|y|}$ have the same length, we must have that $\#^{|y|} \sqcup_T x \neq \emptyset$.

Consider any $z \in \#^{|y|} \sqcup_T x$. As $|y| \neq 0$, $|z|_\# > 0$. As $L_\# \sqcup_T X = R$, we must have that $z \in (\Sigma^+ + \#^+) \sqcup_T \$^*$. Thus, $z \in (\# + \$)^+$, and consequently, $x \in (\# + \$)^*$. Thus, $use_T^{(r)}(X, L_\#) \subseteq (\# + \$)^*$.

Let $\Pi_\Sigma : (\Sigma \cup \{\#, \$\})^* \to \Sigma^*$ be the projection onto Σ. Now as T is left-enabling, note that $\Pi_\Sigma(R) = \Sigma^+$, by definition of $R = (\Sigma^+ + \#^+) \sqcup_T \$^*$. Thus,

$$\begin{aligned}
\Sigma^+ &= \Pi_\Sigma(R) = \Pi_\Sigma(L_\# \sqcup_T X) \\
&= \Pi_\Sigma(L_\# \sqcup_T use_T^{(r)}(X, L_\#)) \subseteq \Pi_\Sigma(L_\# \sqcup_T (\# + \$)^*) \\
&= \Pi_\Sigma((L + \#^+) \sqcup_T (\# + \$)^*) = \Pi_\Sigma((L \sqcup_T (\# + \$)^*) + (\#^+ \sqcup_T (\# + \$)^*)) \\
&= \Pi_\Sigma(L \sqcup_T (\# + \$)^*) \\
&= L \subseteq \Sigma^+.
\end{aligned}$$

The last equality is valid since T is left-enabling. Thus, for all $x \in L$, there is some $j \geq 0$ such that $x \sqcup_T \#^j \neq \emptyset$. We conclude that $L = \Sigma^+$, and thus, by (4), the result follows.

The proof in the case that T is right-enabling is similar. □

We can give an incomparable result which removes the condition that T must be regular, but must strengthen the conditions on words in T. Namely, T must be left-preserving rather than left-enabling:

Theorem 13. *Fix $T \subseteq \{0,1\}^*$ to be a set of left-preserving (resp., right-preserving) trajectories. Given a CFL L and a regular language R, it is undecidable whether there exists a language X such that $L \sqcup_T X = R$ (resp., $X \sqcup_T L = R$).*

The proof is fairly similar to the proof of Theorem 12, and is omitted due to length requirements. We also note that if R and L are interchanged in the equations of the statements of Theorem 13 or Theorem 12, the corresponding problems are trivially undecidable.

5.2 Undecidability of Shuffle Decompositions

It has been shown [1] that it is undecidable whether a context-free language has a nontrivial shuffle decomposition with respect to the set of trajectories {0,1}*. Here we extend this result for arbitrary complete regular sets trajectories.

If T is a complete set of trajectories, then any language L has decompositions $L \sqcup_T \{\epsilon\}$ and $\{\epsilon\} \sqcup_T L$. Below we exclude these trivial decompositions; all other decompositions of L are said to be nontrivial.

Theorem 14. *Let T be any fixed complete regular set of trajectories. For a given context-free language L it is undecidable whether or not there exist languages $X_1, X_2 \neq \{\epsilon\}$ such that $L = X_1 \sqcup_T X_2$.*

In the proof of Theorem 14, whenever the CFL has a nontrivial decomposition along the set of trajectories T, it has a decomposition where the component languages are, in fact, regular. This gives the following:

Corollary 2. *Let T be any fixed complete regular set of trajectories. For a given context-free language L it is undecidable whether or not*

(a) there exist regular languages $X_1, X_2 \neq \{\epsilon\}$ such that $L = X_1 \sqcup_T X_2$.
(b) there exist context-free languages $X_1, X_2 \neq \{\epsilon\}$ such that $L = X_1 \sqcup_T X_2$.

5.3 Undecidability of Existence of Trajectories

We complement the decidability results of Section 4 with the following undecidability results concerning LCFLs:

Theorem 15. *Given an LCFL L and regular languages R_1, R_2, it is undecidable whether there exists $T \subseteq \{0,1\}^*$ such that (a) $R_1 \sqcup_T L = R_2$ or (b) $L \sqcup_T R_1 = R_2$.*

Proof. We first establish (b). We claim that

$$\exists\, T \subseteq \{0,1\}^* \text{ such that } L \sqcup_T \{\epsilon\} = \Sigma^* \iff L = \Sigma^*.$$

If $L = \Sigma^*$, then $T = 0^*$ satisfies the equation. Assume that there exists T such that $L \sqcup_T \{\epsilon\} = \Sigma^*$. Then for all $x \in \Sigma^*$, there exists $y \in L$ and $t \in T$ such that $x \in y \sqcup_t \epsilon$. But this only happens if $x = y$ and $t = 0^{|x|}$. Thus, $x \in L$. Therefore, $L = \Sigma^*$. This establishes part (b) of the lemma.

Let $sym : \{0,1\}^* \to \{0,1\}^*$ be given by $sym(0) = 1$ and $sym(1) = 0$. Part (a) follows on noting that $T \subseteq \{0,1\}^*$ satisfies $L \sqcup_T \{\epsilon\} = \Sigma^*$ iff $sym(T)$ satisfies $\{\epsilon\} \sqcup_{sym(T)} L = \Sigma^*$. □

For a set $I \subseteq \mathbb{N}$, let $\Sigma^I = \{x \in \Sigma^* : |x| \in I\}$. The following undecidability result holds:

Lemma 1. *Given an LCFL L, it is undecidable whether there exists $I \subseteq \mathbb{N}$ such that $L = \Sigma^I$.*

Lemma 1 can be established by appealing to a meta-theorem of Hunt and Rosenkrantz [5]. We note that a similar undecidability result was independently proven by Kari and Sosík [9, Lemma 8.1] (using a reduction from PCP) for proving undecidability results of the following form: given $R_1, R_2 \in \text{REG}$ and $L \in \text{CF}$, does $L \sqcup_T R_1 = R_2$ hold? Necessary and sufficient conditions on regular T such that the above problem is decidable are given by Kari and Sosík. Lemma 1 allows us to prove the following result:

Lemma 2. *Given an LCFL L and regular languages R_1, R_2, it is undecidable whether there exists $T \subseteq \{0,1\}^*$ such that $R_1 \sqcup_T R_2 = L$.*

Proof. Let L be an LCFL. Then we claim that

$$\exists T \subseteq \{0,1\}^* \text{ such that } L = \Sigma^* \sqcup_T \{\epsilon\} \iff \exists I \subseteq \mathbb{N} \text{ such that } L = \Sigma^I.$$

($\Leftarrow$): If $I \subseteq \mathbb{N}$ is such that $L = \Sigma^I$, then let $T = \{0\}^I$. Then we can easily see that $L = \Sigma^* \sqcup_T \{\epsilon\}$.

($\Rightarrow$): If $T \subseteq \{0,1\}^*$ exists such that $L = \Sigma^* \sqcup_T \{\epsilon\}$, let $I = \{i : 0^i \in T\}$. Then we can see that $L = \Sigma^I$.

Therefore, since it is undecidable whether $L = \Sigma^I$, we have established the result. □

We now turn to deletion along trajectories.

Theorem 16. *Given an LCFL L and regular languages R_1, R_2, it is undecidable whether there exists $T \subseteq \{i,d\}^*$ such that (a) $R_1 \rightsquigarrow_T R_2 = L$, (b) $L \rightsquigarrow_T R_1 = R_2$, or (c) $R_1 \rightsquigarrow_T L = R_2$.*

References

1. Câmpeanu, C., Salomaa, K., Vágvölgyi, S.: Shuffle decompositions of regular languages. Int. J. Found. Comp. Sci. **13** (2002) 799–816
2. Daley, M., Ibarra, O., Kari, L.: Closure and decidability properties of some language classes with respect to ciliate bio-operations. Theor. Comp. Sci. **306** (2003) 19–38
3. Domaratzki, M.: Deletion along trajectories. Theor. Comp. Sci. **320** (2004) 293–313
4. Ginsburg, S., Spanier, E.: Bounded regular sets. Proc. Amer. Math. Soc. **17** (1966) 1043–1049
5. Hunt, H., Rosenkrantz, D.: Computational parallels between the regular and context-free languages. SIAM J. Comput. **7** (1978) 99–114
6. Ito, M.: Shuffle decomposition of regular languages. J. Univ. Comp. Sci. **8** (2002) 257–259
7. Kari, L.: On language equations with invertible operations. Theor. Comp. Sci. **132** (1994) 129–150
8. Kari, L.: Deletion operations: Closure properties. Intl. J. Comp. Math. **52** (1994) 23–42
9. Kari, L., Sosík, P.: Language deletions on trajectories. Technical Report 606, Computer Science Department, University of Western Ontario (2003). Submitted for publication
10. Kari, L., Thierrin, G.: k-catenation and applications: k-prefix codes. J. Inf. Opt. Sci. **16** (1995) 263–276
11. Kari, L., Thierrin, G.: Maximal and minimal solutions to language equations. J. Comp. Sys. Sci. **53** (1996) 487–496
12. Leiss, E.: Language Equations. Monographs in Computer Science. Springer (1999)
13. Mateescu, A., Rozenberg, G., Salomaa, A.: Shuffle on trajectories: Syntactic constraints. Theor. Comp. Sci. **197** (1998) 1–56
14. Meduna, A.: Middle quotients of linear languages. Intl. J. Comp. Math. **71** (1999) 319–335
15. Salomaa, A., Yu, S.: On the decomposition of finite languages. In Rozenberg, G., Thomas, W., eds.: Developments in Language Theory. (1999) 22–31
16. Yu, S.: Regular languages. In Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages, Vol. I. Springer-Verlag (1997) 41–110

Efficient View Point Selection for Silhouettes of Convex Polyhedra*

Therese Biedl, Masud Hasan**, and Alejandro López-Ortiz

School of Computer Science, University of Waterloo,
Waterloo, Ontario N2L 3G1, Canada.
{biedl,m2hasan,alopez-o}@uwaterloo.ca

Abstract. The silhouette of polyhedra is an important primitive in application areas such as machine vision and computer graphics. In this paper, we study how to select view points of convex polyhedra such that the silhouette satisfies certain properties. Specifically, we give algorithms to find all projections of a convex polyhedron such that a given set of edges, faces and/or vertices appear on the silhouette.
We present an algorithm to solve this problem in $O(k^2)$ time for k edges. For orthogonal projections, we give an improved algorithm that is fully adaptive in the number l of connected components formed by the edges, and has a time complexity of $O(k \log k + kl)$. We then generalize this algorithm to edges and/or faces appearing on the silhouette.

1 Introduction

Polyhedra are solids in 3-dimensional space. When looking at a polyhedron from a view point, the eye or camera computes a 2-dimensional projection of the polyhedron which may be an orthogonal or a perspective projection, depending on whether the view point is at infinity or not. In particular, some features of the polyhedron, such as vertices, edges or faces, are visible, while others are hidden in this projection. Especially noticeable are those features that reside on the *shadow boundary* of the projection, i.e., those that are just barely visible. Closely related is the concept of the *silhouette,* which are those edges for which exactly one incident face is visible; the two concepts describe the same set for convex polyhedra.

Silhouettes are useful in various settings, especially in the area of **machine vision.** For 3D gauging systems, in order to gauge a given part, video cameras are used to acquire silhouettes of the part along with the locations of the lighting element. These silhouettes help identify key elements of the part [16]. For assembling purposes, silhouettes are used to compute the boundary and the orientation of the mechanical parts to be picked up by a robot for assembly [15]. Silhouettes are also used for quality control [24], object recognition [17,23], illuminating critical features [18] and others.

* Research supported by NSERC.

** On leave from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 735–747, 2004.
© Springer-Verlag Berlin Heidelberg 2004

For **image recognition**, researchers consider the topological graph representation of projections of polyhedra, which are called a *characteristic view* [19]. A similar topological graph is also considered for the silhouette [10]. Silhouettes from the projection of an object can be matched against stored pre-computed characteristic views hence aiding in recognition of the object (see [19] and the references therein). This has the advantage that the views from two nearby view points likely result in the same characteristic view, which makes the system robust under small positional errors. The characteristic views of a polyhedron are better known as *aspect graphs.* See [22] for a detailed survey on aspect graphs. **Reconstruction from images** concerns the problem of approximately reconstructing a 3D object from one or more images [13,14,25]. Instead of full images of an object, often only silhouettes are used in a process called *volume intersection* [13,14]. In **computer graphics**, silhouette edges represent discontinuities in the visibility of an object, and are one of the strongest visual cues of the shape of an object [12]. When rendering the object, it often suffices to render only the edges on the silhouette, which can result in substantial speedup [20].

The computation of silhouettes has been studied extensively both in the computational geometry and in the computer graphics community. Pop et al. [20] gave an algorithm for perspective projections that maintains the silhouette of polyhedra during arbitrary changes of the view point. They use duality theory (see also Section 3.2) to develop a practical and efficient heuristic to maintain the corresponding visibility properties. Efrat et al. [10] presented several combinatorial bounds on the silhouette structure of a collection of convex polyhedra when the view point moves along a straight line or along an algebraic curve. They compute the silhouette map which is the arrangement of the silhouettes of all objects with their hidden parts removed. Their combinatorial complexity is the bound on the number of combinatorial changes in the silhouette map during the motion of the view point. For orthogonal projections only, Benichoe and Elber [4] give output-sensitive algorithms to find silhouettes from polyhedral scenes for a given view point. By mapping all projection directions onto the surface of a sphere and then mapping the sphere onto the surface of a cube, they reduce this problem to a segment intersection problem in 2D. Using known techniques from [1,8], they find the silhouette in time linear in the size of the output.

Our results. In general the field has concentrated in computing the silhouette efficiently, or reconstruction and/or recognition of a polyhedron from a given set of silhouettes. In contrast to this, in this paper we do not consider the view point as given or fixed, instead we ask the question how to choose it suitably. Thus we consider the problem of given a polyhedron and given some desired property of the silhouette, how easy is it to find one or all projections that have the property?

This question is motivated by numerous applications of the silhouettes mentioned before. Two applications specifically benefit from the ability to bring certain features such as edges or faces to the silhouette. In **quality control** of a manufacturing process such as casting, we can check for flaws such as air pockets by examining whether each edge is a smooth and continuous line. This can be

done efficiently if edges appear on the silhouette using video cameras to acquire the silhouette of the part. In **visualization,** crucial features should be forced to the silhouette to make them easily detectable. Also, if features are to be labeled it is advantageous to move them to the silhouette, since the outside area allows for space to place labels.

In this paper we consider how to select a projection of a polyhedron such that the silhouette satisfies certain properties. A straightforward approach to doing so is to compute all possible projections (depending on the conditions imposed on the silhouette, there is usually only a finite number of connected regions of view points that have these properties). Brunet et al. considered the case of computing the viewpoints from which the projections of a given polygonal chain projects a convex shadow [6]. They apply this special case to compute efficiently the occlusion properties in a 3-D scene.

This paper addresses the following question: Given a set of edges $\mathcal{E}$, a set of faces $\mathcal{F}$ and a set of vertices $\mathcal{V}$ of a convex polyhedron, how can we efficiently find all projections such that all elements in $\mathcal{E}, \mathcal{F}$, and/or $\mathcal{V}$ are on the silhouette under a perspective or orthogonal projection?

The straightforward algorithm for this problem would have a runtime of $O(k^3)$ for k edges (see Section 2 for a more detailed discussion). We show in Section 3 that the time can be improved to $O(k^2)$ by using geometric duality and transversal theory. In Section 4, we develop an adaptive algorithm in terms of the number of connected paths l formed by the edges, with time complexity $O(k \log k + kl)$. We conclude in Section 5 with open problems.

2 Preliminaries

A *convex polyhedron* is the intersection of finitely many half-spaces. We will not consider non-convex polyhedra in this paper and hence occasionally drop "convex" from now on. A *face /edge /vertex* of the convex polyhedron is the maximal connected set of points which belong to exactly one/exactly two/at least three planes that support these half-spaces. Every polyhedron with n vertices has $\theta(n)$ edges and $\theta(n)$ faces. Throughout this paper, we assume that the given polyhedron is fully-dimensional and that the origin o is inside the polyhedron. A *perspective projection* is defined relative to a point p, whereas an *orthogonal projection* is defined relative to a point at infinity, i.e., a direction d_p. A face f of a convex polyhedron with supporting plane π is *visible* with respect to a view point p (possibly at infinity) if the normal vector of f has a positive inner product with the vector from the origin to p (which is the direction vector d_p for the orthogonal projection). The *projection* of a polyhedron with respect to a view point p is the set of all those faces visible from p. An edge or vertex is in the projection if and only if at least one incident face is in the projection.

The *silhouette* of a projection from view point p is the set of all those edges for which one incident face is in the projection and the other face is not. In particular, note that we do not consider an edge to be on the silhouette if its projection is the degenerate case of a single point. The *shadow boundary* is the

set of edges of the silhouette that are incident to the unbounded region. For a convex polyhedron, the notion of silhouette and shadow boundary is identical.

In the following, we will study how to find all view points for which a given set of edges is in the silhouette. Note that the set of such view points is in general not connected. We let a *viewing region* be a maximal connected region for which all points in it are view points with the desired property. Note that since projections in which edges project to points are considered degenerate and hence not to be proper view points, the viewing regions are always open sets.

First we study when exactly is an edge on the silhouette. Assume edge e is incident to faces f_1 and f_2, which are defined by half-spaces h_1 and h_2 with supporting planes π_1 and π_2. For $i = 1, 2$, face f_i is visible from view point p if and only if p is not in half-space h_i (recall that the origin is inside the polyhedron and hence belongs to all half-spaces of all faces). Edge e is on the silhouette if and only if exactly one of its incident faces is visible from p, so p must be in exactly one of the half-spaces h_1 and h_2. Thus, p belongs to $(h_1 \cap \overline{h_2}) \cup (h_2 \cap \overline{h_1})$ (or more precisely, to the maximal open set contained within this set), which is a double-wedge formed by planes π_1 and π_2.

3 Perspective Projections

In this section, we study how to find efficiently all view points of a convex polyhedron such that a given set of edges, vertices and/or faces is on the silhouette under perspective projections. Our results rely heavily on duality theory and transversal theory, which we review in Section 3.2.

3.1 Computing Perspective Projections from Plane Arrangements

A straightforward approach to find all view points from which a set $\mathcal{E}$ of edges is on the silhouette is to compute the intersection of all the double-wedges associated with the edges in $\mathcal{E}$. In general the intersection of k double-wedges may have as many as $\theta(k^3)$ connected components under perspective projections. Therefore, finding all projections with certain properties can be done in $O(k^3 T)$ time, where T is the time to check whether a projection from a given view region has the desired property. In what follows we use duality theory to improve on this.

3.2 Geometric Duality and Transversal Theory

Given a point $p = (a, b, c)$, the dual of the point is a plane $dual(p) = \{(x, y, z) : ax + by + cz = 1\}$. For a plane $\pi = \{(x, y, z) : ax + by + cz = d\}$, the dual is a point $dual(\pi) = (a/d, b/d, c/d)$. If π passes through the origin, then $dual(\pi)$ is at infinity. Recall that a point p lies in plane π if and only if $dual(\pi)$ lies in $dual(p)$. One can also easily prove the following observation:

Lemma 1. *Let π be a plane that does not contain the origin o, and let p be a point that is not the origin o. Then π intersects the line segment $[o, p]$ if and only if $dual(p)$ intersects the line segment $[o, dual(\pi)]$.*

For an edge e, the dual is defined as the line segment $[dual(\pi_1), dual(\pi_2)]$, where π_1 and π_2 are the planes supporting the two incident faces of e. Pop et al. [20] made the following crucial observation.

Lemma 2. [20] *An edge e of a convex polyhedron is on the silhouette from view point p if and only if $dual(p)$ intersects $dual(e)$.*

A *geometric transversal* is an affine subspace of $\mathbb{R}^d$, such as a point, line, plane, or hyper-plane, that intersects every member of a family of convex sets. The set of [point, line, plane, or hyper-plane] transversals of a family forms a topological space in $\mathbb{R}^d$. Geometric transversal theory concerns the complexity and efficient computation of this topological space, especially in 2D and 3D. We will use the following result:

Theorem 1. ([9], see also [11], Theorem 5.6) *Let $\mathcal{A}$ be a family of n compact convex polytopes in 3D with a total of n vertices. All plane transversals of $\mathcal{A}$ can be found in $O(n^2\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackerman function. If $\mathcal{A}$ consists of n line segments, then all plane traversals can be found in $O(n^2)$ time.*

We combine duality with transversal theory. Interestingly enough, the theorem above in turn uses dual geometric space (thus returning to the primal space) and analyzes double-wedges; it would thus be possible to express our algorithm directly in terms of double-wedges by tracing the results from transversal theory. We will not do this here for brevity and clarity's sake.

3.3 View Point Selection Algorithm

Lemma 2 characterizes when an edge is on the silhouette of a projection. Combining this with transversal theory gives an algorithm to find all projections with a given set of k edges in $O(k^2)$ time. We apply the same approach to obtain an algorithm for given sets of vertices and faces. We first need to clarify what it means for a vertex or a face to be on the silhouette. This is relatively straightforward for a vertex, which is on the silhouette if and only if two incident edges are on the silhouette. The notion of a face being on the silhouette is not entirely obvious, since the silhouette by nature consists of line segments, so the entire face cannot be on it. However, for the purpose of displaying the face "near" the silhouette, the following definition seems appropriate: A face f is considered to be on the silhouette from view point p if and only if f is visible from p and at least one edge of f is on the silhouette.

As in Lemma 2, we characterize when a vertex or a face is on the silhouette. Assume that v is a vertex, and let $f_1, \ldots, f_l$ be the faces, in circular order, adjacent to the vertex v. In dual space, the dual of v is a plane and the duals of the planes supporting $f_1, \ldots, f_l$ are points in $dual(v)$. So the dual points of the planes of $f_1, \ldots, f_l$ are co-planar and thus form a polygon, which we call the *dual polygon* associated with vertex v. Observe that this dual polygon is convex as we assume that the origin is inside the polyhedron.

Lemma 3. *A vertex v is on the silhouette from view point p if and only if its associated dual polygon is intersected by $dual(p)$.*

Proof. The polygon associated with v consists exactly of the union of the dual of the edges incident to v. If $dual(p)$ intersects this polygon, then it intersects exactly two edges incident to v. These two edges are then on the silhouette, and in consequence, v is also on the silhouette. □

For a face f, let $f_1, f_2, ..., f_l$ be the faces adjacent to f and let $\pi, \pi_1, \ldots, \pi_l$ be the planes that support $f, f_1, \ldots, f_l$. Define the *dual polyhedron* of face f to be the convex hull of $dual(\pi), dual(\pi_1), \ldots, dual(\pi_l)$ (see Figure 1).

Lemma 4. *A face f with supporting plane π of a convex polyhedron is on the silhouette from view point p if and only if $dual(p)$ intersects both the line segment $[o, dual(\pi)]$ and the dual polyhedron associated with f.*

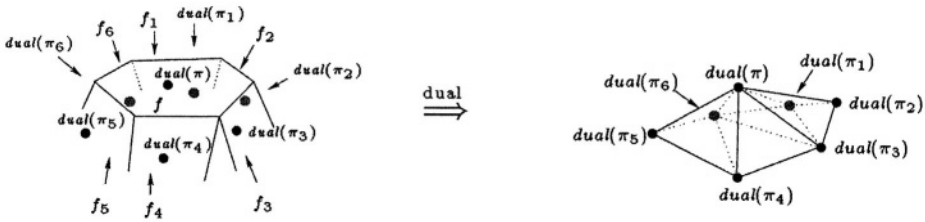


Fig. 1. The dual polyhedron associated with a face.

Proof. Let P' be the dual polyhedron of f. This polyhedron has one vertex v_f for f, which is incident to all other vertices. Now, if $dual(p)$ intersects P', then it separates the vertices of P' into two groups, and in particular intersects some of the edges incident to v_f, since all vertices are incident to v_f. Therefore, some of the edges of f are on the silhouette if and only if $dual(p)$ intersects P'.

For f itself to be on the silhouette, we additionally need that f is visible (i.e., not occluded) from view point p. This holds if and only if the plane through f separates the origin o (which is inside the polyhedron) from p. By Lemma 1, this holds if and only if $dual(p)$ intersects the line segment $[o, dual(\pi)]$. □

Using the above lemmas, in combination with transversal theory, we can now compute all projections that have a given set of features on the silhouette.

Theorem 2. *Let P be a convex polyhedron with n vertices, and let $\mathcal{E}, \mathcal{V}$, and $\mathcal{F}$ be sets of edges, vertices and faces, respectively. All view points from which all edges in $\mathcal{E}$, all vertices in $\mathcal{V}$, and all faces in $\mathcal{F}$ are on the silhouette under perspective projections can be found in $O(n^2\alpha(n))$ time. The time reduces to $O(|\mathcal{E}|^2)$ if only edges are specified.*

Proof. Compute the dual of the edges in $\mathcal{E}$, the dual polygons associated with vertices in $\mathcal{V}$ and the dual polyhedra associated with faces in $\mathcal{F}$. Also, for each face $f \in \mathcal{F}$, compute the line segment $[o, dual(\pi)]$, where π is the plane supporting f. Now find all plane transversals that intersect these convex objects. By the above lemmas this gives exactly the perspective projections for which the features are on the silhouette.

Every edge creates one line segment to be transversed; by Theorem 1 we can find the projections for a set of edges in $O(|\mathcal{E}|^2)$ time. For a vertex v, the associated polygon has $degree(v)$ many vertices, and for a face f, the associated polyhedron has $degree(f) + 1$ many vertices. For a set of vertices and faces, the total size of the associated polygons and polyhedra can be at most the number of edges and faces in P, which is $O(n)$. Hence, by Theorem 1 we can find all projections for a set of edges, vertices and faces in $O(n^2\alpha(n))$ time. □

4 Orthogonal Projections

In this section, we show how to compute efficiently all orthogonal projections such that a given set of edges is on the silhouette. This can be done with the same approach as in Theorem 2 (i.e., using duality theory and transversal theory.) However, a much simpler approach also works. Since the location of the origin is irrelevant in an orthogonal projection, we can identify directly the wedge for each edge and translate it such that all wedges intersect in one point. Then the hyperplane arrangement defined by the k wedges has only $O(k^2)$ cells, and we can thus find all projections in $O(k^2)$ time.

We improve on this further to give an algorithm that is adaptive in the number of paths formed by the set of edges. We study the case of one path in Section 4.1 and the case of many paths in Section 4.2. Finally, the ability to search all $O(k^2)$ cells of the arrangement allows us more flexibility in choosing projections; we study in Section 4.3 how to choose projections such that certain edges are *not* on the silhouette.

4.1 One Connected Path

If a set of edges forms a path in the polyhedron, then it is easier to compute viewing regions for which they are on the silhouette, mostly because (as we will show) there are only two viewing regions. This is not the case for arbitrary edges. For example Figure 2 shows a polyhedron where we have at least four viewing regions with edges e_1, e_2, e_3 on the silhouette. Two views from two different viewing regions are shown in (a) and (c); the other two viewing regions are origin-symmetric to the ones illustrated here. We now show that there are only two regions from which the path can be realized[1].

Theorem 3. *Given a path of k edges on a convex polyhedron P, there are only two viewing regions from which all k edges of the path are in the silhouette of P, and we can find them in $O(k \log k)$ time.*

[1] This theorem is implicitly assumed without proof in [6].

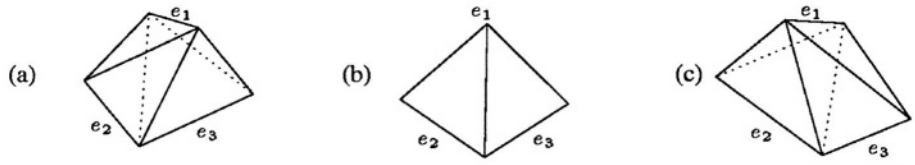


Fig. 2. The polyhedron has four viewing regions for edges e_1, e_2, e_3. (a) One incident face of e_1 is visible. (b) A rotation such that e_1 reduces to a point; this view point is on the boundary between two viewing regions. (c) The other incident face of e_1 is visible.

Proof. Let the path consist of edges $e_1, \ldots, e_k$. For $1 \leq i \leq k$, let f_i be the face to the left of edge e_i, where "to the left" is taken with respect to walking along the path from e_1 to e_k. Let f_i' be the other face incident to e_i. The crucial observation is that if the path is on the silhouette, then we either see all of $f_1, \ldots, f_k$ or all of $f_1', \ldots, f_k'$. To prove this, let v be the common vertex of e_1 and e_2. The clockwise order of faces around v is then f_1, (possibly) some other faces, f_2, f_2', (possibly) some other faces, f_1' (see Figure 3).

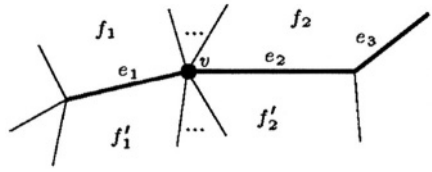


Fig. 3. The order of faces around a vertex.

Since the visible incident faces of v are connected, and for each of $\{f_1, f_1'\}$ and $\{f_2, f_2'\}$ exactly one is invisible, there are only two possibilities: either all of $f_1, \ldots, f_2$ are visible (and the others are invisible), or all of $f_2', \ldots, f_1'$ are visible (and the others are invisible). So either $\{f_1, f_2\}$ are visible or $\{f_1', f_2'\}$ are visible, but it is not possible (for example) that f_1 and f_2' are visible. Assume f_1 and f_2 are visible. Repeating the argument for e_2 and e_3, this shows that f_3 must also be visible, and so on, so $f_1, \ldots, f_k$ are all visible (and $f_1', \ldots, f_k'$ are invisible). Alternatively, if f_1' and f_2' are visible, then all of $f_1', \ldots, f_k'$ are visible (and $f_1, \ldots, f_k$ are invisible).

Recall that a face f is visible if and only if the view point is not in the half-space that defined the face. Thus, the view points from which $f_1, \ldots, f_k$ are all visible and $f_1', \ldots, f_k'$ are all invisible are defined as the intersection of $2k$ half-spaces. This defines one viewing region. A second viewing region is the one from which $f_1', \ldots, f_k'$ are all visible and $f_1, \ldots, f_k$ are invisible. This region is again the intersection of half-spaces —the opposite half-spaces as those for the first viewing region. The half-spaces can be found in $O(k)$ time, and their intersection can be computed in $O(k \log k)$ time (see e.g. [21]). There are no other viewing regions. Thus, all viewing regions can be computed in $O(k \log k)$ time. □

Note that the above theorem applies to both orthogonal and perspective projections, and it is hence possible to find all viewpoints for which a path P with k edges is on the silhouette in $O(k \log k)$ time for both orthogonal and perspective projections. In contrast, the results in the next subsections apply only to orthogonal projections.

4.2 Multiple Paths

In this section, we show how to use the above results to improve the time complexity in the case when k edges are not all disjoint. Assume that k edges are in l connected components which we refer to as $\mathcal{P}_1, \ldots, \mathcal{P}_l$; obviously $l \leq k$. From Theorem 3 we know how to compute all projections from which $\mathcal{P}_i$ is on the silhouette. We now show that for orthogonal projections, we can intersect these viewing regions in $O(k \log k + kl)$ time, which is an improvement over the $O(k^2)$ result of Section 3 if $l = o(k)$. Observe that for orthogonal projections, the set of view points from which e is on the silhouette is translation-invariant, since the view points are at infinity and correspond to directions. Hence, we are allowed to translate the double-wedge arbitrarily, and in particular we may assume that the intersection of the two planes contains the origin.

Theorem 4. *Given l disjoint paths of a convex polyhedron P, we can find all orthogonal projections of P such that all k edges of the paths are on the silhouette in $O(k \log k + kl)$ time.*

Proof. From Theorem 3 we know that for $1 \leq i \leq l$ there are exactly two viewing regions from which $\mathcal{P}_i$ is on the silhouette. Since we are considering orthogonal projections, these two viewing regions are two disjoint convex cones, say C_i^+ and C_i^-, and after a suitable translation we may assume that the apex of each cone is at the origin. Let $C_i = C_i^+ \cup C_i^-$ be the *double-cone* for path $\mathcal{P}_i$, and set $C^* = \bigcap C_i$; the desired viewing regions are then exactly the connected components of C^*.

In general, l double-cones may have $\theta(l^3)$ connected components in their intersection. For double-cones with origin at the apex this reduces to $O(l^2)$, but computing all connected components of C^* directly is still too slow. We therefore consider a projection of the double-cones onto a 2D surface (see [4,5]). Consider a unit cube D centered at the origin. To compute C^*, it suffices to compute the intersection of C^* with each face of D. We explain how to do this in the following for one face f of D only.

For any i, where $1 \leq i \leq l$, both C_i^+ and C_i^- can intersect f, but these intersections are disjoint (see Figure 4). Set $B_i = C_i \cap f$; then B_i is a single convex polygon or the union of two convex polygons. We call each B_i a *cone polygon*. Let $B^* = \bigcap B_i$, then each connected component of B^* corresponds to one viewing region.

To compute B^*, we compute the arrangement $\mathcal{A}$ of the cone polygons $B_1, \ldots, B_l$, which has at most $2l$ convex polygons with a total of at most $2k$ edges. This can be done in $O(k \log k + I)$ time where I is the number of intersecting points [3,7]. Within the same time bound, we can also compute the

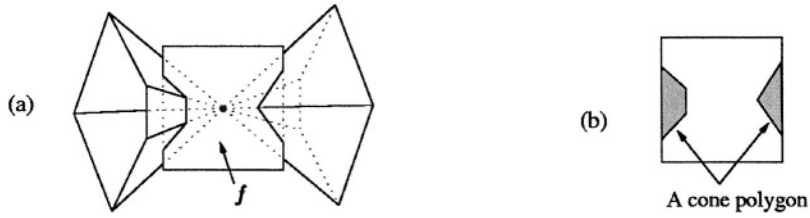


Fig. 4. (a) Intersection of a double-cone C_i with a face f. (b) The corresponding cone polygon B_i has two disjoint components.

planar graph G defined by this arrangement (see Figure 5(a)). G has $O(k+I)$ vertices, edges and faces. Now we want to find all cells in the arrangement $\mathcal{A}$ that belong to all cone polygons. To do so, we compute a *modified directed dual graph* G' of G by computing the dual graph of G and replacing each edge by a directed 2-cycle (see Figure 5(b)). Note that here we use the term "dual" in the graph theoretical sense, which is distinct from the geometric duality used before.

Each vertex in G' is a cell in $\mathcal{A}$ and each directed edge e in G' corresponds to entering or leaving a polygon of $B_1, \ldots, B_l$. We store with each edge of G' whether traversing this edge means entering or leaving a cone polygon. Finding all cells for which we are inside all l cone polygons can then be done by traversing the graph G' in such a way that all vertices are visited (e.g. with a DFS-traversal) and maintaining a counter of the number of cone polygons that the currently visited vertex is in. Since G' has $O(k+I)$ vertices and edges, this can be done in $O(k+I)$ time. The time complexity of our algorithm hence is $O(k \log k + I)$. To find an upper bound on I, observe that there are $O(l)$ convex polygons of $O(k)$ edges total. Each edge can intersect each convex polygon at most twice, so $I \in O(kl)$ (and examples can be found where this is tight [2]). So the run-time of our algorithm is $O(k \log k + kl)$. □

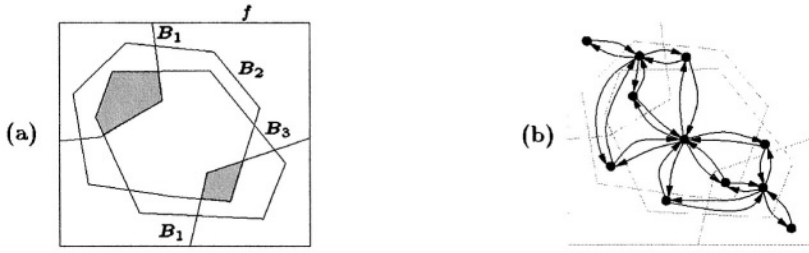


Fig. 5. (a) The arrangement $\mathcal{A}$; the desired viewing regions are shown shaded. (b) The corresponding modified dual graph. Some edges and vertices have been omitted for clarity's sake.

4.3 Hiding Edges

Another application of view point selection is industrial design, where we might wish to make certain features easily visible or prominent, while some other features (such as service trap doors or unsightly wiring) should be hidden. Thus we would like to force edges, vertices or faces *not* to be on the silhouette; we say that these features are *hidden from the silhouette.* Note that a face/vertex is hidden from the silhouette if and only if all its incident edges are hidden from the silhouette, so it suffices to explain how to hide edges.

Lemma 5. *The set of all view points from which a set of k edges is not visible under an orthogonal projection can be computed in $O(k^2)$ time.*

Proof. Each double-wedge from which an edge is visible also defines, by its complement, a double-wedge from which the edge is *not* visible. Since we are considering orthogonal projections, we can translate all double-wedges such that all hyperplanes that define them intersect the origin. Therefore, the hyperplane arrangement now has only $O(k^2)$ cells and can be computed in $O(k^2)$ time. By using a traversal technique similar to the one in Section 4.2, we can check whether there is any cell in which all edges are hidden in $O(k^2)$ time. □

Note that we can at the same time force some edges into the silhouette and hide some other edges from the silhouette. With the same proof as in Lemma 5, this can be done in $O(k^2)$ time. However, hiding edges unfortunately cannot easily be made adaptive in the number of paths that these edges form. The main obstacle is that Theorem 3 ("there are only two viewing regions") does not necessarily hold for hiding edges in one path.

5 Conclusions

In this paper, we studied the question of how to find all view points of a polyhedron that force a given set of edges onto the silhouette of a convex polyhedron, and gave an efficient algorithm to do so, as well as an adaptive algorithm in terms of the number of connected paths formed by the edges for orthogonal projections. A number of open questions remain:

- Transversal theory allowed us to force not only edges, but also vertices and faces, onto the silhouette but at a higher cost since the size of the objects to be transversed is proportional to the degree of the vertex or face involved. Is it possible to improve on this, say find all projections with k vertices or k faces on the silhouette in $O(k^2)$ or even $O(k)$ time?
- The general adaptive algorithm (Section 4.2) applies only to orthogonal projections. Does a similar result hold for the perspective projections?
- Hiding edges in a perspective projection can effectively be phrased as a transversal problem again, but Theorem 1 cannot be applied, since the resulting shapes are not convex and compact. So how can we efficiently find all view points from which a given set of edges is hidden?

- Given a family of edge sets $\mathcal{E}_1, \ldots, \mathcal{E}_s$, how quickly can we find all projections such that for each set $\mathcal{E}_i$, at least one (but not necessarily all) of the edges in $\mathcal{E}_i$ are on the silhouette?
- Rather than specifying the edges that must be on the silhouette, we might be interested in the number of such edges. So given a number k, how can we find all projections such that the silhouette has exactly k edges?

Last but not least, all our results were for convex polyhedra. What are efficient algorithms for the same problems for non-convex polyhedra?

References

1. P. K. Agarwal and M. Sharir. Applications of a new space partitioning technique. *Disc. Comp. Geom.,* 9:11–38, 1993.
2. B. Aronov and M. Sharir. The common exterior of convex polygons in the plane. *Comp. Geom.,* 8:139–149, 1997.
3. I. J. Balaben. An optimal algorithm for finding segments intersections. *ACM Symp. Comp. Geom.,* pages 211–219, 1995.
4. F. Benichou and G. Elber. Output sensitive extraction of silhouettes from polygonal geometry. In *7th Pac. Conf. Comp. Grap. Appl.,* pages 60–69, 1999.
5. P. Bose, F. Gomez, P. Ramos, and G. T. Toussaint. Drawing nice projections of objects in space. *J. Vis. Comm. Image Represent.,* 10:155–172, 1999.
6. P. Brunet, I. Navazo, J. Rossignac, and C. Saona-Vazquez. Hoops: 3d curves as conservative occluders for cell-visibility. *Comp. Grap. Forum,* 20(3):431–442, 2001.
7. B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM,* 39:1–54, 1992.
8. B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Disc. Comp. Geom.,* 8:407–429, 1992.
9. H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Disc. Comp. Geom.,* 4:311–336, 1989.
10. A. Efrat, L. Guibas, O. Hall-Holt, and L. Zhang. On incremental rendering of silhouette maps of a polyhedral scene. In *ACM-SIAM Symp. Disc. Alg.,* pages 910–917, 2000.
11. J. E. Goodman, R. Pollack, and R. Wenger. Geometric transversals theory. In *New Trends in Discrete and Computational Geometry.* Springer-Verlag, 1993.
12. J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception,* 13:321–330, 1984.
13. A. Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object? *Comp. Vis. Image Under.,* 67(1):81–87, 1997.
14. A. Laurentini. Introducing a new problem: Shape-from-silhouette when the relative positions of the viewpoints is unknown. *IEEE Pat. Ana. Mach. Int.,* 25(11):1484–1493, 2003.
15. Melles Griot Corporation. *Machine Vision Guide,* 2003. http://www.mellesgriot.com/products/machinevision/lif_3.htm.
16. J. Miller. Low-cost in-process machine vision gauging system. Technical report, Dept. Elec. Comp. Eng., Univ. Michigan-Dearborn, April 1998. http://www.engin.umd.umich.edu/ceep/reports/96-97/jmiller.html.
17. F. Mokhtarian. Silhouette-based occluded object recognition through curvature scale space. *J. Mach. Vis. Appl.,* 10(3):87–97, 1997.

18. J. A. Muratore. Illumination for machine vision. http://www.pinnaclevision.co.uk/illum02.htm.
19. H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *Intl. J. Comp. Vis.,* 5(2):137–160, 1990.
20. M. Pop, G. Barequet, C. A. Duncan, M. T. Goodrich, W. Huang, and S. Kumar. Efficient perspective-accurate silhouette computation and applications. In *ACM Symp. Comp. Geom.,* pages 60–68, 2001.
21. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1995.
22. R. D. Schiffenbauer. *A Survey of Aspect Graph.* PhD thesis, Dept. Comp. Inf. Sci., Polytechnic Univ., New York, 2001.
23. Siemens. *Outline inspection with SIMATIC VS 110.* Product literature. http://www.ad.siemens.de/dipdata/mk/pdf/e20001-a60-p285-x-7600.pdf.
24. SIGHTech Vision Systems. *Eyebot Application, Inspecting Hard Disk Media.* Product literature. http://www.sightech.com/hard_disk_app_note.pdf.
25. K. Sugihara. *Machine Interpretation of Line Drawing.* MIT Press, 1986.

Angles and Lengths in Reconfigurations of Polygons and Polyhedra*

Therese Biedl, Anna Lubiw, and Michael J. Spriggs

School of Computer Science, University of Waterloo,
Waterloo, ON N2L 3G1, Canada,
{biedl,alubiw,mjspriggs}@uwaterloo.ca

Abstract. We explore the possibility of reconfiguring, or "morphing", one simple polygon to another, maintaining simplicity, and preserving some properties of angles and edge lengths. In linkage reconfiguration, edge lengths must be preserved. By contrast, a *monotone morph* preserves edge directions and changes edge lengths monotonically. A monotone morph is only possible for *parallel* polygons—ones with corresponding edges parallel. Our main results are that monotone morphs exist for parallel pairs of polygons that are: (1) convex; or (2) orthogonally convex. Our morphs either move vertices in straight lines, or change few edge lengths at once. On the negative side, we show that it is NP-hard to decide if two simple parallel orthogonal polygons have a monotone morph. We also establish which of these results extend to 3-dimensional polyhedra.

1 Introduction

In transforming one geometric object to another we often wish to preserve certain geometric properties. For example, it is possible to continuously reconfigure one planar triangulation to another combinatorially identical planar triangulation while maintaining simplicity [5,13,7]. More generally, one can "morph" between different straight line drawings of the same embedded graph while maintaining simplicity. There are unfortunately no known bounds on how edge lengths and angles might change during these morphs, and in fact the method of [13] explicitly shrinks edges to infinitesimals.

Studies have been done on transforming configurations while restricting the amount of change to angles and/or edge lengths. The most stringent requirement on edge lengths is that they not change at all, i.e., the source and target configuration have the same edge lengths. This is the problem of linkage reconfiguration, which can be done (while maintaining simplicity) only for a limited class of graphs: between any two simple chains in the plane with corresponding edges of the same length, there is a transformation that preserves simplicity and edge lengths [6,12]. The result extends to cycles (or equivalently, polygons) but not to trees [2].

* Research partially supported by NSERC.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 748–759, 2004.
© Springer-Verlag Berlin Heidelberg 2004

This paper is about preserving simplicity and *angles*. We say that two combinatorially identical configurations of straight line segments and vertices in the plane are *parallel* if corresponding edges are parallel. Guibas, Hershberger, and Suri [10], and independently, Grenander et al. [9], prove that between any two parallel simple polygons in the plane there is a transformation that preserves simplicity and angles. Such a transformation—preserving simplicity and angles—will be called a *parallel morph*. Thomassen had proved earlier [13] that orthogonal polygons have parallel morphs. In papers related to this one we establish other classes of graphs that have parallel morphs: chains and trees drawn in any dimension [3], and orthogonal graphs drawn in the plane [4].

Preserving angles OR edge lengths should not mean ignoring the other measure. Linkage results preserve edge lengths and change angles nicely: for chains, the angles open to π, then close to their target values [6]; for convex polygons the angles change monotonically [1]. The situation for parallel morphs is not so good: the parallel morph between polygons given by Guibas et al. shrinks edges to infinitesimals. They morph via an intermediate "reduced form" polygon in which the longest and shortest edges have a length ratio that is a product of $\Theta(n)$ infinitesimals, n the size of the polygon. They note that even if each infinitesimal could be replaced by a constant, the length ratio would still be exponential in n. The algorithms in the other papers share this defect.

Our goal in this paper is to find parallel morphs that alter edge lengths as little as possible. The strongest possibility is a *monotone* morph—a parallel morph in which each edge changes length monotonically from its initial to its final value.

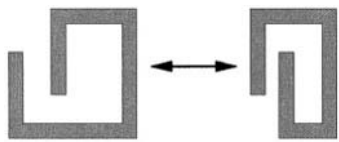


Fig. 1. Two polygons that do not admit a monotone morph.

Monotone morphs are often not possible—see Figure 1—and we show in Section 4 that it is NP-hard to decide if two parallel polygons have a monotone morph, even for the special case of orthogonal polygons, where there are only two possible edge directions.

As main results, we prove that monotone morphs do exist in two cases: (1) between any two parallel convex polygons (Section 3); and (2) between any two parallel orthogonally convex polygons (Section 5). Result (1) is easily obtained via the *linear* morph which moves all vertices at once at uniform speed in a straight line from source to target. The linear morph does not work in case (2), and, in any case, it is not always desirable to move everything at once. For both (1) and (2) we give "discrete" morphs that change few edge lengths at once—3 for the convex case, and 2 for the orthogonally convex case.

Result (1) is analogous—by exchanging the roles of angles and edge lengths—to the result [1] that there is a discrete morph between any two convex polygons with the same edge lengths that preserves edge lengths and changes angles monotonically. (Here "discrete" means changing few angles at once.) We note that these two results cannot be unified: it is not always possible to morph between a pair of convex polygons where both angles and edge-lengths change, but all angles and edge-lengths change monotonically.

Finally, we go up to 3 dimensions, and consider monotone morphs of parallel polyhedra. The linear morph still works for convex polyhedra, but discrete monotone morphs do not always exist. Result (2) does not extend either: we give an example of a pair of parallel orthogonally convex polyhedra with no monotone morph.

At the beginning of this section we mentioned the relationship of our work to the general topic of morphing. Further description of this background can be found in the full version of the paper. Our work is also related to "parallel redrawings" [14,11] which are like parallel morphs except that simplicity need not be maintained. For more information see the full version of the paper.

2 Definitions

We define a *configuration* as a set of points in d-dimensional space and straight line segments between some pairs of those points. Two such configurations P, Q are *parallel* if there is an isomorphism between them with corresponding edges parallel, and endpoints in the same order with respect to the coordinate axes. If P and Q are parallel configurations then a connected set of edges that are co-planar in P will also be co-planar in Q. Thus, in particular, if P and Q are the (connected) edge sets of parallel 3-dimensional polyhedra then corresponding faces are parallel as well.

A *morph* is a transformation, or "metamorphosis" of one configuration to another. More technically, for a source configuration P and a target configuration Q, a *morph* from P to Q is a continuously changing family of configurations, $R(t), 0 \leq t \leq 1$ with $R(0) = P$ and $R(1) = Q$. A *parallel morph* between two simple parallel configurations is one where $R(t)$ is simple, and parallel to $R(0)$, for all t, $0 \leq t \leq 1$. A *monotone* morph is a parallel morph in which edge lengths change monotonically.

For an edge e, let $e(P)$ and $e(Q)$ denote the corresponding parallel edges in P and Q. We say that edge e is a *+-edge* of P if $||e(P)|| < ||e(Q)||$—i.e., if the length of e needs to increase during a monotone morph from P to Q. We define *–-edges* analogously. Any +-edge of P corresponds to a –-edge of Q and vice versa. We say that e is a 0-edge if it has the same length in P and Q—i.e., its length must stay unchanged during a monotone morph from P to Q.

If the parallel configurations form a simple cycle in the plane, we are thus morphing parallel polygons. An *orthogonal polygon* is a polygon such that each edge is horizontal or vertical. An orthogonal polygon is *orthogonally convex* if every horizontal and every vertical line intersects the interior of the polygon in

a (possibly empty) connected line segment. An *orthogonal polyhedron* is a polyhedron such that every face is perpendicular to one of the three coordinate axes. An orthogonal polyhedron is *orthogonally convex* if every plane perpendicular to a coordinate axis intersects the polyhedron in a (possibly empty) connected, orthogonally convex polygon.

3 Morphing Convex Polyhedra

In this section we show that, in any dimension, any two parallel convex polyhedra admit a monotone morph. Our first morph changes the lengths of all non-0-edges simultaneously. In the second subsection we explore discrete-step morphing, where only a small number of edges change length in each step.

3.1 Linear Morphing

Let P and Q be parallel configurations consisting of points and straight line segments between some pairs of those points. Suppose that P and Q are parallel configurations with corresponding points $p_1, \ldots p_n$ and $q_1, \ldots, q_n$ respectively. The *linear morph* between P and Q is given by the configurations $R(t), 0 \leq t \leq 1$ with vertices $r_1(t), \ldots, r_n(t)$ defined by $r_i(t) = (1-t)p_i + tq_i$, and with edges corresponding to those of P (and Q).

From the definition, one easily verifies the following result:

Lemma 1. *For any two parallel configurations the linear morph is a monotone morph, except that it does not necessarily maintain simplicity.*

Though linear morphs do not always preserve simplicity, they do so for convex polyhedra.

Theorem 1. *For any two parallel convex polyhedra the linear morph is a monotone morph.*

Proof. Since P and Q are both convex and the morph is parallel, all intermediate polyhedra are also convex and hence simple.

3.2 Discrete-Step Morphing

The linear morph described in Section 3.1 changes all (non-static) edge lengths simultaneously. Below, we consider monotone morphing where only a small number of edges change in length at one time.

Discrete-step morphing in 2D

Theorem 2. *For any two parallel convex polygons P and Q with n vertices, there exists a monotone morph from P to Q that consists of at most n steps, each of which changes the lengths of at most three edges.*

Proof. Let P and Q be the source and target polygons, where the vertices of P are $p_0, \ldots, p_{n-1}, p_n = p_0$ and the corresponding vertices of Q are $q_0, \ldots, q_{n-1}, q_n = q_0$. For $i = 1, \ldots, n$, define $v_i = (q_i - q_{i-1}) - (p_i - p_{i-1})$. We morph by picking a small set of non-zero vectors v_i that have the 0-vector as a convex combination, and altering the lengths of the corresponding edges until one becomes a 0-edge. After n such steps all edges must be 0-edges. We begin by describing what to do with such a small set of vectors, and later argue existence.

Claim. If three non-0 vectors v_i, v_j, v_k have the 0-vector as a convex combination, then there exists a monotone morph that changes the lengths of only those three edges and converts at least one of them into a 0-edge.

Proof. (Sketch) Assume that $i < j < k$ and that $\lambda_i v_i + \lambda_j v_j + \lambda_k v_k = 0$, for some $\lambda_i, \lambda_j, \lambda_k \geq 0$ and $\lambda_i + \lambda_j + \lambda_k = 1$. For $t > 0$, define $P(t) = \langle p_1(t), \ldots, p_n(t) \rangle$ as follows (see also Figure 2):

1. $p_\ell(t) = p_\ell$ for $1 \leq \ell < i$.
2. $p_\ell(t) = p_\ell + t\lambda_i v_i$ for $i \leq \ell < j$.
3. $p_\ell(t) = p_\ell + t\lambda_i v_i + t\lambda_j v_j$ for $j \leq \ell < k$.
4. $p_\ell(t) = p_\ell + t\lambda_i v_i + t\lambda_j v_j + t\lambda_k v_k = p_\ell$ for $k \leq \ell \leq n$.

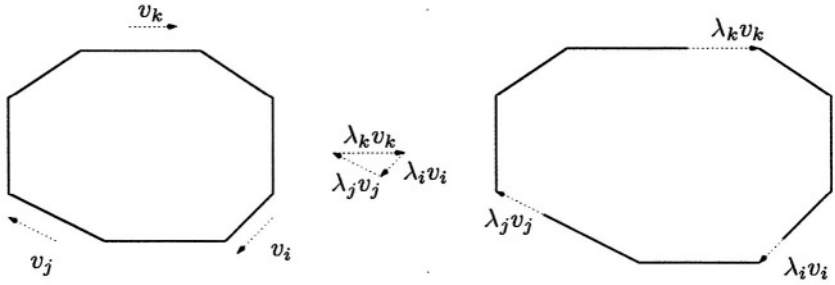


Fig. 2. Discrete-step monotone morphing for convex polygons. The target polygon Q is not shown.

One can verify that this is a monotone morph that keeps $P(t)$ parallel to P. Since the polygon was (and remains) convex, it remains simple, so we can increase t until one of the three changing edges becomes a 0-edge, which happens at $t = 1/\max\{\lambda_i, \lambda_j, \lambda_k\}$.

We note that a similar monotone morph exists if there are two non-0 vectors that have a convex combination to 0. It remains to show that one of these two situations occurs.

Claim. If the morph is not finished, i.e. not all edges are 0-edges, then there exist two or three non-0 vectors among $v_1, \ldots, v_n$ that have a convex combination to the 0-vector.

Proof. First observe that

$$\sum_{i=1}^{n} v_i = \sum_{i=1}^{n} (q_i - q_{i-1}) - (p_i - p_{i-1}) = \sum_{i=1}^{n} q_i - \sum_{i=0}^{n-1} q_i - \sum_{i=1}^{n} p_i + \sum_{i=0}^{n-1} p_i = 0,$$

since $q_0 = q_n$ and $p_0 = p_n$. Let $\{v'_1, \ldots, v'_m\} \subseteq \{v_1, \ldots, v_n\}$ be those vectors that are not the 0-vector; since P and Q are not identical there is at least one of them. Clearly $\sum_{i=1}^{m} v'_i = \sum_{i=1}^{n} v_i = 0$, since adding 0-vectors does not make a difference to the sum. Since the vectors v'_i have the 0-vector as a convex combination, thus, by Caratheodory's theorem (see e.g. [15]), there is a subset of at most three of these vectors that has the 0-vector as a convex combination.

Discrete-step morphing in 3D

Theorem 3. *A parallel morph of parallel convex polyhedra may need to change all edge lengths simultaneously.*

Proof. Let P be a polyhedron with all faces triangular, and let Q be a scaled-up version of P. Note that in any parallel morph of a triangle all three edge lengths must change simultaneously. Thus, by connectivity, any parallel morph from P to Q must change all edge lengths simultaneously.

4 The NP-Hardness of Monotone Morphing

Theorem 4. *Given parallel, orthogonal polygons P, Q (without holes) in the plane, it is NP-hard to decide whether P and Q admit a monotone morph.*

We only sketch the proof here, and give full details in the full version of the paper. The reduction is from 3-SAT, i.e., we are given n variables $x_1, \ldots, x_n$ and m clauses $c_1, \ldots, c_m$ that are conjunctions of three literals, and we want to know whether a satisfying boolean assignment exists.

We create two polygons P and Q that are almost identical except for a comb-like structure which has moved downward in Q. Also, some edges (marked with bold lines in Figure 3) are "elastic edges", which are capable of increasing and decreasing in length even in a monotone morph. (This is achieved by replacing each edge by two consecutive parallel edges.)

In order for the comb-like structure to move downward, the blocks that interleave with it have to move to the right temporarily. Each such block (or row of blocks) corresponds to a clause, and moving it to the right corresponds to satisfying the clause. A block can move to the right only if there is a gap in one of the blocks in its row (and elastic edges are set up such that any such gap can indeed be used to move the block to the right.) For each variable we have two columns of blocks. One column has flexibility to move up or down, and encodes whether the variable is set to true or false. (A surrounding frame prevents this column from moving too far.) The second column leaves gaps in each row depending on whether the variable or its negation is in the clause of this row. Thus, one can show that there is a monotone morph from P to Q if and only if the instance of 3-SAT is satisfiable.

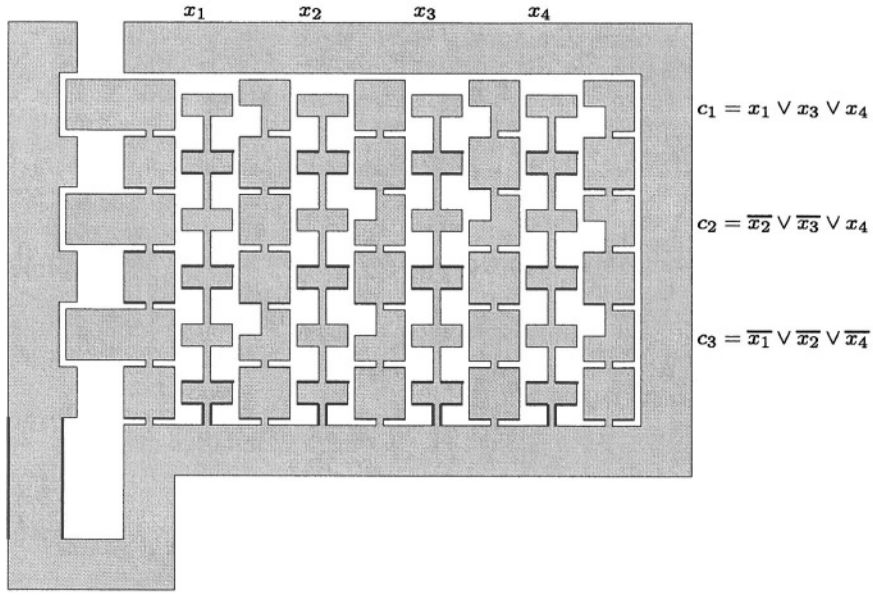


Fig. 3. The construction for the NP-hardness.

5 Orthogonally Convex Polygons and Polyhedra

In this section we show that there is a monotone morph between any two parallel orthogonally convex polygons, but that the analogous result for polyhedra fails.

5.1 Morphing Orthogonally Convex Polygons in 2D

Theorem 5. *For any two parallel orthogonally convex polygons P and Q there exists a monotone morph from P to Q. Furthermore, if P and Q have n vertices, there exists a morph that consists of at most n steps, each of which changes the lengths of two edges.*

Several times in this proof we use the technique of monotonically morphing P to an intermediate polygon P' and Q to an intermediate polygon Q' so that the changes in edge lengths match the labels, i.e., if e is a +-edge of P then $||e(P)|| \leq ||e(P')|| \leq ||e(Q')|| \leq ||e(Q)||$. This reduces the problem to that of finding a monotone morph from P' to Q', because such a morph, preceded by the morph from P to P', and followed by the reverse of the morph from Q to Q', provides a monotone morph from P to Q.

Applying this idea for the first time, we will morph the horizontal edges of P and Q, without changing the lengths of the vertical edges. Those are morphed later.

The edges of an orthogonally convex polygon can be partitioned into four *chains* by breaking the cycle of edges at the leftmost and rightmost vertical

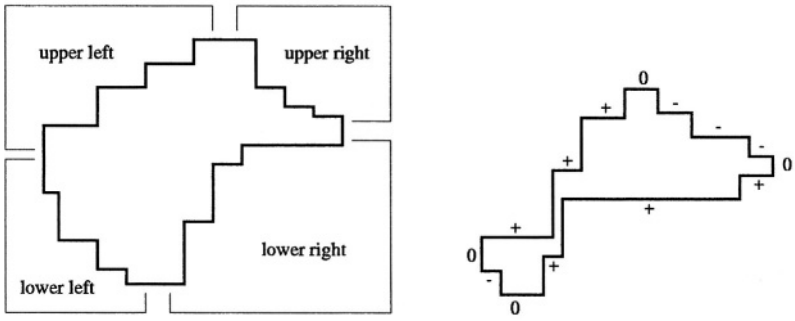


Fig. 4. The four chains, and a polygon in alternating form.

edges, and the uppermost and lowermost horizontal edges, as shown in Figure 4. We will refer to these chains as upper-left, upper-right, lower-left, and lower-right, in the obvious way. Each of the four extreme edges is considered to be part of two chains.

We morph to realize certain properties of these chains. In particular, a chain is *positive [negative]* if all horizontal edges in it are +-edges or 0-edges [—edges or 0-edges]; and a chain is *mixed* if it contains both +- and —edges. A polygon is in *alternating form* if its chains are alternately positive and negative. See Figure 4. Our monotone morph has the following steps: (1) morph to eliminate mixed chains; (2) morph to put each polygon in alternating form; (3) morph so that all horizontal edges are 0-edges. The morphs in steps (1) and (2) (see Figure 5) strictly increase the polygon (by containment) and are thus simple; the morphs in step (3) (see Figure 6) need a more careful justification of simplicity.

Lemma 2. *Given a pair of parallel orthogonally convex polygons P and Q, we can morph P and Q monotonically to arrive at two intermediate polygons P' and Q' with no mixed chains.*

Proof. Without loss of generality consider the horizontal edges of the upper-left chain of P and Q. Suppose the chain has two edges e_1 and e_2, in that order, with opposite sign. In one of P or Q, e_1 must be a —edge and e_2 a +-edge. Then the morph shown in Figure 5 is monotone and preserves simplicity, so it can be continued until either e_1 or e_2 becomes a 0-edge. Repeating this process yields the result.

Lemma 3. *Given a pair of parallel orthogonally convex polygons P and Q with no mixed chains, we can morph P and Q monotonically to arrive at two intermediate polygons P' and Q' in alternating form.*

Proof. Because we can morph in P or in Q (which has opposite signs) it suffices to show how to eliminate consecutive chains with +-edges. If both top chains of P contain +-edges then, because they are not mixed chains, the width of P must be less than the width of Q. In this case a lower chain of P must also

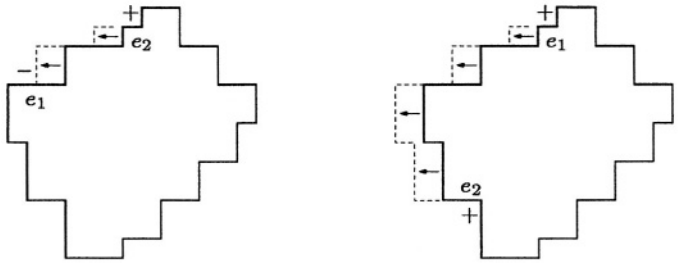


Fig. 5. Morphing to eliminate a mixed chain, and morphing to eliminate two positive left chains.

have a +-edge. Thus we reduce to the case where, say, the two left chains of P contain +-edges. Then the morph shown in Figure 5 is monotone and preserves simplicity, so it can be continued until one edge becomes a 0-edge. Repeating this process yields the result.

Lemma 4. *Given a pair of parallel orthogonally convex polygons P and Q in alternating form, we can morph P and Q monotonically to arrive at two intermediate polygons P' and Q' whose horizontal edges are all 0-edges.*

Proof. Let $l^+(P)$ be the horizontal length of P's left positive chain not counting the last (topmost) edge. Define $l^-(P)$, $l^+(Q)$, and $l^-(Q)$ for the other left chains analogously. Note that $l^+(P) \leq l^-(Q)$ and $l^+(Q) \leq l^-(P)$, since these are corresponding chains that must be morphed into each other.

Claim. If not all horizontal edges are 0-edges, then in at least one of P or Q we have $l^- > l^+$.

Proof. If $l^-(P) \leq l^+(P)$ and $l^-(Q) \leq l^+(Q)$ then

$$l^-(P) \leq l^+(P) \leq l^-(Q) \leq l^+(Q) \leq l^-(P)$$

so all are equal, and all left chains have only 0-edges. Since P and Q are in alternating form, this implies that the right chains have only 0-edges, too, and we are done.

Assume (by renaming if necessary) that $l^+(P) < l^-(P)$. Assume (by flipping upside down, if necessary) that the upper left chain of P is positive. The idea is to trade off a +-edge in P's upper left chain against either a —edge in P's upper right chain, or a +-edge in P's lower right chain. See Figure 6. This maintains simplicity for as long as $l^+(P) \leq l^-(P)$. We morph in such a way until $l^+(P') = l^-(P')$. Similarly we morph Q if necessary until $l^+(Q') = l^-(Q')$; then by the claim all horizontal edges are 0-edges.

This completes the description of the monotone morph between two parallel orthogonally convex polygons. One verifies that each step changes the lengths of

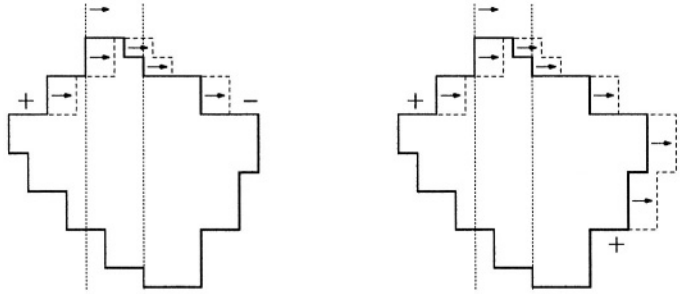


Fig. 6. Morphing to trade-off a +-edge in the upper left chain against an edge of the right chain.

at most two edges. It remains to bound the number of steps, which we show to be at most $n/2$ for the horizontal edges (and hence n in total). With every step of our morph, we either convert a horizontal edge into a 0-edge (this happens at most once per edge), or we achieve that $l^+(P) = l^-(P)$ or $l^+(Q) = l^-(Q)$ (each of these also happens at most once). The last motion must convert two edges into 0-edges, and hence accounts for the last two types of steps. So there are $n/2$ steps for horizontal edges and $n/2$ steps for vertical edges.

5.2 Morphing Orthogonally Convex Polyhedra in 3D

Moving into three dimensions, we find that not all pairs of orthogonally convex polyhedra admit a monotone morph. In order to show this, we find it convenient to first consider the problem of monotonically morphing between parallel orthogonal graphs embedded in the plane. See [4] for more on morphing graphs. It is not true that all such graphs admit a monotone morph.

Claim. The graphs shown in Figure 7 do not admit a monotone morph.

Proof. Let the leftmost graph in Figure 7 be the source graph, and let the graph on the right be the target. In these figures, letters denote edges of a particular length: $A = 35$, $B = 37$, $C = 7$. Each of the edges labeled C is actually composed of three incident, parallel edges of the graph.

Let us suppose that the source and target graphs admit a monotone morph. The central region of these graphs is shaded, serving to highlight that in the monotone morph, all edges bounding this region are static. The area contained in the shaded region is at all times greater than $|B|^2 = 37^2 = 1369$.

Now consider the edges that bound the outer face. The edges of length C do not change in length throughout the monotone morph. In a morph from source to target, there must be some time at which the two horizontal edges labeled C are aligned with respect to the horizontal axis. All of the other outer edges must have length $\leq A$. It is not difficult to show that the outer edges contains a region of area $\leq 4|A||C| + |C|^2 = 1029$. This is less than the area contained by

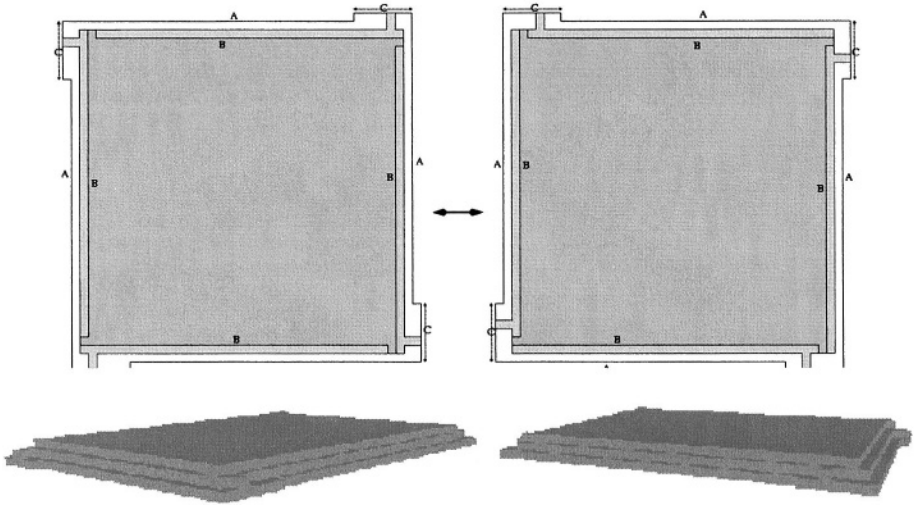


Fig. 7. Parallel, orthogonally convex polyhedra that do not admit a parallel morph.

the inner, shaded region. Therefore, by contradiction these graphs do not admit a monotone morph.

We use the graphs from Figure 7 to construct a pair of parallel, orthogonally convex polyhedra that do not admit a simple, monotone morph. Let x, y, z denote the axes of three-dimensional space. Generate two polyhedra such that the highest face (perpendicular the z axis) in both polyhedra is the same as the shaded region in the graphs. The other regions are transformed into faces of the polyhedra that are perpendicular with the z axis, at various heights. It is not difficult to see that such a pair of polyhedra can be constructed in which all faces are orthogonally convex polygons. See also Figure 7.

Theorem 6. *Not all pairs of orthogonally convex, parallel polyhedra admit a monotone morph, even if all faces are restricted to be orthogonally convex polygons.*

6 Conclusions

We have begun an exploration of parallel morphs that change edge lengths as little as possible, showing that monotone morphs are possible for convex and orthogonally convex polygons, but that, in general, deciding the existence of monotone morphs is NP-hard. We mention three open questions: (1) How badly must edge lengths change during parallel morphs between parallel polygons? (2) What other classes of configurations admit monotone morphs? (3) For what classes of configurations are there efficient algorithms to test for monotone morphs? We conjecture that this is possible for parallel orthogonal polygons that are monotone with respect to one axis (see Figure 1).

References

1. O. Aichholzer, E. D. Demaine, J. Erickson, F. Hurtado, M. Overmars, M. A. Soss, and G. T. Toussaint. Reconfiguring convex polygons. *Computational Geometry: Theory and Applications,* 20(1-2):85–95, 2001.
2. T. Biedl, E. Demaine, M. Demaine, S. Lazard, A. Lubiw, J. O'Rourke, S. Robbins, I. Streinu, G. Toussaint, and S. Whitesides. On reconfiguring tree linkages: Trees can lock. *Discrete Applied Mathematics,* 117:293–297, 2002.
3. T. Biedl, A. Lubiw, and M. Spriggs. Parallel morphing of trees and cycles. In *15th Canadian Conference on Computational Geometry (CCCG'03),* pages 29–32, 2003.
4. T. Biedl, A. Lubiw, and M. Spriggs. Parallel morphs of graphs. *In preparation,* 2004.
5. S. S. Cairns. Deformations of plane rectilinear complexes. *American Math. Monthly,* 51:247–252, 1944.
6. R. Connelly, E. D. Demaine, and G. Rote. Straightening polygonal arcs and convexifying polygonal cycles. *Discrete and Computational Geometry,* 30(2):205–239, 2003.
7. M. Floater and C. Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics,* 101:117–129, 1999.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.
9. U. Grenander, Y. Chow, and D. M. Keenan. *Hands: A Pattern Theoretic Study of Biological Shapes* (*Appendix D*). Springer-Verlag, 1991.
10. L. Guibas, J. Hershberger, and S. Suri. Morphing simple polygons. *Discrete and Computational Geometry,* 24(1):1–34, 2000.
11. B. Servatius and W. Whiteley. Constructing plane configurations in computer-aided design: combinatorics of directions and lengths. *SIAM J. Discrete Math,* 12, No. 1:136–153, 1999.
12. I. Streinu. A combinatorial approach to planar non-colliding robot arm motion planning. In *41st ACM Annual Symposium on Foundations of Computer Science (FOCS),* pages 443–453, November 2000.
13. C. Thomassen. Deformations of plane graphs. *Journal of Combinatorial Theory,* Series B: 34:244–257, 1983.
14. W.Whiteley. Parallel redrawing of configurations in 3-space. *preprint, Dept. of Mathematics and Statistics, York University, Ontario,* 1986.
15. G. Ziegler. *Lectures on Polytopes.* Springer-Verlag, 1995.

Improved Bounds and Schemes for the Declustering Problem*

Benjamin Doerr, Nils Hebbinghaus, and Sören Werth

Mathematisches Seminar, Bereich II, Christian-Albrechts-Universität zu Kiel
Christian-Albrechts-Platz 4, 24118 Kiel, Germany.
{bed,nhe,swe}@numerik.uni-kiel.de

Abstract. The declustering problem is to allocate given data on parallel working storage devices in such a manner that typical requests find their data evenly distributed among the devices. Using deep results from discrepancy theory, we improve previous work of several authors concerning rectangular queries of higher-dimensional data. For this problem, we give a declustering scheme with an additive error of $O_d(\log^{d-1} M)$ independent of the data size, where d is the dimension, M the number of storage devices and $d-1$ not larger than the smallest prime power in the canonical decomposition of M. Thus, in particular, our schemes work for arbitrary M in two and three dimensions, and arbitrary $M \geq d-1$ that is a power of two. These cases seem to be the most relevant in applications. For a lower bound, we show that a recent proof of a $\Omega_d(\log^{\frac{d-1}{2}} M)$ bound contains a critical error. Using an alternative approach, we establish this bound.

1 Introduction

The last decade saw dramatic improvements in computer processing speed and storage capacities. Nowadays, the bottleneck in data-intensive applications is disk I/O, the time needed to retrieve typically large amount of data from storage devices. One idea to overcome this obstacle is to spread the data on disks of multi-disk systems so that it can be retrieved in parallel. The data allocation is determined by so-called declustering schemes. Their aim is to allocate the data in such a manner that typical requests find their data evenly distributed on the disks.

A common example would be two dimensional geographic data. A typical request might ask for rectangular submap covering a particular region. The data blocks are associated with the tiles of a two dimensional grid and the queries are axis-parallel rectangles with borders along the grid, that request the data assigned to the tiles covered by the rectangle. The aim is to assign the tiles to the disks such that all disks have almost the same workload for all queries. A three dimensional application could regard the temperature distribution in a (human) body.

* Supported by the DFG-Graduiertenkolleg 357 "Effiziente Algorithmen und Mehrskalenmethoden".

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 760–771, 2004.
© Springer-Verlag Berlin Heidelberg 2004

We consider the problem of declustering uniform multi-dimensional data that is arranged in a multi-dimensional grid. There are many data-intensive applications that deal with this kind of data, especially multi-dimensional databases as remote-sensing databases [1]. A range query Q requests the data blocks that are associated with a hyper-rectangular subspace of the grid. We denote the number of requested blocks by $|Q|$. The response time of a query is the maximum number of blocks that are assigned to the same disk. In an ideal declustering scheme for a system with M disks, the response time of all disks for all queries Q would be exactly $|Q|/M$. The performance of a declustering scheme is measured by the worst case additive deviation from $|Q|/M$.

Declustering is an intensively studied problem and a lot of schemes with different approaches [2,3,4,5,6] have been developed in the last twenty years. It was an important turning point when discrepancy theory was connected to declustering.

Before the introduction of discrepancy in declustering, no known declustering scheme had theoretical performance bounds in arbitrary dimension d. Such bounds were only known for a few declustering schemes in two dimensions. The known results for these schemes considered only special cases, e. g., for the scheme proposed in [2] a proof for the average performance is given if the number M of disks is a Fibonacci number, and for the construction of the scheme in [4] M has to be a power of 2.

A breakthrough was marked by noting that the declustering problem is a discrepancy problem. Sinha, Bhatia and Chen [7] and Anstee, Demetrovics, Katona and Sali [8] developed declustering schemes for all M for two dimensional problems and proved their asymptotically optimal behavior via geometric discrepancy. The schemes of Sinha et al. [7] are based on two dimensional low discrepancy point sets. They also give generalizations to arbitrary dimension d, but without bounds on the error. Both papers show a lower bound of $\Omega(\log M)$ for the additive error of any declustering scheme in dimension two. The result of Anstee et al. [8] applies to latin square type colorings only, but their proof can easily be extended to the general case as well. Sinha et al. [7] claim that their proof technique yields a bound of $\Omega(\log^{\frac{d-1}{2}} M)$ for arbitrary $d \geq 3$, but their proof contains a crucial error (cf. Section 3).

The first non-trivial upper bounds for declustering schemes in arbitrary dimension were proposed by Chen and Cheng [9]. They present two schemes for the d–dimensional declustering problem. The first one has an additive error of $O(\log^{d-1} M)$, but works only if $M = p^k$ for some $k \in \mathbb{N}$ and p a prime such that $d \leq p$. The second works for arbitrary M, but the error increases with the size of the data.

Our Results: We work both on upper and lower bounds. For the upper bound, we present an improved scheme that yields an additive error of $O(\log^{d-1} M)$ for all values of M independent of the data size and all d such that $d \leq q_1 + 1$, where q_1 is the smallest factor in the canonical decomposition of M into prime powers. Thus, in particular, our schemes work for M being a power of two (such that $M \geq d - 1$) and for all M in dimension 2 and 3, which

is very useful from the viewpoint of application. We also show that the latin hypercube construction used by Chen and Cheng [9] is much better than proven there. Where they show that a latin hypercube coloring extended to the whole grid has an error of at most 2^d times the one of the latin hypercube, we show that both errors are the same.

For the lower bound, we present the first correct proof of the $\Omega(\log^{\frac{d-1}{2}} M)$ bound. Again, a more careful analysis shows that the positive discrepancy is at least $1/2d$ times the normal discrepancy instead of 3^{-d} as in [7].

2 Discrepancy Theory

In this section, we sketch the connection between the declustering problem and discrepancy theory. We start by noting that declustering is in fact a combinatorial discrepancy problem.

2.1 Combinatorial Discrepancy

Recall that the declustering problem is to assign data blocks from a multi-dimensional grid system to one of M storage devices in a balanced manner. The aim is that queries to a rectangular sub-grid use all storage devices in a similar amount. More precisely, our grid is $V = [n_1] \times \cdots \times [n_d]$ for some positive integers $n_1, \ldots, n_d$.[1] A query Q requests the data assigned to a sub-grid $[x_1..y_1] \times \cdots \times [x_d..y_d]$ for some integers $1 \leq x_i \leq y_i \leq n_i$. We assume that the time to process such a query is proportional to the maximum number of requested data blocks that are stored in a single device. If we represent the assignment of the data blocks to the devices through a mapping $\chi : V \to [M]$, then the time of the query above is $\max_{i \in [M]} |\chi^{-1}(i) \cap Q|$, where we identify the query Q with its associated sub-grid. Clearly, no declustering scheme can do better than $|Q|/M$. Hence a natural performance measure is the additive deviation from this lower bound.

This makes the problem a combinatorial discrepancy problem in M colors. Denote by $\mathcal{E}$ the set of all sub-grids in V. Then $\mathcal{H} = (V, \mathcal{E})$ is a hypergraph. For a coloring $\chi : V \to [M]$, the discrepancy of a hyperedge $E \in \mathcal{E}$ with respect to χ is

$$\operatorname{disc}(E, \chi) := \max_{i \in [M]} \left| |\chi^{-1}(i) \cap E| - \tfrac{1}{M}|E| \right|,$$

the discrepancy of $\mathcal{H}$ with respect to χ is

$$\operatorname{disc}(\mathcal{H}, \chi) := \max_{i \in [M], E \in \mathcal{E}} \left| |\chi^{-1}(i) \cap E| - \tfrac{1}{M}|E| \right|$$

and the discrepancy of $\mathcal{H}$ in M colors is

$$\operatorname{disc}(\mathcal{H}, M) := \min_{\chi : V \to [M]} \operatorname{disc}(\mathcal{H}, \chi).$$

[1] We use the notations $[n] := \{1, 2, \ldots, n\}$ and $[n..m] := \{n, \ldots, m\}$ for $n, m \in \mathbb{N}$, $n \leq m$.

These notions were introduced by Srivastav and the first author in [10,11] extending the well-known notion of combinatorial discrepancy to arbitrary numbers of colors. Similar notions concerning this problem were used by Biedl et al. [12] and Babai, Hayes and Kimmel [13]. For our purposes, only a positive deviation has to be regarded. We adapt the multi-color discrepancy notion in the obvious way:

$$\operatorname{disc}^+(\mathcal{H},\chi) := \max_{i\in[M],E\in\mathcal{E}} \left(|\chi^{-1}(i)\cap E| - \tfrac{1}{M}|E|\right)$$
$$\operatorname{disc}^+(\mathcal{H},M) := \min_{\chi:V\to[M]} \operatorname{disc}^+(\mathcal{H},\chi)$$

For many problems a distinction of these two concepts is not necessary as $\frac{1}{M-1}\operatorname{disc}(\mathcal{H}) \le \operatorname{disc}^+(\mathcal{H}) \le \operatorname{disc}(\mathcal{H})$ holds for all hypergraphs $\mathcal{H}$, and the influence of the number of colors is not known for many classes of hypergraphs. This is different for the declustering problem. Summarizing the above discussion, we have

Theorem 1 *The additive error of an optimal declustering scheme for the higher-dimensional interval query problem is* $\operatorname{disc}^+(\mathcal{H},M)$.

Since a central result of this paper is the fact that the discrepancy bounds are independent of the size of the grid, we usually work with the hypergraph $\mathcal{H}_N^d = ([N]^d,\mathcal{E}_N^d)$, $\mathcal{E}_N^d = \{\prod_{i=1}^d [x_i..y_i] \mid 1 \le x_i \le y_i \le N\}$ for some sufficiently large integer N. Furthermore, we regard only the case that $M \ge 3$. For the case $M = 2$, a multi-dimensional checkerboard coloring yields a declustering scheme with an additive error of 1/2. We prove the following result.

Theorem 2 *Let $M \ge 3$ and $d \ge 2$ be positive integers and q_1 the smallest prime power in the canonical factorization of M into prime powers. We have*

(i) $\operatorname{disc}^+(\mathcal{H}_N^d,M) = O(\log^{d-1} M)$ *for* $d \le q_1 + 1$, *independently of* $N \in \mathbb{N}$,
(ii) $\operatorname{disc}^+(\mathcal{H}_N^d,M) = \Omega(\log^{\frac{d-1}{2}} M)$ *for* $N \ge M$,
(iii) $\operatorname{disc}^+(\mathcal{H}_N^d,M) = \Theta(\log M)$ *for* $d = 2$.

2.2 Geometric Discrepancy

As mentioned before, the use of geometric discrepancies in [7,8] in the analysis of declustering problems was a major breakthrough in this area. We refer to the recent book of Matoušek [14] for both a great introduction and a thorough treatment of this area.

The problem of geometric discrepancy in the unit cube $[0,1)^d$ is to distribute $n \in \mathbb{N}$ points evenly with respect to axis-parallel boxes: In every box R should be approximately $n\,\mathrm{vol}(R)$ points, where $\mathrm{vol}(R)$ denotes the volume of R. Again, discrepancy quantifies the distance to a perfect distribution. The discrepancy of a point set $\mathcal{P}$ with respect to a box $R \subseteq [0,1)^d$ is defined by

$$D(\mathcal{P},R) = |\,|\mathcal{P}\cap R| - n\,\mathrm{vol}(R)\,|\,,$$

the discrepancy of $\mathcal{P}$ for the set of all axis-parallel boxes $\mathcal{R}_d$ is

$$D(\mathcal{P}, \mathcal{R}_d) = \sup_{R \in \mathcal{R}_d} |D(\mathcal{P}, R)|$$

and the discrepancy of $\mathcal{R}_d$ for n-point sets is

$$D(n, \mathcal{R}_d) = \inf_{\mathcal{P} \subset [0,1)^d; |\mathcal{P}|=n} D(\mathcal{P}, \mathcal{R}_d).$$

3 The Lower Bound

The general idea in the proofs of the lower bound in Sinha et al. [7] and Anstee et al. [8] is the same, here described in two dimensions:

Starting with an arbitrary M–coloring of $[M]^2$, there is a monochromatic set $\hat{P}$ with M vertices. Based on this set, an M–point set $\mathcal{P}$ in $[0,1)^2$ is constructed. Schmidt's lower bound [15] ensures the existence of a rectangle R such that $D(\mathcal{P}, R) = \Omega(\log M)$. Rounding R to the $[M]^2$ grid, they construct a hyperedge $\hat{R}$ that has approximately the volume as R. Additionally $\hat{R}$ contains as many vertices of $\hat{P}$ as R points of $\mathcal{P}$. With the help of $\hat{R}$ and a short calculation the lower bound of the additive error $\Omega(\log M)$ is shown.

The small, but crucial mistake in the proof of Sinha et al. [7] is in the transfer from the geometric discrepancy setting back to the combinatorial one. Recall that the authors started with a color class of exactly M^{d-1} points (we lift their analysis to arbitrary dimension). They down-scaled it by a factor of M to a set in the unit cube (that, note this fact, is a subset of $\{0, \frac{1}{M}, \frac{2}{M}, \ldots, \frac{M-1}{M}\}^d$). Then their geometric discrepancy argument yields a rectangle of polylogarithmic discrepancy, which is "rounded" to obtain a subgrid with polylogarithmic discrepancy in the combinatorial setting. However, the rectangle $[0, \frac{M-1}{M}]^d$ has a much larger discrepancy: It contains all M^{d-1} points, but has a volume of $(\frac{M-1}{M})^d$ only.

This yields a discrepancy of $M^{d-1}(1-(\frac{M-1}{M})^d) = \Omega(M^{d-2})$. If the rounding argument of Sinha et al. [7] was correct, it would yield a subgrid with a discrepancy polynomial in M(for $d \geq 3$), which contradicts the known and new upper bounds. The problem is that rounding an arbitrary box to a box in the grid can cause a roundoff error which is of magnitude larger than the discrepancy. For this reason, a straight generalization of the proof of Anstee et al. [8] of the lower bound in two dimensions is not possible. In particular, we have to ensure the existence of a *small* box having large discrepancy. Beck and Chen [16] showed a lower bound for cubes with side at most $s := n^{-2/(2d+1)}$, where n is the number of points distributed in the unit cube $[0,1]^d$. Still, this is too large to control the rounding error. Following the notation introduced in Beck and Chen [16], the cube $[-s, s]^d$ has side s, we show

Theorem 3 *For any n–point set $\mathcal{P}$ in the unit cube $[0,1)^d$, there is an axis-parallel cube Q with side at most $n^{-\frac{(2d-3)d}{(d-1)^2(2d+1)}}$ fully contained in $[0,1)^d$ with*

$$D(\mathcal{P}, Q) = \Omega(\log^{\frac{d-1}{2}} n).$$

We first deduce Theorem 2 (ii) from Theorem 3.

Proof (Theorem 2 (ii)). We show the claim for $N = M$, which clearly implies the result for arbitrary $N \geq M$. Let $\chi : [M]^d \to [M]$ be a M–coloring of $\mathcal{H}_M^d$. Without loss of generality we may assume $|\chi^{-1}(1)| \geq M^{d-1}$. In the case $|\chi^{-1}(1)| \geq M^{d-1} + \frac{k}{2}\log^{\frac{d-1}{2}} M$, where k is the constant implicitly given in Theorem 3, we have $\operatorname{disc}(\mathcal{H}_M^d, \chi) \geq \left||\chi^{-1}(1)| - M^{d-1}\right| \geq \frac{k}{2}\log^{\frac{d-1}{2}} M$. Therefore, we may assume $|\chi^{-1}(1)| < M^{d-1} + \frac{k}{2}\log^{\frac{d-1}{2}} M$. For every vertex $z = (z_1, z_2, \ldots, z_d) \in \chi^{-1}(1)$ we define $x_z := \left(\frac{2z_1-1}{2M}, \frac{2z_2-1}{2M}, \ldots, \frac{2z_d-1}{2M}\right)$. Let $\mathcal{P} := \{x_z \mid z \in \chi^{-1}(1)\}$ and $n := |\mathcal{P}|$. By Theorem 3, there is a cube $Q = \prod_{i=1}^d [x_i, x_i + 2s)$ such that the side s is at most $n^{-\frac{(2d-3)d}{(d-1)^2(2d+1)}}$ and

$$D(\mathcal{P}, Q) = \left||\mathcal{P} \cap Q| - n\operatorname{vol}(Q)\right| \geq k\log^{\frac{d-1}{2}} M.$$

Now we construct a box B by rounding the x_i and $x_i + 2s$ to the nearest multiple of $\frac{1}{M}$. We ensure $\mathcal{P} \cap B = \mathcal{P} \cap Q$ by rounding up $x_i + 2s$ if $x_i + 2s = \frac{h}{2M}$ and rounding x_i down if $x_i = \frac{h}{2M}$ for an odd h.

Since we have chosen a relatively small cube Q, our rounding changes the volume not to much. Using $n \geq M^{d-1}$, we get

$$|\operatorname{vol}(Q) - \operatorname{vol}(B)| \leq 2d\tfrac{1}{2M}\left(\tfrac{1}{M} + 2s\right)^{d-1} < d3^{d-1}M^{-(d-1)}.$$

The combinatorial counterpart of B is the box

$$\hat{B} := \left\{x \in [M]^d \,\middle|\, \left(\tfrac{2x_1-1}{2M}, \ldots, \tfrac{2x_d-1}{2M}\right) \in B\right\}.$$

One can easily check that $M^d \operatorname{vol}(B) = |\hat{B}|$. By construction,

$$\begin{aligned}
\operatorname{disc}(\mathcal{H}_M^d, \chi) &\geq \left||\chi^{-1}(1) \cap \hat{B}| - \tfrac{1}{M}|\hat{B}|\right| \\
&= \left||\mathcal{P} \cap Q| - M^{d-1}\operatorname{vol}(B)\right| \\
&= \big|\left(|\mathcal{P} \cap Q| - n\operatorname{vol}(Q)\right) + \left(n\operatorname{vol}(Q) - M^{d-1}\operatorname{vol}(Q)\right) \\
&\quad + M^{d-1}\left(\operatorname{vol}(Q) - \operatorname{vol}(B)\right)\big| \\
&\geq \tfrac{k}{2}\log^{\frac{d-1}{2}} M - O(1) = \Omega\left(\log^{\frac{d-1}{2}} M\right).
\end{aligned}$$

Thus, $\operatorname{disc}(\mathcal{H}_M^d, M) = \Omega(\log^{\frac{d-1}{2}} M)$. It remains to show that this bound also holds for the positive discrepancy. To this end, let us assume that the discrepancy of the box $\hat{B}$ in color 1 is caused by a lack of vertices in color 1. Since $|\chi^{-1}(1)| \geq M^{d-1}$, the complement of $\hat{B}$ in $[M]^d$ has at least the same discrepancy as $\hat{B}$, but caused by an excess of vertices in color 1.

Though this complement is not a box, it is the union of at most $2d$ boxes. Therefore, one of these boxes has a positive discrepancy that is at least $\frac{1}{2d}$ times the discrepancy of $\hat{B}$ in color 1. □

This last argument increases the implicit constant of the lower bound by a factor of $\frac{3^d}{2d}$ compared to the approach of Sinha et al. [7].

To prove Theorem 3, we need some notions from Fourier analysis. Let $\mathcal{P} := \{p_1, p_2, \ldots, p_n\} \subseteq \mathbb{R}^d$ and $\nu := \sum_{i=1}^n \delta_{p_i} - n\mu$, where δ_{p_i} denotes the Dirac measure concentrated on p_i and μ is the d–dimensional Lebesgue measure on $[0,1]^d$ with $\mu([0,1]^d) = 1$. For any $\lambda \in (0,1]$ and $g \in L^2(\mathbb{R}^d)$ write $g_\lambda(x) := g(\lambda^{-1}x)$ for all $x \in \mathbb{R}^d$. Put $F_g := g * \nu$. Then we have

$$F_g(x) = \int_{\mathbb{R}^d} g(x-y)d\nu(y) = \sum_{i=1}^n g(x-p_i) - n\int_{\mathbb{R}^d} g(x-y)d\mu(y).$$

Let $\mathbb{1}_r$ be the characteristic function of the cube $[-r,r]^d$. Then $|F_{\mathbb{1}_r}(x)|$ is the discrepancy of $Q_r(x) := \big(x + [-r,r]^d\big) \cap [0,1]^d$ with respect to the set $\mathcal{P}$:

$$|F_{\mathbb{1}_r}(x)| = \big||\mathcal{P} \cap Q_r(x)| - n\,\mathrm{vol}(Q_r(x))\big| = \mathrm{disc}(\mathcal{P}, Q_r(x)).$$

Let $\Delta_1(g) := \int_{\mathbb{R}^d} |F_g(x)|^2 dx$ and $\Delta_2(g) := \int_0^1 \int_{\mathbb{R}^d} |F_{g_\lambda}(x)|^2 dx d\lambda$. Using Parseval's theorem for Fourier transforms we can show $\Delta_1(g) := \int_{\mathbb{R}^d} |\hat{g}(t)|^2 |\hat{\nu}(t)|^2 dt$ and $\Delta_2(g) := \int_{\mathbb{R}^d} \left(\int_0^1 |\hat{g}_\lambda(t)|^2 d\lambda\right) |\hat{\nu}(t)|^2 dt$. Here $\hat{f}$ denotes the Fourier transform

$$\hat{f} : \mathbb{R}^d \to \mathbb{C}, t \mapsto \hat{f}(t) = \frac{1}{(2\pi)^{\frac{d}{2}}} \int_{\mathbb{R}^d} f(x) e^{-ix\cdot t} dx$$

of $f : \mathbb{R}^d \to \mathbb{C}$. Let $m := n^{\frac{(2d-3)d}{(d-1)^2(2d+1)}}$. Note that $m > 1$. For the proof of Theorem 3 we need the following main lemma, which determines an average discrepancy for all cubes of side at most $\frac{1}{m}$ that intersect the unit cube $[0,1]^d$.

Lemma 4. *We have* $\Delta_2(\mathbb{1}_{\frac{1}{m}}) = \Omega(\log^{d-1} n)$.

Let us first derive Theorem 3 from Lemma 4.

Proof (Theorem 3). We distinguish two cases. Either there exists some $r \in [0, \frac{1}{m}]$ and $x_0 \in \mathbb{R}^d$ with $|F_{\mathbb{1}_r}(x_0)| > 2n(\frac{2}{m})^d$ or there does not. In the former case, the cube Q_0 with center x_0 and side r has discrepancy at least $2n(\frac{2}{m})^d$, as we have mentioned above. This cube may cross the border of $[0,1]^d$, but we can find a cube Q with side $\frac{1}{m}$ and $Q_0 \cap [0,1)^d \subseteq Q$ fully contained in $[0,1)^d$. With $n\,\mathrm{vol}(Q_0) = n(2r)^d \leq n(\frac{2}{m})^d$, we see that the discrepancy of Q_0 must be caused by the excess of points in Q_0. Therefore we have

$$D(\mathcal{P}, Q) \geq |\mathcal{P} \cap Q| - n\,\mathrm{vol}(Q) \geq n(\tfrac{2}{m})^d = 2^d n^{\frac{1}{(d-1)^2(2d+1)}} = \Omega(\log^{\frac{d-1}{2}} n).$$

Let us assume the latter case. Lemma 4 gives us a lower bound for the average square discrepancy of all cubes of side at most $\frac{1}{m}$. Since the contribution of cubes

intersecting the border of $[0,1]^d$ to this average square discrepancy is

$$O\left(\tfrac{1}{m}\left(n(\tfrac{1}{m})^d\right)^2\right) = O\left(n^{-\frac{d-2}{(d-1)^2}}\right) = O(1),$$

there is a cube Q with side at most $\frac{1}{m}$ and discrepancy $\Omega(\log^{\frac{d-1}{2}} n)$ fully contained in $[0,1]^d$. □

It remains to prove Lemma 4. We set for all $l = (l_1, l_2, \ldots, l_d) \in \mathbb{Z}^d$

$$h_l(x) := \prod_{i=1}^{d} \exp(-\tfrac{1}{2} l_i^2 x_i^2).$$

By the fact that $\hat{f}(t) = a^{-1}\exp(-\frac{t^2}{2a^2})$ for $f(x) = \exp(-\frac{1}{2}a^2x^2)$, the Fourier transform of h_l is $\hat{h}_l(t) = \prod_{i=1}^{d} \frac{1}{l_i}\exp\left(-\frac{t_i^2}{2l_i^2}\right)$. Now let L be the integer power of 2 satisfying $4(2\pi)^{\frac{d}{2}} n \le L < 8(2\pi)^{\frac{d}{2}} n$ and

$$\mathbb{Z}^d(L,m) := \left\{ l \in \mathbb{Z}^d \mid l_i = 2^{s_i} \ge m, s_i \in \mathbb{Z}, \prod_{i=1}^{d} l_i = L \right\}.$$

The following three lemmas yield the Lemma 4.

Lemma 5. $|\mathbb{Z}^d(L,m)| > \Omega(\log^{d-1} n)$.

Proof. Set $L' := \log_2 L$ and $m' := \lceil \log_2 m \rceil$. Then $|\mathbb{Z}^d(L,m)|$ is the number of integral lattice points $(s_1, s_2, \ldots, s_d)$ with $\sum_{i=1}^{d} s_i = L'$ and $s_i \ge m'$ for all $1 \le i \le d$. Hence

$$|\mathbb{Z}^d(L,m)| = \binom{L' - (m'-1)d - 1}{d-1} \ge \frac{(L' - m'd + 1)^{d-1}}{(d-1)!}.$$

With $L' \ge \log_2\left(4(2\pi)^{\frac{d}{2}} n\right) > \log_2 n + d + 1$ and $m' < \frac{(2d-3)d\log_2 n}{(d-1)^2(2d+1)} + 1$ we get

$$|\mathbb{Z}^d(L,m)| = \Omega(\log^{d-1} n).$$

□

The following two lemmas are taken from Beck and Chen [16]:

Lemma 6 ([16], Lemma 6.3). $\Delta_2(\mathbb{1}_{\frac{1}{m}}) = \Omega\left(\sum_{l \in \mathbb{Z}^d(L,m)} \Delta_1(h_l)\right)$.

Lemma 7 ([16], Lemma 6.4). *For every $l \in \mathbb{Z}^d(L,m)$ we have*

$$\Delta_1(h_l) = \Omega(1).$$

Now Lemma 4 is a direct consequence of Lemma 5, 6 and 7. We get

$$\Delta_2(\mathbb{1}_{\frac{1}{m}}) = \Omega\left(\sum_{l\in\mathbb{Z}^d(L,m)} \Delta_1(h_l)\right) = \sum_{l\in\mathbb{Z}^d(L,m)} \Omega(1) = \Omega(\log^{d-1} n).$$

It remains to prove the lower bound of Theorem 2 (iii). Anstee et al. [8] only treated latin square type colorings of $[M]^2$. However, the proof is easily extended through the triangle inequality argument used in the proof of Theorem 2 (ii).

4 The Upper Bound

In this section, we present a declustering scheme showing our upper bound. As in previous work, we use geometric discrepancies to construct the declustering scheme. In the following we use the notation of Niederreiter [17]. For an integer $b \geq 2$, an elementary interval in base b is an interval of the form $E = \prod_{i=1}^{d} \left[a_i b^{-d_i}, (a_i+1)b^{-d_i}\right)$, with integers $d_i \geq 0$ and $0 \leq a_i < b^{d_i}$ for $1 \leq i \leq d$. For integers t, m such that $0 \leq t \leq m$, a (t,m,d)–net in base b is a point set of b^m points in $[0,1)^d$ such that all elementary intervals with volume b^{t-m} contain exactly b^t points.

Note that any elementary interval with volume b^{t-m} has discrepancy zero in a (t,m,d)–net. Since any subset of an elementary interval of volume b^{t-m} has discrepancy at most b^t and any box can be packed with elementary intervals in a way that the uncovered part can be covered by $O(\log^{d-1} n)$ elementary intervals of volume b^{t-m}, the following is immediate:

Theorem 8 *A (t,m,d)–net $\mathcal{P}_{net}$ in base b with $n = b^m$ points has discrepancy*

$$D(\mathcal{P}_{net}, \mathcal{R}_d) = O(\log^{d-1} n).$$

The central argument in our proof of the upper bound is the following result of Niederreiter [17] on the existence of $(0,m,d)$–nets. From the view-point of application it is important that his proof is constructive.

Theorem 9 *Let $b \geq 2$ be an arbitrary base and $b = q_1 q_2 \dots q_u$ be the canonical factorization of b into prime powers such that $q_1 < \dots < q_u$. Then for any $m \geq 0$ and $d \leq q_1 + 1$ there exists a $(0,m,d)$–net in base b.*

We use $(0,m,d)$–nets to construct an M–coloring of $\mathcal{H}_M^d$ in Lemma 10. For the definition of these colorings, we need the following special elements of $\mathcal{E}_M^d$: A set $\prod_{j=1}^{d} I_j \in \mathcal{E}_M^d$ is called a *row* of $[M]^d$ if there is an $i \in [d]$ with $I_i = [1..M]$ and $|I_j| = 1$ for all $j \neq i$. In Lemma 11 we use the M–coloring of $\mathcal{H}_M^d$ to construct an M–coloring of $\mathcal{H}_N^d$ with same discrepancy.

Lemma 10. *Let $\mathcal{P}_{net}$ bea $(0,d-1,d)$–net in base M in $[0,1)^d$. Then there is an M–coloring χ_M of $\mathcal{H}_M^d = ([M]^d, \mathcal{E}_M^d)$ such that all rows of $[M]^d$ contain every color exactly once*[2] *and*

$$\mathrm{disc}(\mathcal{H}_M^d, \chi_M) \leq D(\mathcal{P}_{net}, \mathcal{R}_d).$$

[2] Some authors call this a permutation scheme for $[M]^d$

Proof. The net $\mathcal{P}_{net}$ consists of M^{d-1} points and all elementary intervals with volume M^{-d+1} contain exactly one point. In particular, all elementary "rows", i.e., all subsets $\prod_{j=1}^{d} I_j$ of $[0,1)^d$ such that there is an $i \in [d]$ with $I_i = [0,1)$ and for all $j \neq i$ there exist $a_j \in [0..M-1]$ with $I_j = [\frac{a_j}{M}, \frac{a_j+1}{M})$, contain exactly one point.

We construct a coloring χ_M of $\mathcal{H}_M^d = ([M]^d, \mathcal{E}_M^d)$ corresponding to the set $\mathcal{P}_{net}$. Let $\hat{\mathcal{P}} := \left\{ x \in [M]^d \,\middle|\, \mathcal{P}_{net} \cap \prod_{i=1}^{d} [\frac{x_i-1}{M}, \frac{x_i}{M}) \neq \emptyset \right\}$. Then each row of $[M]^d$ contains exactly one point of $\hat{\mathcal{P}}$. We define the coloring $\chi_M : [M]^d \to [M]$ by $\chi_M(y, x_2, \ldots, x_d) = i$ for all $x = (x_1, x_2, \ldots, x_d) \in \hat{\mathcal{P}}$, $i, y \in [M]$ such that $y \equiv x_1 + (i-1) \mod M$. Hence $\hat{\mathcal{P}}$ receives color 1, color class 2 is obtained from shifting $\hat{\mathcal{P}}$ along the first coordinate and so on. This defines an M–coloring χ_M of $\mathcal{H}_M^d = ([M]^d, \mathcal{E}_M^d)$ such that each row of $\mathcal{H}_M^d$ contains every color exactly once.

For this coloring it is sufficient to calculate $\max_{\hat{R} \in \mathcal{E}_M^d} \left| |\chi_M^{-1}(1) \cap \hat{R}| - \frac{1}{M}|\hat{R}| \right|$, because for each color $i \in [M]$ and each box $\hat{R} \in \mathcal{E}_M^d$ we get the same discrepancy for the box $\hat{R}'$, which is a copy of $\hat{R}$ shifted along the first dimension by $i-1$ and wrapped around perhaps, with respect to the color 1. If $\hat{R}'$ is wrapped around, it is the union of two boxes. Since whole rows have discrepancy zero, the discrepancy of those boxes is the same as the discrepancy of the the box between them, and we have

$$\operatorname{disc}(\mathcal{H}_M^d, \chi_M) = \max_{\hat{R} \in \mathcal{E}_M^d} \left| |\hat{\mathcal{P}} \cap \hat{R}| - \tfrac{1}{M}|\hat{R}| \right|.$$

Let $\hat{R} = \prod_{i=1}^{d} [x_i..y_i]$ an arbitrary hyperedge of $\mathcal{H}_M^d$. The associated box in $[0,1)^d$ is $R = \prod_{i=1}^{d} [\frac{x_i-1}{M}, \frac{y_i}{M})$. Then $|\hat{\mathcal{P}} \cap \hat{R}| = |\mathcal{P}_{net} \cap R|$ and $|\hat{R}| = M^d \operatorname{vol}(R)$. Thus the combinatorial discrepancy of $\hat{R}$ equals the geometric one of R. We have

$$\left| |\chi_M^{-1}(1) \cap \hat{R}| - \tfrac{1}{M}|\hat{R}| \right| = \left| |\mathcal{P}_{net} \cap R| - M^{d-1} \operatorname{vol}(R) \right| \leq D(\mathcal{P}_{net}, \mathcal{R}_d).$$

Hence we get $\operatorname{disc}(\mathcal{H}_M^d, \chi_M) \leq D(\mathcal{P}_{net}, \mathcal{R}_d)$. □

Lemma 11. *Let χ_M be an M–coloring of $\mathcal{H}_M^d$ such that all rows of $[M]^d$ contain every color exactly once and χ a coloring of $\mathcal{H}_N^d$ defined by $\chi(x_1, \ldots, x_d) = \chi_M(y_1, \ldots, y_d)$ with $x_i \equiv y_i \mod M$ for $i \in [d]$, $x_i \in [N]$, $y_i \in [M]$. Then*

$$\operatorname{disc}(\mathcal{H}_N^d, \chi) = \operatorname{disc}(\mathcal{H}_M^d, \chi_M).$$

Proof. Let $\hat{R} = \prod_{i=1}^{d} [x_i..y_i]$ be an arbitrary hyperedge of $\mathcal{H}_N^d$. For all $i \in [d]$ there exist unique $\tilde{x}_i, \tilde{y}_i \in [M]$ with $x_i \equiv \tilde{x}_i \mod M$ respectively $y_i \equiv \tilde{y}_i \mod M$. Set $\bar{x}_i := \min\{\tilde{x}_i, \tilde{y}_i\}$ and $\bar{y}_i := \max\{\tilde{x}_i, \tilde{y}_i\}$ for all $i \in [d]$. We have $\operatorname{disc}(\hat{R}, \chi) = \operatorname{disc}([\bar{x}_1..\bar{y}_1] \times [x_2..y_2] \times \ldots \times [x_d..y_d], \chi)$, since whole rows have discrepancy zero. Applying this successively in every coordinate we get

$$\operatorname{disc}(\hat{R}, \chi) = \operatorname{disc}(\textstyle\prod_{i=1}^{d} [\bar{x}_i..\bar{y}_i], \chi) = \operatorname{disc}(\prod_{i=1}^{d} [\bar{x}_i..\bar{y}_i], \chi_M).$$ □

Lemma 11 is a remarkable improvement of Theorem 4.2 in [9], where $\mathrm{disc}(\mathcal{H}_N^d, \chi) \leq 2^d \,\mathrm{disc}(\mathcal{H}_M^d, \chi_M)$ is shown. Note that this reduces the implicit constant in the upper bound by factor of 2^d.

It remains to show that the upper bound in Theorem 2 follows from Lemma 10 and Lemma 11.

Proof (*Theorem 2*(*i*)). Let $M \geq 3$ and $d \geq 2$ be positive integers and $d \leq q_1 + 1$, where q_1 is the smallest prime power in the canonical factorization of M into prime powers. Theorem 9 provides a $(0, d-1, d)$–net $\mathcal{P}_{net}$ in base M in $[0,1)^d$. Using Lemma 10 , we get an M–coloring χ_M of $\mathcal{H}_M^d$ such that all rows contain each color exactly once and $\mathrm{disc}(\mathcal{H}_M^d, \chi_M) \leq D(\mathcal{P}_{net}, \mathcal{R}_d)$. With Lemma 11 and Theorem 8, we have $\mathrm{disc}(\mathcal{H}_N^d, M) \leq D(\mathcal{P}_{net}, \mathcal{R}_d) = O(\log^{d-1} M)$. □

5 Conclusion

We gave lower and upper bounds for the declustering problem. This paper contains the first complete and correct proof of the lower bound $\Omega(\log^{\frac{d-1}{2}} M)$ for arbitrary values of M and d. Moreover, the implicit constant was improved by a factor of $\frac{3^d}{2d}$.

We propose a declustering scheme that has an additive error of $O(\log^{d-1} M)$ with the sole condition that $d \leq q_1 + 1$, where q_1 is the smallest prime power in the canonical factorization of M into prime powers. This improves the former best declustering schemes of Chen and Cheng [9], where either bounds depend on the data size N^d or $M = p^t$ and $p \geq d$ was required for a prime p and $t \in \mathbb{N}$. Furthermore, Lemma 11 improves the analysis of Chen and Cheng [9] of the discrepancy of latin square colorings by a factor of 2^{-d}.

The natural problem to close the gap between the lower and upper bound is probably a very hard one. The reason is that the corresponding problem of geometric discrepancies of rectangles seems to be extremely difficult. Closing the gap between the $\Omega(\log^{\frac{d-1}{2}} n)$ lower and the $O(\log^{d-1} n)$ upper bound was baptized 'the great open problem' already in Beck and Chen [16]. Since then no further progress has been made for the general problem (note that in the proof of a slight improvement due to Baker [18] recently a serious bug was found [talk of József Beck, Oberwolfach Seminar on Discrepancy Theory and Applications, March 2004]).

References

1. Chang, C., Moob, B., Archarya, A., Shock, C., Sussman, A., Saltz, J.: Titan: a high performance remote-sensing database. In: Proc. of International Conference on Data Engineering. (1997) 375–384
2. Chen, C.M., Bhatia, R., Sinha, R.K.: Multidimensional declustering schemes using golden ratio and kronecker sequences. In: IEEE Trans. on Knowledge and Data Engineering. Volume 15. (2003)

3. Prabhakar, S., Abdel-Ghaffar, K., Agrawal, D., Abbadi, A.E.: Cyclic allocation of twodimensional data. In: 14th International Conference on Data Engineering, Orlando, Florida (1998) 94–101
4. Atallah, M.J., Prabhakar, S.: (Almost) optimal parallel block access for range queries. In: Symposium on Principles of Database Systems, Dallas (2000) 205–215
5. Du, H.C., Sobolewski, J.S.: Disk allocation for cartesian product files on multiple disk systems. ACM Trans. Database Systems **7** (1982) 82–101
6. Faloutsos, C., Bhagwat, P.: Declustering using fractals. In: Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems, San Diego, CA (1993) 18 – 25
7. Sinha, R.K., Bhatia, R., Chen, C.M.: Asymptotically optimal declustering schemes for 2-dim range queries. Theoret. Comput. Sci. **296** (2003) 511–534
8. Anstee, R., Demetrovics, J., Katona, G.O.H., Sali, A.: Low discrepancy allocation of two-dimensional data. In: Foundations of Information and Knowledge Systems, First International Symposium. Volume 1762 of Lecture Notes in Computer Science. (2000) 1–12
9. Chen, C.M., Cheng, C.: From discrepancy to declustering: near optimal multidimensional declustering strategies for range queries. In: ACM Symp. on Database Principles, Madison, WI (2002) 29–38
10. Doerr, B., Srivastav, A.: Approximation of multi-color discrepancy. In Hochbaum, D., Jansen, K., Rolim, J.D.P., Sinclair, A., eds.: Randomization, Approximation and Combinatorial Optimization (Proceedings of APPROX-RANDOM 1999). Volume 1671 of Lecture Notes in Computer Science. (1999) 39–50
11. Doerr, B., Srivastav, A.: Multicolour discrepancies. Combinatorics, Probability and Computing **12** (2003) 365–399
12. Biedl, T., Čenek, E., Chan, T., Demaine, E., Demaine, M., Fleischer, R., Wang, M.: Balanced k-colorings. Discrete Math. **254** (2002) 19–32
13. Babai, L., Hayes, T.P., Kimmel, P.G.: The cost of the missing bit: communication complexity with help. Combinatorica **21** (2001) 455–488
14. Matoušek, J.: Geometric Discrepancy. Springer-Verlag, Berlin (1999)
15. Schmidt, W.M.: On irregularities of distribution VII. Acta Arith. **21** (1972) 45–50
16. Beck, J., Chen, W.L.: Irregularities of distribution. Volume 89 of Cambridge Tracts in Mathematics. Cambridge University Press, Cambridge (1987)
17. Niederreiter, H.: Point sets and sequences with small discrepancy. Monatsh. Math. **104** (1987) 273–337
18. Baker, R.C.: On irregularities of distribution II. J. London Math. Soc.(2) **59** (1999) 50–64

Crossing Number Is Hard for Cubic Graphs

Petr Hliněný

Department of Computer Science
VŠB – Technical University of Ostrava,
17. listopadu 15, 708 33 Ostrava, Czech Republic
and Institute for Theoretical Computer Science*
Charles University, Czech Republic
petr.hlineny@vsb.cz

Abstract. It was proved by [Garey and Johnson, 1983] that computing the crossing number of a graph is an *NP*-hard problem. Their reduction, however, used parallel edges and vertices of very high degrees. We prove here that it is *NP*-hard to determine the crossing number of a simple cubic graph. In particular, this implies that the minor-monotone version of crossing number is also *NP*-hard, which has been open till now.

Keywords: Crossing number, cubic graph, NP-completeness.

2000 Math Subjects Classification: 05C10, 05C62, 68R10

1 Background on Crossing Number

We assume that the reader is familiar with basic terms of graph theory. In this paper we consider finite simple graphs, unless we specifically speak about multigraphs. A graph is cubic if it has all vertices of degree 3.

In a *(proper) drawing* of a graph G in the plane the vertices of G are points and the edges are simple curves joining their endvertices. Moreover, it is required that no edge passes through a vertex (except at its ends), and that no three edges intersect in a common point which is not a vertex. An *edge crossing* is an intersection point of two edges-curves in the drawing which is not a vertex. The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum number of edge crossings in a proper drawing of G in the plane (thus, a graph is planar if and only if its crossing number is 0). We remark that there are other possible definitions of crossing number which are supposed, but not(!) known [9], to be equivalent to each other.

Crossing number problems were introduced by Turán, whose work in a brick factory during the Second World War led him to inquire about the crossing number of the complete bipartite graphs $K_{m,n}$. Turán devised a natural drawing of $K_{m,n}$ with $\lfloor \frac{m}{2} \rfloor \lfloor \frac{m-1}{2} \rfloor \lfloor \frac{n}{2} \rfloor \lfloor \frac{n-1}{2} \rfloor$ crossings, but the conjecture of Zarankiewic that such a drawing is the best possible, is still wide open. (Look at an interesting

* ITI is supported by the Ministry of Education of the Czech Republic as project LN00A056.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 772–782, 2004.
© Springer-Verlag Berlin Heidelberg 2004

story of a false "proof" of the conjecture [6].) Not surprisingly, exact crossing numbers are in general very difficult to compute. As an example of another graph family whose crossing number has been deeply studied, we mention the Cartesian products of cycles $C_m \times C_n$ — their crossing number $m(n-2)$ for $m \geq n$ was conjectured in [7]. There has been a number of particular results on this difficult problem (such as [8] as one example), and, remarkably, the problem is almost solved now [4]. That is one of only a few nontrivial exact crossing numbers known today.

The algorithmic problem CROSSINGNUMBER is given as follows:

Input: A multigraph G and an integer k.
Question: Is it true that $\mathrm{cr}(G) \leq k$?

Computing the crossing number has important applications in, for example, VLSI design, or in graph visualization. The problem is in *NP* since one could guess the optimal drawing, replace the crossings in it with new (simultaneously subdividing) vertices, and verify planarity of the resulting graph. It has been proved by Garey and Johnson [3] that CROSSINGNUMBER is an *NP*-complete problem for k on the input.

Since then, a new significant complexity result about graph crossing number has appeared only recently — a paper by Grohe [5] presenting a quadratic-time (FPT) algorithm for CROSSINGNUMBER(k) with constant k. There is also a long-standing open question, originally asked by Seese: What is the complexity of CROSSINGNUMBER for graphs of fixed tree-width? (Here we leave aside other results dealing with various restricted versions of the crossing number problem appearing in connection with VLSI design or with graph drawing, such as the "layered" or "rectilinear" crossing numbers etc.)

Before the above mentioned FPT algorithm of Grohe for crossing number appeared; Fellows [1] had observed that there are finitely many excluded minors for the cubic graphs of crossing number at most k, which implied a (nonconstructive) algorithm for CROSSINGNUMBER(k) with constant k over cubic graphs. That observation might still suggest that CROSSINGNUMBER was easier to solve over cubic graphs than in general. However, that is not so, as we show in this paper.

2 Crossing Number and OLA

We first define another classical *NP*-complete combinatorial problem [2] called OPTIMALLINEARARRANGEMENT, which is given as follows:

Input: An n-vertex graph G, and an integer a.
Question: Is there a bijection $\alpha : V(G) \to \{1, \ldots, n\}$ (a *linear arrangement* of vertices) such that the following holds

$$\sum_{uv \in E(G)} |\alpha(u) - \alpha(v)| \leq a\,? \tag{1}$$

The sum on the left of (1) is called the *weight* of α.

The above mentioned paper [3] actually reduces CROSSINGNUMBER from OPTIMALLINEARARRANGEMENT. We, however, consider that reduction "unrealistic" in the following sense: The reduction in [3] creates many large classes of parallel edges, and it uses vertices of very high degrees. (There seems to be no easy modification avoiding those.) So we consider it natural to ask what can be said about the crossing number problem on simple graphs with small vertex degrees.

It might be tempting to construct a "nicer" polynomial reduction for CROSSINGNUMBER from another *NP*-complete problem called Planar-SAT (a version of the satisfiability problem with a planar incidence graph). There have been, to our knowledge, a few attempts in this directions, so far unsuccessful. We consider this phenomenon remarkable since Planar-SAT seems to be much closer to crossing-number problems than the Linear Arrangement is.

Still, we have found another construction reducing CROSSINGNUMBER from OPTIMALLINEARARRANGEMENT, which produces cubic graphs. The basic idea of our construction is similar to [3], but the restriction to degree-3 vertices brings many more difficulties to the proofs. The construction establishes our main result which reads:

Theorem 1. *The problem* CROSSINGNUMBER *is NP-complete for 3-connected (simple) cubic graphs.*

Let us, moreover, define the *minor-monotone crossing number* $\mathrm{mcr}(G)$: A *minor* F of a graph G is a graph obtained from a subgraph of G by contractions of edges. Then $\mathrm{mcr}(G)$ as the smallest crossing number $\mathrm{cr}(H)$ over all graphs H having G as a minor. The traditional versions of crossing number do not behave well with respect to taking minors; one may find graphs G such that $\mathrm{cr}(G) = 1$ but $\mathrm{cr}(G')$ is arbitrarily large for a minor G' of G. On the other hand, $\mathrm{mcr}(G') \leq \mathrm{mcr}(G)$ for a minor G' of G by definition. The algorithmic problem MM-CROSSINGNUMBER (from "Minor-Monotone") is defined as follows:

Input: A multigraph G and an integer k.
Question: Is it true that $\mathrm{mcr}(G) \leq k$?

Our main result immediately extends to a proof that also $\mathrm{mcr}(G)$ is *NP*-hard to compute, which has been an open question till now.

Corollary 1. *The problem* MM-CROSSINGNUMBER *is NP-complete.*

Observation. Let a cubic graph G be a minor of a multigraph H. Then some subdivision of G is contained as a subgraph in H. Hence $\mathrm{cr}(G) \leq \mathrm{cr}(H)$.

Thus $\mathrm{cr}(G) = \mathrm{mcr}(G)$ for cubic graphs, and the corollary follows directly from Theorem 1.

3 The Cubic Reduction

Let us call a *cubic grid* the graph illustrated in Figure 1 (looking like a "brick wall"). We say that the cubic-grid *height* equals the number of the "horizontal" paths, and the *length* equals the number of edges on the "top-most" horizontal path. (The positions are referred to as in Figure 1.) Formally, the cubic grid of even height h and length ℓ, denoted by $\mathcal{C}'_{h,\ell}$, is defined

$$V(\mathcal{C}'_{h,\ell}) = \{v_{i,j} : i = 1,2,\ldots,h;\ j = 0,1,\ldots,\ell\} \cup$$

$$\cup \{w_{i,j} : i = 2,3,\ldots,h-1;\ j = 1,2,\ldots,\ell\},$$

$$E(\mathcal{C}'_{h,\ell}) = \{v_{2i-1,j}v_{2i,j} : i = 1,2,\ldots,h/2;\ j = 0,1,\ldots,\ell\} \cup$$

$$\cup \{w_{2i,j}w_{2i+1,j} : i = 1,2,\ldots,h/2-1;\ j = 1,2,\ldots,\ell\} \cup$$

$$\cup \{v_{i,j-1}w_{i,j}, w_{i,j}v_{i,j} : i = 2,3,\ldots,h-1;\ j = 1,2,\ldots,\ell\} \cup$$

$$\cup \{v_{i,j-1}v_{i,j} : i = 1,h;\ j = 1,2,\ldots,\ell\}.$$

Suppose we now identify the "left-most" vertices in the grid $\mathcal{C}'_{h,\ell}$ with the "right-most" ones, formally $v_{i,0} = v_{i,\ell}$ for $i = 1,2,\ldots,h$, and simplify the resulting graph. Then we obtain the *cyclic cubic grid* $\mathcal{C}_{h,\ell}$ (which is, indeed, a cubic graph).

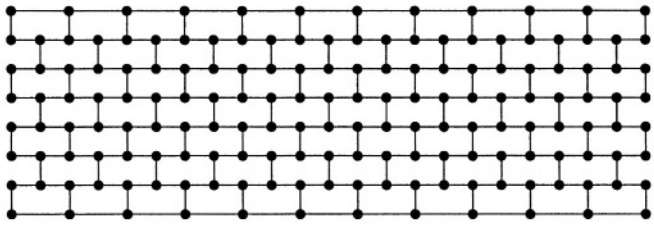


Fig. 1. An illustration of a cubic grid (a fragment of length 11 and height 8)

Let us have a cubic grid $\mathcal{C}'_{h,\ell}$ or $\mathcal{C}_{h,\ell}$ as above. We say that an edge f is *attached to the grid at low position* j if the edge $v_{1,j-1}v_{1,j}$ is subdivided with a vertex x_f, where x_f is an endvertex of f as well. We say that f is *attached at high position* j if an analogous construction is done for the edge $v_{h,j-1}v_{h,j}$. Notice that the new vertex x_f introduced when attaching an edge f has degree 3, and that the degrees of other vertices are unchanged. Similarly, a vertex x is *attached to the grid at position* j if two new edges f, f' with a common endvertex x are attached via their other endvertices at low and high positions j, respectively, to our cubic grid. This is illustrated on a detailed picture in Figure 2.

In a cyclic cubic grid $\mathcal{C}_{h,\ell}$, the cycles M^i on vertices $v_{i,0}w_{i,1}v_{i,1}w_{i,2}$-$\ldots v_{i,\ell-1}w_{i,\ell}$ for $i = 2,3,\ldots,h-1$, and on vertices $v_{i,0}v_{i,1}\ldots v_{i,\ell-1}$ for $i = 1,h$, are called the *main cycles* of the grid $\mathcal{C}_{h,\ell}$. M^1 and M^h are also referred to as

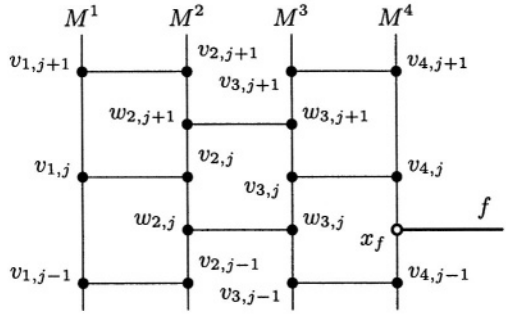


Fig. 2. A detail of the cyclic cubic grid $\mathcal{C}_{4,\ell}$, with an edge f attached at high position j

the *outer* main cycles. We use the same names, main cycles, for the subdivisions of the cycles M^i in graphs created from the grid $\mathcal{C}_{h,\ell}$ by attaching edges.

Assume now that we are given a graph G on n vertices. In order to prove Theorem 1, we are going to construct a cubic graph H_G depending on G. (Although our graph H_G is huge, it has polynomial size in G.) We show then how one can compute the weight of an optimal linear arrangement for G from the crossing number $\mathrm{cr}(H_G)$, and vice versa. Our construction uses several size parameters defined next:

$$\begin{aligned} n &= |V(G)|,\ m = |E(G)|, \\ t &= 2mn, \\ r &= t^2 = 4m^2n^2, \\ s &= m^3 r = 4m^5n^2, \\ q &= (m^3+n+1)r = 4m^5n^2 + 4m^2n^3 + 4m^2n^2, \\ z &= 2((s+rn)nt+r) = 16m^6n^4 + 16m^3n^5 + 8m^2n^2. \end{aligned} \tag{2}$$

Without loss of generality we may assume that the graph G is sufficiently large, say

$$m > n > 100\,. \tag{3}$$

We start with two copies B_1, B_2 of the cyclic cubic grid $\mathcal{C}_{z,q}$, called here the *boulders* (for their huge size that keeps the rest of our graph "in place"). Then we make n disjoint copies $R_1, \ldots, R_n$ of the cyclic cubic grid $\mathcal{C}_{t,q}$, called here the *rings*. An intermediate step in the construction – our graph $H_{m,n}$, is obtained by the following operations:

- Start with the disjoint union $B_1 \cup B_2 \cup R_1 \cup \ldots \cup R_n$ of the two boulders and the n rings.
- For every pair of integers $0 \le i < m^3$ and $0 \le j < r$, take a new edge κ_{i+jm^3}, and attach κ_{i+jm^3} at low positions $i + j(m^3+n+1) < q$ to the boulder B_1 via one end, and to B_2 via the other end. These s new edges $\kappa_0, \ldots, \kappa_{s-1}$ are called the *free spokes* in $H_{m,n}$.

– For every pair of integers $1 \leq i \leq n$ and $0 \leq j < r$, set $p = i - 1 + m^3 + j(m^3 + n + 1) < q$, and take two new vertices $\nu_{i,j}$ and $\nu'_{i,j}$ connected by an edge $\mu^3_{i,j}$. Then attach a new edge $\mu^1_{i,j}$ with one end $\nu_{i,j}$ (new edge $\mu^5_{i,j}$ with one end $\nu'_{i,j}$) to the boulder B_1 (B_2) at low position p via the other end. Finally, attach a new edge $\mu^2_{i,j}$ with one end $\nu_{i,j}$ (new edge $\mu^4_{i,j}$ with one end $\nu'_{i,j}$) to the ring R_i at low (high) position p via the other end. The path formed by three edges $\mu^1_{i,j}, \mu^3_{i,j}, \mu^5_{i,j}$ is called the *j-th ring spoke* of R_i in $H_{m,n}$.

We remark that the above construction attaches only one edge at the same position of each of the boulders and rings, and so the operations are well-defined. (Figure 3.) This remark applies also to further constructions on the graph H_G.

To simplify our notation, the above names of the boulders B_1, B_2 and the rings R_i are inherited to the subdivisions of those boulders and rings created in the construction of $H_{m,n}$. The same simplified notation is used further for the graph H_G, too.

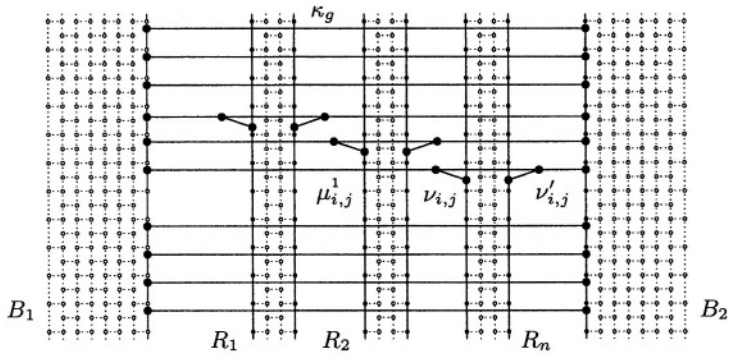


Fig. 3. How to attach free and ring spokes in the graph $H_{m,n}$

So far, the constructed graph $H_{m,n}$ does not depend on a particular structure of G, but only on its size and our choice of the parameters (2). One may say that $H_{m,n}$ acts as a skeleton in the forthcomming construction, in which the rings of $H_{m,n}$ shall model the vertices of G, and the order the rings are drawn in shall correspond to a linear arrangement of vertices of G. The following simple lemma shows necessary "flexibility" of drawings of $H_{m,n}$ with any order of the rings. (Actually, the number of crossings in the lemma is optimal, as we implicitly show in Section 4.)

Lemma 1. *For any permutation π of the set $\{1, 2, \ldots, n\}$, there is a drawing of the graph $H_{m,n}$ with $(s + rn)nt$ crossings conforming to the following: The subdrawings of all the rings are pairwise disjoint, each ring separates the two boulders in $H_{m,n}$ from each other, and any free spoke in the drawing intersects all the rings in order $R_{\pi(1)}, \ldots, R_{\pi(n)}$ from B_1 to B_2.*

Finally, the particular graph H_G needed for our polynomial reduction from G is constructed as follows:

- Start with the graph $H_{m,n}$, for $n = |V(G)|$ and $m = |E(G)|$. Number the vertices $V(G) = \{1, 2, \ldots, n\}$.
- For every ordered pair $0 < i, j \leq n$ such that $\{i, j\} \in E(G)$, set $p = (i - 1 + jn - n)4m^2(m^3 + n + 1) + m^3 + n < q$. In the graph $H_{m,n}$, attach new vertices χ_{ij}, χ'_{ij} to the rings R_i, R_j, respectively, at positions p, and add a new edge $\chi_{ij}\chi'_{ij}$. The subgraph $X_{i,j}$ induced on the five new edges incident with χ_{ij}, χ'_{ij} is called a *handle* of the edge ij in H_G. (Figure 4.)

That is, the rings in H_G model the vertices of G, and the handles model the edges of G. As we show later, an optimal drawing of H_G uniquely determines an ordering of the rings of $H_{m,n}$, and hence the weight of an optimal linear arrangement of G corresponds to the number of crossings between the rings and the handles in an optimal drawing of the graph H_G.

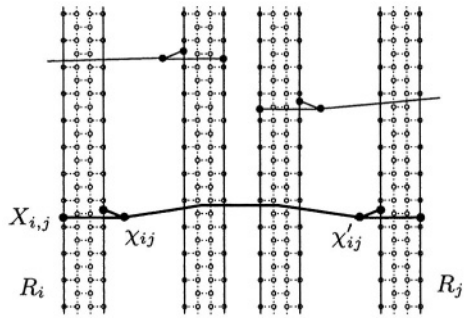


Fig. 4. How to attach handles of the edges of G in the graph H_G

We conclude this section with an upper bound on the crossing number of our constructed graph, which naturally follows from the drawings introduced in Lemma 1.

Lemma 2. *Let us, for a given graph G, construct the graph H_G as described above. If G has a linear arrangement of weight A, then the crossing number of H_G is*

$$\mathrm{cr}(H_G) \leq (s + rn)nt + 2(A + m)t - 4m\,,$$

where the weight of a linear arrangement is defined by (1) on page 773, and m, n, r, s, t are given by (2) on page 776.

Corollary 2. *For any G conforming to (3), $\mathrm{cr}(H_G) < z/2 = (s + rn)nt + r$.*

4 Sketch of the Proof

To prove correctness of our reduction, we now have to show a lower bound on the crossing number of our graph H_G, depending on weight of the optimal linear arrangement of G. We achieve this goal by showing that an optimal drawing of H_G has to look (almost) like the drawing described in the proof of Lemma 2. Since the whole proof is quite long and technical, and it uses more topological rather than combinatorial arguments, we give here only a brief outline of the main steps.

We argue that the boulders of H_G have to be drawn without crossings at all, and that each ring has to separate the two boulders from each other (and hence the rings are "nested" in each other). Such a configuration already forces the number of crossings of $H_{m,n}$ as in Lemma 1. Then we identify a linear ordering of the rings, and show that every edge handle in H_G generates at least as many additional crossings as expected from the ordering of rings. A special attention has to be paid to proving that no edge crossing is counted twice in our arguments.

The following particular claim will be useful during the proof.

Lemma 3. *Let k, ℓ, t be integers, and let $(p_1, p_2, \ldots, p_k)$ be an increasing sequence of integers such that $p_1 > 4kt$, $p_k < \ell$, and $p_{j+1} - p_i \geq 4kt$ for $j = 1, 2, \ldots, k$. Assume that the graph F is constructed from the cyclic cubic grid $C_{t,\ell}$ by attaching a vertex z_j at position p_j for each $j = 1, 2, \ldots, k$. Then* $\mathrm{cr}(F) = k(t-2)$.

We need to be a bit more formal in this section. A *curve* γ is a continuous function mapping the interval [0, 1] to a topological space. A curve γ is a *closed* curve if $\gamma(0) = \gamma(1)$. A closed curve γ is *contractible* in a topological space if γ can be continuously deformed to a single point there. We call a *cylinder* the topological space obtained from the unit square by identifying one pair of opposite edges in the same direction. (A cylinder has two disjoint closed curves as the boundary.)

Recall the notation from Section 3, and assume that G is a graph on the vertex set $\{1, 2, \ldots, n\}$. Let H_G denote the graph constructed along the description on page 778. The following statement, together with Lemma 2, validates our reduction.

Proposition 1. *If an optimal linear arrangement of a graph G has weight A, then the crossing number of the graph H_G is at least*

$$\mathrm{cr}(H_G) \geq (s + rn)nt + 2(A + m)t - 8m\,.$$

(See (2) and Lemma 2 for details on the notation.)

We proceed the proof of Proposition 1 along the following sequence of claims. Assume that we have an optimal drawing of the graph H_G at hand.

Lemma 4. *In the optimal drawing of H_G, the boulders B_1, B_2 are drawn with no edge crossings.*

Hence, in particular, the first main cycles N_j of the boulder B_j, $j = 1, 2$ are drawn with no crossings. Then there is a uniquely defined cylinder Π with the boundary curves N_1 and N_2 in the plane. Realize that the whole subgraph $H_G - V(B_1) - V(B_2)$ is drawn on Π.

Lemma 5. *In the optimal drawing of H_G, each main cycle M of every ring R_i, $i \in \{1, 2, \ldots, n\}$ is drawn as a closed curve separating the subdrawing of the boulder B_1 from the subdrawing of B_2, i.e. noncontractible on Π.*

This claim is the first key step in the proof of Proposition 1. The idea behind is that a main cycle drawn as a contractible curve on Π would have to have too many crossings with the free spokes of H_G.

Corollary 3. *In the optimal drawing of H_G, there are at least $(s+rn)nt$ crossings between edges of the main cycles of the rings and edges of the free and ring spokes in H_G.*

Lemma 6. *There is a selection of main cycles $M_i \subset R_i$, $i = 1, 2, \ldots, n$ of the rings in H_G, such that the cycles $M_1, \ldots, M_n$ are drawn as pairwise disjoint closed curves in the above optimal drawing of H_G. Hence, there is a permutation π of $\{1, \ldots, n\}$ such that, for each $j = 1, \ldots, n$, the closed curve $M_{\pi(j)}$ separates the subdrawing $B_1 \cup M_{\pi(1)} \cup \ldots \cup M_{\pi(j-1)}$ from the subdrawing $B_2 \cup M_{\pi(j+1)} \cup \ldots \cup M_{\pi(n)}$.*

Lemma 7. *For every $k = 0, 1, \ldots, 4n^2 - 1$, there is an index $c_k \in C_k = \{km^5 - 2m^4, \ldots, km^5 + 2m^4\}$ such that the edge of the c_k-th free spoke κ_{c_k} is crossed exactly once by each of the main cycles of all the rings, and that κ_{c_k} has no more crossings than those in the optimal drawing of H_G.*

Recall that the vertices of G are numbered as $\{1, 2, \ldots, n\}$, and that $X_{i,j}$ denotes the subgraph of the handle in the constructed graph H_G corresponding to an edge $ij \in E(G)$ (page 778).

Lemma 8. *Let π be the permutation from Lemma 6, let Π be the cylinder defined after Lemma 4 for the optimal drawing of H_G, and let $\{i, j\} \in E(G)$ be an edge. For $\ell = i + n(j-1)$, consider the indices $c_{4\ell-2}$ and $c_{4\ell+2}$ given by Lemma 7, and denote by Σ_ℓ the region on Π bounded by the drawings of the $c_{4\ell-2}$, $c_{4\ell+2}$-th free spokes and containing the subdrawing of the handle $X_{i,j}$. Then Σ_ℓ contains at least*

$$t\left(|\pi^{-1}(i) - \pi^{-1}(j)| - 1\right)$$

crossings between edges of the subgraph $X_{i,j} \cup R_i \cup R_j$ and edges of the main cycles of other rings R_k in H_G for $k \neq i, j$.

The last claim presents the second key step of our proof of Proposition 1. It shows that the handles $X_{i,j}$ (of edges of G) really have to cross all the rings which are between R_i, R_j in the order given by π. Moreover, the crossings can

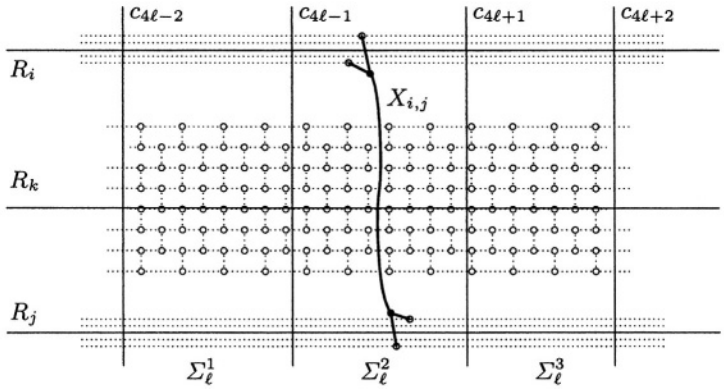


Fig. 5. An illustration to Lemma 8

be separated in distinct regions of the cylinder Π, and so it is clear that they will not be counted twice. See an illustration in Figure 5. Now we are ready to finish the proof.

We are going to count three collections of edge crossings in the optimal drawing of H_G. Firstly, there are (at least) $(s+rn)nt$ crossings described in Corollary 3. Secondly, denote by d_i the degree of the vertex i in G. Let us consider the subgraph F_i of H_G formed by the ring R_i and by $2d_i$ pairs of incident edges from all handles which are attached to R_i in H_G. Then, by Lemma 3, the subgraph F_i itself has at least $2d_i(t-2)$ edge crossings in any drawing of H_G. Thirdly, the permutation π from Lemma 6 defines a linear arrangement $\alpha = \pi^{-1}$ of the vertices of G. An edge $\{i,j\}$ of G then contributes (via its two handles in H_G) with at least $2t\big(|\alpha(i)-\alpha(j)|-1\big)$ crossings in H_G by Lemma 8.

Altogether, we have found at least this many distinct edge crossings in the optimal drawing of H_G:

$$(s+rn)nt + \sum_{i\in V(G)} 2d_i(t-2) + \sum_{\{i,j\}\in E(G)} 2t\big(|\alpha(i)-\alpha(j)|-1\big) =$$

$$= (s+rn)nt + 2t \sum_{\{i,j\}\in E(G)} |\alpha(i)-\alpha(j)| - 2tm + 4tm - 8m =$$

$$= (s+rn)nt + 2tA + 2tm - 8m$$

Proof of Theorem 1. Assume that G, a is an input instance of the OPTIMAL-LINEARARRANGEMENT problem, and that G is sufficiently large (3). The above described graph H_G is clearly cubic, it has polynomial size in $n = |V(G)|$, and H_G has been constructed efficiently. We now ask the problem CROSSING-NUMBER on the input $\langle H_G, (s+rn)nt + 2t(a+m)\rangle$, and give the same answer to OPTIMALLINEARARRANGEMENT on $\langle G, a\rangle$.

If there is a linear arrangement of G of weight at most a, then our correct answer is YES according to Lemma 2. Conversely, if the optimal linear arrangement of G has weight greater than a, then the crossing number of H_G is by Proposition 1

$$\mathrm{cr}(H_G) \geq (s+rn)nt + 2t(a+1+m) - 8m > \\ > (s+rn)nt + 2t(a+m)\,,$$

and so the correct answer is NO. Since the OPTIMALLINEARARRANGEMENT problem is known to be *NP*-complete [2], the statement of Theorem 1 follows. □

References

1. M.R. Fellows, *The Robertson–Seymour Theorems: A Survey of Applications,* Contemp. Math. 89 (1989), 1–18.
2. M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman and Company, New York 1979.
3. M.R. Garey, D.S. Johnson, *Crossing number is NP-complete,* SIAM J. Algebraic Discrete Methods 4 (1983), 312–316.
4. L.Yu. Glebsky, G. Salazar, *The crossing number of $C_m \times C_n$ is as conjectured for $n > m(m+1)$,* J. Graph Theory, to appear.
5. M. Grohe, *Computing Crossing Numbers in Quadratic Time,* In: 32nd ACM Symposium on Theory of Computing STOC '01, (2001), 231–236.
6. R.K. Guy, *The decline and fall of Zarankiewic's theorem,* in: Proc. Proof Techniques in Graph Theory (F. Harary ed.), Academic Press, N.Y. 1969, 63–69.
7. F. Harary, P.C. Kainen, A.J. Schwenk, *Toroidal graphs with arbitrarily high crossing numbers,* Nanta Math. 6 (1973), 58–67.
8. M. Klešč, R.B. Richter, I. Stobert, *The crossing number of $C_5 \times C_n$,* J. Graph Theory 22 (1996), 239–243.
9. J. Pach, G. Tóth, *Which crossing number is it, anyway?,* Proc. 39th Foundations of Computer Science (1998), IEEE Press 1999, 617–626.
10. R.B. Richter, A problem submitted to: *Problems in Topological Graph Theory* (D. Archdeacon), `http://www.emba.uvm.edu/~archdeac/problems/problems.html`.

A Reducibility for the Dot-Depth Hierarchy

Victor L. Selivanov[1]* and Klaus W. Wagner[2]

[1] A.P. Ershov Institute of Informatics Systems
Siberian Division of the Russian Academy of Sciences
vseliv@ngs.ru
[2] Institut für Informatik
Julius-Maximilians-Universität Würzburg
wagner@informatik.uni-wuerzburg.de

Abstract. Hierarchies considered in computability theory and in complexity theory are related to some reducibilities in the sense that levels of the hierarchies are downward closed and have complete sets. In this paper we propose a reducibility having similar relationship to the Brzozowski's dot-depth hierarchy and some its refinements. We prove some basic facts on the corresponding degree structure and discuss relationships of the reducibility to complexity theory (via the leaf-language approach).

1 Introduction

Hierarchies considered in descriptive set theory, computability theory and complexity theory are related to some reducibilities in the sense that levels of the hierarchies are downward closed and have complete sets under the respective reducibility. The complete sets are then used for estimating "complexity' of other sets and problems. In descriptive set theory this is the Wadge reducibility, in computability theory the m-reducibility, in complexity theory the polynomial-time m-reducibility.

In this paper we propose a reducibility having similar relationship to Brzozowski's dot-depth hierarchy (DDH) [5]. The reducibility is a version of the so called *logical reducibilities* defined by interpretations. Such reducibilities are intensively considered in the finite model theory (see [7,13] and references therein). To the best of our knowledge, such reducibilities were not so far employed in the context of automata theory. We establish the relationship of our reducibility to the DDH and some of its refinements, prove some basic facts about the corresponding degree structure and discover its close relationship to complexity theory (via the well-known leaf-language approach).

As we will see in Section 7 below, a natural question about our reducibility is closely related to the well-known open question on the decidability of the DDH. The proofs of this paper demonstrate deep interconnections of our reducibility with automata-theoretic and complexity-theoretic techniques. All this supports

* This paper was written during a research visit of the first author to the University of Würzburg supported by the Alexander von Humboldt Foundation.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 783–793, 2004.
© Springer-Verlag Berlin Heidelberg 2004

our belief that the reducibility provides a useful classification of regular star-free languages refining some popular hierarchies within this class.

We use (mostly without definitions here) some standard terminology and notation from automata theory and complexity theory, say the well-known notation of languages by regular expressions or the concept of a polynomial-time non-deterministic Turing machine. Letters *A, B, C* (sometimes with subscripts) will denote alphabets which are always assumed to contain at least two symbols. By A^+ we denote the set of all non-empty words over *A*, and by A^* the set of all words (including the empty word ε). Since we use the logical approach to (star-free) regular languages developed in [14,24] we work mostly with languages of non-empty words $L \subseteq A^+$. Correspondingly, the complement $\overline{L}$ of such a language *L* is defined by $\overline{L} = A^+ \setminus L$. For a class $\mathcal{K}$ of languages, let $\mathrm{BC}(\mathcal{K})$ be the boolean closure of $\mathcal{K}$, i.e., the closure of $\mathcal{K}$ under union and complement.

Relate to any alphabet $A = \{a, b, \ldots\}$ the signature $\sigma_A = \{\leq, Q_a, Q_b, \ldots, \bot, \top, p, s\}$, where $\leq$ is a binary relation symbol, Q_a (for any $a \in A$) is a unary relation symbol, $\bot$ and $\top$ are constant symbols, and p, s are unary function symbols. A word $u = u_0 \ldots u_n \in A^+$ may be considered as a structure $\mathbf{u} = (\{0, \ldots, n\}; <, Q_a, Q_b, \ldots)$ of the signature σ_A, where $<$ has its usual meaning, $Q_a (a \in A)$ are unary predicates on $\{0, \ldots, n\}$ defined by $Q_a(i) \leftrightarrow u_i = a$, the symbols $\bot$ and $\top$ denote the least and the greatest elements, while p and s are respectively the predecessor and successor functions on $\{0, \ldots, n\}$ (with $p(0) = 0$ and $s(n) = n$). For a sentence ϕ of σ_A, let $L_\phi = \{u \in A^+ | \mathbf{u} \models \phi\}$. In [14] it was shown that the class of all languages of the form L_ϕ, where ϕ ranges through first-order sentences of σ_A, coincides with the class of star-free languages. For $n > 0$, let Σ_n be the class of all languages L_ϕ, where ϕ ranges through the Σ_n-sentences of σ_A. In [24] it was shown that the sequence $\{\Sigma_n\}_{n>0}$ essentially coincides with the DDH. For this reason we call the sequence $\mathrm{DDH} = \{\Sigma_n\}$ here the *dot-depth hierarchy*. (Note that in the literature also the classes $\mathrm{BC}(\Sigma_n)$ belong to the dot depth hierarchy.) Let $\Pi_n = \mathrm{co}(\Sigma_n)$ denote the class of complements of Σ_n-languages, and $\Delta_n = \Sigma_n \cap \Pi_n$. When we want to stress that Σ_n is a level of the DDH over an alphabet *A* we may use the more exact notation $A\text{-}\Sigma_n$. We will do in the same way with other classes of languages.

Words correspond bijectively to (isomorphism types of) finite 'colored linear orderings', i.e. finite models of the theory CLO_A of signature σ_A with the following axioms:

- $<$ is a linear ordering,
- any element satisfies exactly one of the predicates $Q_a (a \in A)$,
- $\forall x(\bot \leq x \leq \top)$,
- $\forall x(p(x) \leq x \wedge \neg\exists y(p(x) < y < x))$,
- $\forall x(x \leq s(x) \wedge \neg\exists y(x < y < s(x)))$,
- $\forall x(x > \bot \rightarrow p(x) < x)$ and $\forall x(x < \top \rightarrow x < s(x))$.

Sometimes it is technically more convenient to consider 'relational' versions of the signature σ_A (and of the theory CLO_A). E.g., one could take sigature $\sigma'_A = \{\leq, Q_a, Q_b, \ldots, \bot, \top, S\}$, where $S(x, y)$ is a binary relation symbol interpreted as 'x is an immediate predecessor of y'. The both languages are clearly equivalent,

and we will use any of them when appropriate. More information on the logical approach to star-free languages maybe found in [24,18].

Due to the space bounds, we were unable to include proofs of our results in the conference version of our paper. They will appar in the journal version.

2 Definitions and Basic Properties

Let us recall a well-known logical notion of interpretation (see e.g. [13]); actually we use a particular case of this notion (namely, interpretation by quantifier-free formulas) which seems sufficient for our intentions here. The *qf-interpretation I over alphabets $A = \{a, \ldots\}$ and $B = \{b, \ldots\}$* is given by a tuple

$$(\phi_U(\bar{x}), \phi_<(\bar{x}, \bar{y}), \phi_b(\bar{x}), \ldots, \phi_\perp(\bar{x}), \phi_\top(\bar{x}), \phi_S(\bar{x}, \bar{y})),$$

where $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{y} = (y_1, \ldots, y_n)$ are sequences of different variables of the same length $n > 0$ (n is fixed in advance) and $\phi_U(\bar{x}), \ldots, \phi_S(\bar{x}, \bar{y})$ are quantifier-free formulas of σ_A with the following properties. Let $u = u_0 \cdots u_l$ be any word over A of length $|u| = l + 1$. Then the set $T = \{\bar{x} \in \{0, \ldots, l\}^n | \mathbf{u} \models \phi_U(\bar{x})\}$ should be non-empty and $\phi_<(\bar{x}, \bar{y}), \phi_b(\bar{x}), \ldots, \phi_\perp(\bar{x}), \phi_\top(\bar{x}), \phi_S(\bar{x}, \bar{y}))$ interpreted in $\mathbf{u}$ should define a model of CLO_B with the universe T (the formulas $\phi_\perp(\bar{x}), \phi_\top(\bar{x})$ should be true exactly on the first and the last element, respectively). Since finite models of CLO_B are in a bijective correspondence with elements of B^+, any qf-interpretation I induces a function $u \mapsto u^I$ from A^+ into B^+.

Examples. 1. Let $\phi_U(\bar{x})$ be a valid formula, let $\phi_<(\bar{x}, \bar{y})$ define the lexicographic ordering between $\bar{x}$ and $\bar{y}$, let $\phi_\perp(\bar{x}), \phi_\top(\bar{x}), \phi_S(\bar{x}, \bar{y})$ be defined in the obvious way according to their intended qf-interpretations, and let $\phi_b(\bar{x}, \ldots)$ be choosen in a way to get a model of CLO_B on this ordering. Then u^I is of length $|u|^n$, and letters of the word u^I are easily computed from the qf-interpretation.

2. Let $\phi_U(x)$ be a valid formula of one variable x, let $\phi_<(x, y)$ be $x > y$, let $\phi_a(x)$ be $Q_a(x)$ for any $a \in A$, let $\phi_\perp(\bar{x}), \phi_\top(\bar{x})$ be $x = \top$, $x = \perp$ respectively, and let $\phi_S(x, y)$ be $S(y, x)$. Then we get an qf-interpretation I over A and A such that u^I is the reverse of the word $u \in A^+$.

3. Let $u \mapsto pu$ be the function on A^+ which adds a fixed prefix $p \in A^*$ to a word u. Is there an qf-interpretation I over A and A such that $u^I = pu$ for any u? For $p = \varepsilon$ the answer is of course positive, otherwise it is negative (since any qf-interpretation sends words of length 1 to words of length 1). But it is easy to see that there is an qf-interpretation I such that $u^I = pu$ for any u of length > 1. The same of course applies to the operation of adding a suffix to a word.

4. Let $h : A^+ \to B^+$ be a semigroup morphism. Such functions are defined by their values $h : A \to B^+$ on the letters of A (i.e., words of length 1) because we have $h(a_0 \cdots a_l) = h(a_0) \cdots h(a_l)$, where $a_i \in A$. It is easy to see that for any such h there is an qf-interpretation I over A and B such that $u^I = h(u)$ for almost all $u \in A^+$ (i.e. for all but finitely many of words).

We are ready to give the main definition of this paper.

Definition 1. – *A function* $f : A^+ \to B^+$ *is called a qf-function if there is an qf-interpretation* I *over* A *and* B *such that* $u^I = f(u)$ *for almost all* $u \in A^+$.
– *Let* $L \subseteq A^+$ *and* $K \subseteq B^+$. *We say that* L *is qf-reducible to* K *(in symbols* $L \leq_m^{qf} K$*) if* $L = f^{-1}(K)$ *for some qf-function* $f : A^+ \to B^+$.

When we check some property of a qf-function $f(u)$ we usually for simplicity identify it with the corresponding function $u \mapsto u^I$. This is possible in the cases (which are mostly considered here) when the finite set of 'exceptional' words does not destroy the property under consideration.

We start with establishing an important technical property of $\leq_m^{qf}$.

Theorem 1. *The relation* $\leq_m^{qf}$ *is reflexive and transitive.*

Let $\mathcal{P}(A^+)$ be the power-set of A^+. The preorder $\leq_m^{qf}$ on $\mathcal{P}(A^+)$ induces in the usual way an equivalence relation on $\mathcal{P}(A^+)$ denoted by $\equiv_m^{qf}$. The corresponding quotient partial order is denoted $(\mathbf{D}_{qf}; \leq)$ and is called here *the structure of qf-degrees.* Our next goal is to get some information about this ordering. Let $\mathbf{D}'_{qf}$ consist of degrees of proper (or non-trivial) subsets of A^+.

The next theorem is parallel to corresponding properties of the structures of m-degrees and of polynomial-time m-degrees.

Theorem 2. *1.* $\{\emptyset\}$ *and* $\{A^+\}$ *are two distinct minimal elements of* $\mathbf{D}_{qf}$ *which are below any other element.*
2. The structure $(\mathbf{D}_{qf}; \leq)$ *is an upper semilattice where the supremum operation* $\cup$ *is induced by the join operation.*
3. The structure $(\mathbf{D}'_{qf}; \leq, \cup)$ *is a distributive upper semilattice with a least element which consists exactly of non-trivial* Δ_1*-languages.*

Below we will get additional information about qf-degrees which will help to understand the structure near the minimal elements. Here we will show that some natural classes of languages correspond to ideals of qf-degrees. Recall that an *ideal* of an upper semilattice is a nonempty downward closed set of its elements which is closed under supremum. An ideal is *principal* if it contains a biggest element. Let $\mathcal{R}$ and $\mathcal{SF}$ denote the class of regular and of regular star-free languages, respectively.

Theorem 3. *All the classes* $\Sigma_n, \Pi_n, \Delta_n (n > 0)$, *as well as the classes* $\mathcal{SF}$ *and* $\mathcal{R}$, *are ideals of* $(\mathbf{D}_{qf}; \leq)$.

Next we consider some refinements of the DDH introduced in [19]. Let $(B; \cup, \cap, \bar{}, 0, 1)$ be a boolean algebra (b.a.). Without loss of generality, one may think that B is class of subsets of some set. By *a base in* B we mean any sequence $\mathcal{L} = \{\mathcal{L}_n\}_{n<\omega}$ of sublattices of $(B; \cup, \cap, 0, 1)$, satisfying the inclusions $\mathcal{L}_n \cup \mathrm{co}(\mathcal{L}_n) \subseteq \mathcal{L}_{n+1}$ (here $\mathrm{co}(\mathcal{L}_n)$ denotes the dual set $\{\bar{x} | x \in \mathcal{L}_n\}$ for $\mathcal{L}_n$). With any base $\mathcal{L} = \{\mathcal{L}_n\}_{n<\omega}$ one can associate a family of new subsets of B as follows. Let T be the set of terms of signature $\{\cup, \cap, \bar{}, 0, 1\}$ with variables $v_k^n (k, n < \omega)$. We call $v_k^n (k < \omega)$ *variables of type* n, and elements of *T—typed boolean terms.* Relate to any term $t \in T$ the set $t(\mathcal{L})$ of all its values when variables of type

n range over $\mathcal{L}_n$. We call the family $\{t(\mathcal{L})\}_t$ *the typed boolean hierarchy over* $\mathcal{L}$. Note that $\mathrm{DDH} = \{\Sigma_{n+1}\}$ form a base, hence we may (and will) consider levels $t(\mathrm{DDH})$ of the typed boolean hierarchy over this base.

Corollary 1. *All levels of the typed boolean hierarchy over the DDH are ideals of* $(\mathbf{D}_{\mathrm{qf}}; \leq)$.

The most important particular case of the mentioned refinements is the difference hierarchy $\{\mathcal{D}_k(\Sigma_n)\}_{k<\omega}$ over any level Σ_n of the DDH. The k-th level $\mathcal{D}_k(\Sigma_n)$ of this hierarchy consists of languages of the form $\bigcup_i (L_{2i} \setminus L_{2i+1})$, where $L_0 \supseteq L_1 \supseteq \cdots$ is a descending sequence of Σ_n-languages and $L_i = \emptyset$ for $i \geq k$. For the case of difference hierarchy over Σ_1 we simplify notation of the levels to $\mathcal{D}_k$. As is well-known, $\mathcal{D}_k \cup \mathrm{co}(\mathcal{D}_k) \subseteq \mathcal{D}_{k+1}$, and similarly for the difference hierarchy over any level. The class $\mathrm{BC}(\Sigma_1) = \bigcup_k \mathcal{D}_k$ is well-known in automata theory as the class of languages *of dot-depth one.* The last corollary applies of course to the levels of the difference hierarchies.

In the next section we will show that some of ideals discussed above are indeed principal, i.e. contain complete sets w.r.t. qf-reducibility.

3 Complete Sets

First we show the existence of complete sets in the levels of the dot-depth hierarchy. The alphabets in this section are assumed to have at least two letters.

Theorem 4. *For any* $n > 0$ *and any alphabet B, the class* B-Σ_n *has a qf-complete set.*

Next we show that many levels of the typed boolean hierarchy also have complete sets.

Theorem 5. *For any alphabet A, all levels of the typed boolean hierarchy over* $\{A\text{-}\Sigma_n\}_{n\geq 2}$ *have qf-complete sets.*

The last theorem leaves open the question of existence of complete sets for some natural classes, say for some levels of the plus-hierarchy [16,19] over the DDH We guess that actually the complete sets the exist in any level of the typed boolean hierarchy over the DDH. As an important example, we show the existence of complete sets in levels of the difference hierarchy over Σ_1 (the last theorem implies of course that all levels of the difference hierarchy over any $\Sigma_n, n > 1$, have complete sets).

Theorem 6. *For any alphabet A and any* $k < \omega$, *the class* A-$\mathcal{D}_k$ *has a qf-complete set.*

A natural question is whether complete sets constructed above are reducible to their compliments. The analogy with computability theory [17] prompts the negative answer, the analogy with complexity theory envokes some doubts...

We can prove the corresponding result at least for the levels of the difference hierarchies. And actually this answer follows immediately from results of Section 2, results from [24,21,22] on the non-collapse of the difference hierarchies and from the existence of complete sets established above.

Corollary 2. *If C is qf-complete in some level of the difference hierarchy over $\Sigma_n, n > 0$, then $C \not\leq_{\mathrm{m}}^{\mathrm{qf}} \overline{C}$.*

We guess that the last result generalizes to all levels of the typed boolean hierarchy over the DDH.

Let D_k be the qf-complete language in the level $\mathcal{D}_k$ of the difference hierarchy over Σ_1, and let $\mathbf{d}_k, \bar{\mathbf{d}}_k$ be the qf-degrees of D_k and $\overline{D}_k$, respectively. From results of the last section it follows that $\mathbf{d}_k$ and $\bar{\mathbf{d}}_k$ are incomparable and both of them are below $\mathbf{d}_{k+1}$ and $\bar{\mathbf{d}}_{k+1}$.

We conclude this section with establishing the following relationship between these degrees which is similar to the corresponding property [16] of the difference hierarchy over recursively enumerable sets.

Theorem 7. *For any $k < \omega$, $\mathbf{d}_k \cup \bar{\mathbf{d}}_k$ is the maximum qf-degree in $\mathcal{D}_k \cap \mathrm{co}(\mathcal{D}_k)$.*

Theorem 7 implies that the difference hierarchy over Σ_1 is discrete, i.e., it has no refinement in any level (under a natural exact definition of the refinement [17]). But (as is also the case in computability theory) this does not exclude the existence of rich degree structures inside intervals like $(\bar{\mathbf{d}}_k, \mathbf{d}_{k+1})$. In Section 5 below we will show that such intervals are really rather complicated.

4 Leaf Languages and Polylogtime Reducibility

In this section we will see how the leaf language approach to complexity classes (well-known from literature) will help us to get new insights into the structure of qf-reducibility.

Let us recall some relevant definitions. Consider a polynomial-time nondeterministic Turing machine M working on an input word x over some alphabet B and printing a letter from another alphabet A after finishing any computation path. These values are the leaves of the binary tree defined by the nondeterministic choices of M on input x. An ordering of the tuples in the program of M determines a left-to-right ordering of all the leaves. In this way, M may be considered as a deterministic transducer computing a total function $M : B^* \to A^+$. Now, relate to any language $L \subseteq A^+$ (called in this situation a leaf language) the language $M^{-1}(L) \subseteq B^*$. Denote by $\mathrm{Leaf}_{\mathrm{b}}(L)$ the set of languages $M^{-1}(L)$, for all machines M specified above which have balanced computation trees, and denote by $\mathrm{Leaf}_{\mathrm{u}}(L)$ the set of languages $M^{-1}(L)$, for all machines M specified above (which may have unbalanced computation trees). For a set of languages $\mathcal{C}$, let $\mathrm{Leaf}_{\mathrm{b}}(\mathcal{C})$ be the union of $\mathrm{Leaf}_{\mathrm{b}}(L)$, for all $L \in \mathcal{C}$, analogously define $\mathrm{Leaf}_{\mathrm{u}}(\mathcal{C})$.

Obviously, we have $\mathrm{Leaf}_{\mathrm{b}}(L) \subseteq \mathrm{Leaf}_{\mathrm{u}}(L)$ for every language L, and there exist languages L where $\mathrm{Leaf}_{\mathrm{b}}(L) = \mathrm{Leaf}_{\mathrm{u}}(L)$ is unlikely. (For example, it is not hard

to see that $\mathrm{Leaf_b}((01)^*) = \mathrm{P}$ and $\mathrm{Leaf_u}((01)^*) = \oplus\mathrm{P}$.) However, $\mathrm{Leaf_b}(L) = \mathrm{Leaf_u}(L)$ is always true for *paddable languages* (we call a language $L \subseteq A^*$ paddable iff there exists an $a \in A$ such that $xay \in L \leftrightarrow xy \in L$ for all $x, y \in A^*$.

It turns out that many inportant complexity classes have natural and useful descriptions in terms of leaf languages (e.g. [2,26,11,12]).

Let us consider a special class of leaf languages namely the languages of finite counting type (cf. [9,10,6]). For a set $V \subseteq \mathbb{N}^k$ we define

$$L(V) =_{\mathrm{def}} \{x \in \{0,1,\ldots,k\}^+ : (\#_1(x),\ldots,\#_k(x)) \in V\}$$

where $\#_i(x)$ is the number of occurences of i in x. The set V and the language $L(V)$ are said to be *of finite counting type* iff there exists an $m \geq 0$ such that

$$(n_1,\ldots,n_k) \in V \Leftrightarrow (\min(n_1,m),\ldots,\min(n_k,m)) \in V.$$

It is obvious that the languages of finite counting type are aperiodic and hence starfree. It is easy to see that in fact any language of finite counting type is in $BC(\Sigma_1)$.

In [6] a very useful characterization of relativizable inclusions between complexity classes defined by leaf languages of finite counting type is proved. For $z = (z_1,\ldots,z_k) \in \mathbb{N}$ and $v = (v_1,\ldots,v_k) \in \mathbb{N}^k$ let $z \leq v$ iff $z_i \leq v_i$ for $i = 1,\ldots,k$, and let $\binom{v}{z} =_{\mathrm{def}} \prod_{i=1}^k \binom{v_i}{z_i}$.

Theorem 8. *[6] If $V \subseteq \mathbb{N}^k$ and $W \subseteq \mathbb{N}^l$ are of finite counting type then the following statements are equivalent:*

(1) $\mathrm{Leaf_b}(L(V))^{\mathcal{O}} \subseteq \mathrm{Leaf_b}(L(W))^{\mathcal{O}}$ *for every oracle* $\mathcal{O}$.

(2) *there exist functions* $\alpha_1,\ldots,\alpha_l : \mathbb{N}^k \to \mathbb{N}$ *such that*

$$v \in V \Leftrightarrow \left(\sum_{z\leq v} \alpha_1(z)\tbinom{v}{z},\ldots,\sum_{z\leq v} \alpha_l(z)\tbinom{v}{z}\right) \in W.$$

At this point another notion of reducibility comes into the play. A language $L \subseteq A^*$ is *polylogtime reducible* to $K \subseteq B^*$, for short $L \leq_{\mathrm{m}}^{\mathrm{plt}} K$, iff there exist functions $f : A^* \times \mathbb{N} \to B$ and $g : A^* \to \mathbb{N}$, computable in polylogarithmic time on a deterministic Turing machine with random access to the input bits, such that $x \in L \leftrightarrow f(x,1)f(x,2)\ldots f(x,g(x)) \in K$ for every $x \in A^*$. In the early 1990s, BOVET, CRESCENZI, SILVESTRI and VERESHCHAGIN proved the following ingenious result.

Theorem 9. *[2,26] The following are equivalent for languages L and K:*

(1) $L \leq_{\mathrm{m}}^{\mathrm{plt}} K$

(2) $\mathrm{Leaf_b}(L)^{\mathcal{O}} \subseteq \mathrm{Leaf_b}(K)^{\mathcal{O}}$ *for every oracle* $\mathcal{O}$.

Generally, $L \leq_{\mathrm{m}}^{\mathrm{qf}} K$ does not seem to imply $L \leq_{\mathrm{m}}^{\mathrm{plt}} K$, but for special cases this implication is valid.

Theorem 10. *Let $L \subseteq A^+$ and $V \subseteq \mathbb{N}^k$. Then $L \leq_{\mathrm{m}}^{\mathrm{qf}} L(V)$ implies $L \leq_{\mathrm{m}}^{\mathrm{plt}} L(V)$.*

The main theorem of this section gives two characterizations of the qf-reducibility between languages of finite counting type.

Theorem 11. *For languages $V \subseteq \mathbb{N}^k$ and $W \subseteq \mathbb{N}^l$ of finite counting type the following are equivalent:*

(1) $L(V) \leq_m^{qf} L(W)$
(2) $L(V) \leq_m^{plt} L(W)$
(3) *there exist an $r \in \mathbb{N}^k$ and functions $\alpha_1, \ldots, \alpha_l : \mathbb{N}^k \to \mathbb{N}$ such that*

$$v \in V \Leftrightarrow \left(\sum_{z \leq r} \alpha_1(z)\binom{v}{z}, \ldots, \sum_{z \leq r} \alpha_l(z)\binom{v}{z} \right) \in W.$$

Examples of applications of the preceding theorem are given in the next section.

5 Degrees Between $\overline{\mathbf{d}}_1$ and $\mathbf{d}_2$

In this section we will show that the degree structure between $\overline{\mathbf{d}}_1$ and $\mathbf{d}_2$ is sufficiently rich (we use the notation $D_k, \mathbf{d}_k$ from Section 3). For this reason we define for every $m \geq k \geq 0$ the language

$$U_{k,m} =_{\text{def}} \{x \in \{0,1\}^+ : \#_1(x) \in \{k, k+1, \ldots, m\}\}.$$

Furthermore, denote by $\mathbf{u}_{k,m}$ the qf-degree of $U_{k,m}$, for $k \geq m \geq 0$, and set $\mathbf{u} = \mathbf{u}_{1,1}$. The next theorem gives some information on the structure of the set $\{\mathbf{u}_{k,m} : m \geq k \geq 0\}$ of qf-degrees (see also Figure 2). Note that some of the results in this theorem can also be derived from results in [9] and [10] via Theorem 9 and the equivalence of 1 and 2 in Theorem 11. However, our proof via the equivalence of 1 and 3 in Theorem 11 is easier.

Theorem 12. *1.* $\overline{\mathbf{d}}_1 \leq \mathbf{u}_{k,m} \leq \mathbf{d}_2$ *for all* $m \geq k \geq 0$.
2. $\mathbf{u}_{0,m} = \overline{\mathbf{d}}_1$ *for all* $m \geq 0$.
3. $\mathbf{u}_{m,m} = \mathbf{u}$ *for all* $m \geq 1$.
4. $\overline{\mathbf{d}}_1 < \mathbf{u}$
5. $\mathbf{u}_{k,m} < \mathbf{u}_{k+1,m+1}$ *for all* $m > k \geq 1$.
6. $\mathbf{u}_{k,m} < \mathbf{u}_{k,m+1}$ *for all* $m \geq k \geq 1$.
7. $\mathbf{u}_{m-1,m} < \mathbf{u}_{1,m}$ *for all* $m \geq 3$.
8. $\mathbf{d}_1 \not\leq \mathbf{u}_{k,m}$ *for all* $m \geq k \geq 0$.
9. $\mathbf{u}_{k,m} \not\leq \overline{\mathbf{d}}_2$ *for all* $m \geq k \geq 1$.
10. $\mathbf{u}_{k,m} \not\leq \mathbf{u}_{l,m}$ *for all* $k, l \in \{1, 2, \ldots, m-2\}$ *such that* $k \neq l$ *and* $m \geq 4$.

Corollary 3. *Between $\overline{\mathbf{d}}_1$ and $\mathbf{d}_2$ there exist upward infinite chains and finite antichains of arbitrary length.*

6 The Principal Ideal Generated by $\mathbf{u} \cup \bar{\mathbf{u}}$

In this section we describe the principal ideal of $(\mathbf{D}_{qf}; \leq)$ generated by $\mathbf{u} \cup \bar{\mathbf{u}}$. We work here with the binary alphabet $B = \{0, 1\}$ and use notation from the last section. The next theorem follows rather easily from Theorem 2 and results in [1,15].

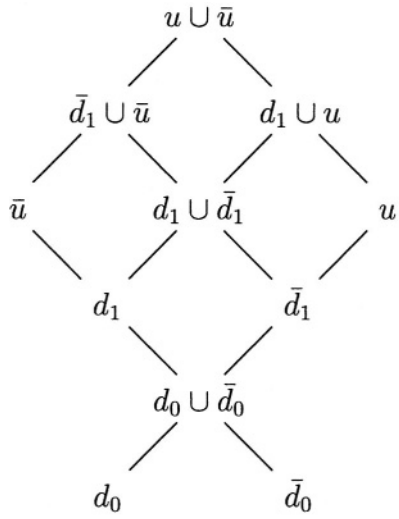


Fig. 1. The principal ideal generated by $\mathbf{u} \cup \bar{\mathbf{u}}$

Theorem 13. *The principal ideal* $\{\mathbf{x} | \mathbf{x} \leq \mathbf{u} \cup \bar{\mathbf{u}}\}$ *is shown in Figure 1.*

Theorem 13 shows that (at least near the bottom) the structure of qf-degrees is simpler than those of m-degrees and of polynomial-time m-degrees (though it is certainly more complicated that the structure of Wadge degrees). This support our belief that the qf-degrees are useful classifications of regular languages.

7 Future Work

There are many interesting open questions related to this paper. Some of them were already mentioned above. Here we mention some more, sometimes in the form of conjectures.

Conjecture 1. The relations $\leq_m^{plt}$ and $\leq_m^{qf}$ are decidable on the regular languages. This means (for $\leq_m^{qf}$) that there exists an algorithm which computes from finite automata recognizing languages L, M whether or not $L \leq_m^{qf} M$.

This conjecture seems very hard because (together with Theorems 3 and 4) implies the decidability of all levels of the DDH which is a well-known open question of automata theory. One could weaken the Conjecture 1 in different ways to get less hard problems, e.g.

Conjecture 2. The relations $\leq_m^{plt}$ and $\leq_m^{qf}$ are decidable on $\mathrm{BC}(\Sigma_1)$.

The last conjecture seems rather plausible (though not easy to prove!) since in [8,18] the decidability of several natural problems related to $\mathrm{BC}(\Sigma_1)$ was established.

Many natural questions on the introduced degree structures also remain open, e.g.:

Question. Is there an infinite antichain or an infinite descending chain within $(\mathrm{BC}(\Sigma_1); \leq_{\mathrm{m}}^{\mathrm{qf}})$?

The complete sets for the DDH and its refinements provide a useful benchmark to classify languages of 'practical' interest. E.g., one may wonder on estimating the languages corresponding to so called n-buffers [3,25]. This direction is parallel to estimation of 'practically' important sets in computability theory and complexity theory.

An interesting open problem related to Section 4 is the following: find a reducibility describing the relation $\forall \mathcal{O}(\mathrm{Leaf}_{\mathrm{u}}(L)^{\mathcal{O}} \subseteq \mathrm{Leaf}_{\mathrm{u}}(K)^{\mathcal{O}})$ in the same way as the polylogtime reducibility describes the corresponding relation for the balanced trees. We hope that our reducibility could be of use in solving this problem.

Several natural variants of the qf-reducibility are possible. E.g., one could consider the corresponding reducibility for the Straubing-Therien hierarchy by considering another signature in place of σ_A [24,18] (actually, in this context it is better to consider qf-interpretations by Δ_1-formulas). This reducibility does not induce the upper semilattice and is probably not so closely related to complexity theory as the one considered in this paper.

The notion of qf-reducibility seems rather flexible and applicable in other situations, say in the case of tree languages. It would be interesting to look at a similar development in that field.

Acknowledgement. We thank Christian Glaßer, Daniel Meister, and Bernhard Schwarz (Würzburg) for interesting and helpful discussions. We are also grateful to Erich Grädel, Wolfgang Thomas (Aachen) and Denis Thérien (Montreal) for useful remarks and bibliographical hints.

References

1. B. Borchert, D. Kuske and F. Stephan. On existentially first–order definable languages and their relation to *NP*. *Theor. Informatics Appl.,* 33 (1999), 259–269.
2. D.P. Bovet, P. Crescenzi and R. Silvestri. A uniform approach to define complexity classes. *Theoret. Comp. Sci,* 104 (1992), 263–283.
3. J.A. Brzozowski and R. Knast. The dot-depth hierarchy of star–free languages is infinite. *J. Comp. Systems Sci.,* 16 (1978), 37–55.
4. J.R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logic Grundl. Math.,* 6 (1960), 66–92.
5. R.S. Cohen and J.A. Brzozowski. Dot-depth of star-free events. *J. Comp. Systems Sci.* 5 (1971), 1–16.
6. K. Cronauer, U. Hertrampf, H. Vollmer and K.W. Wagner. The chain method to separate counting classes. *Theory Comput. Systems,* 31 (1998), 93–108.
7. H.-D. Ebinghaus and J. Flum. *Finite Model Theory,* Berlin, Springer, 1999.
8. C. Glaßer amd H. Schmitz. The Boolean Structure of Dot-Depth One. *J. of Automata, Languages and Combinatorics,* 6 (2001), 437–452.

9. T. Gundermann and G. Wechsung. Counting classes of finite accepting types. *Computers and artificial intelligence,* 6 (1987), 395–409.
10. T. Gundermann, N.A. Nasser and G. Wechsung. A survey on counting classes. *Proc. of Structures in Complexity Theory* (1990), 140–153.
11. U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer and K.W. Wagner. On the power of polynomial time bit-reductions, *Proc. 8th Structure in Complexity Theory,* 1993, 200–207.
12. U. Hertrampf, H. Vollmer and K.W. Wagner. On balanced vs. unbalanced computation trees, *Math. Systems Theory* 29 (1996), 411–421.
13. N. Immermann. *Descriptive Complexity.* Berlin, Springer, 1999.
14. R. McNaughton and S. Papert. *Counter–free automata.* MIT Press, Cambridge, Massachussets, 1971.
15. J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems,* 30 (1997), 383–422.
16. V.L. Selivanov. Two refinements of the polynomial hierarchy. *Lecture Notes in Computer Science,* v. 775. Springer, Berlin 1994, p. 439–448.
17. V.L. Selivanov. Fine hierarchies and Boolean terms. *J. Symbolic Logic,* 60 (1995), 289–317.
18. V.L. Selivanov. A logical approach to decidability of hierarchies of regular star-free languages. *Lecture Notes in Computer Science,* v. 2010. Berlin, Springer, 2001, 539–550
19. V.L. Selivanov. Relating automata-theoretic hierarchies to complexity-theoretic hierarchies. *Theoret. Informatics Appl.,* 36 (2002), 29–42.
20. J. Shoenfield. *Mathematical Logic,* Addison Wesley, Massachussets, 1967.
21. A.G. Shukin. Difference hierarchies of regular languages. *Comp. Systems,* Novosibirsk, 161 (1998), 141–155 (in Russian).
22. V.L. Selivanov and A.G. Shukin. On hierarchies of regular star-free languages (in Russian). Preprint 69 of A.P. Ershov Institute of Informatics Systems, 2000, 28 p.
23. H. Schmitz and K.W. Wagner. The Boolean hierarchy over level 1/2 of the Sraubing–Therien hierarchy, Technical Report 201, Inst. für Informatik, Univ. Würzburg, http://www.informatik.uni-wuerzburg.de.
24. W. Thomas. Classifying regular events in symbolic logic. *J. Comp. and Syst. Sci.,* 25 (1982), 360–376.
25. W. Thomas. A concatenation game and the dot-depth hierarchy. *Lecture Notes in Computer Science,* v. 270 (1987), 415–426.
26. N.K. Vereshchagin. Relativizable and non-relativizable theorems in the polynomial theory of algorithms. *Izvestiya Rossiiskoi Akademii Nauk,* 57 (1993), 51–90 (in Russian).
27. K.W. Wagner. A note on parallel queries and the symmetric-difference hierarchy. Information Processing Letters 66 (1998), 13–20.
28. G. Wechsung and K.W. Wagner. On the Boolean closure of NP, *Manuskript.* Extended abstract as: G. Wechsung, On the Boolean closure of NP, *Proceedings of the 1985 Int. Conf. on Fundamentals of Computation theory,* v.199 of *Lecture Notes in Computer Science,* p.485–493. Springer-Verlag, 1985.

Sublogarithmic Ambiguity

Klaus Wich

Institut für formale Methoden der Informatik, Universität Stuttgart,
Universitätsstr. 38, 70569 Stuttgart.
wich@informatik.uni-stuttgart.de

Abstract. So far the least growth rate known for a divergent inherent ambiguity function was logarithmic. This paper shows that for each computable divergent total non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ there is a context-free language L with a divergent inherent ambiguity function g below f. This proves that extremely slow growing divergent inherent ambiguity functions exist. For instance there is a context-free language L with infinite inherent ambiguity below log*.

1 Introduction

A context-free grammar G is unambiguous if it does not have two different derivation trees for any word. A context-free language is unambiguous if it is generated by an unambiguous context-free grammar. Context-free grammars and languages are ambiguous if they are not unambiguous. Ambiguous context-free languages are also called inherently ambiguous. The existence of ambiguous context-free languages is shown in [12]. Ambiguous context-free grammars and languages can be distinguished by their degree of ambiguity, that is, by the least upper bound for the number of derivation trees which a word can have. There are examples for k-ambiguous languages for each $k \in \mathbb{N}$ [9]. But even languages with infinite degree of ambiguity exist [14,5]. They can be distinguished by the asymptotic behaviour of their ambiguity with respect to the length of the words. There is an efficient algorithm which computes for each cycle-free context-free grammar G a constant k with the promise that the ambiguity of G is either $\mathcal{O}(n^k)$ or $2^{\Theta(n)}$ [17]. However which alternative is the case is undecidable [15]. Languages with ambiguity $2^{\Theta(n)}$ and with ambiguity $\Theta(n^k)$ for each $k \in \mathbb{N}$ are presented in [10].

The class of languages with polynomially bounded ambiguity ($PCFL$) has some interesting properties. For instance [17] shows that $PCFL$ is the closure of the class of unambiguous context-free language under a restricted form of projection. Combined with [13, Theorem 3] one easily obtains that each language $L \in PCFL$ can be recognised in $\mathcal{O}(\log n)$ time on a CREW PRAM. Polynomially bounded ambiguity is also important in the field of random generation [4].

Infinite sublinear ambiguity is interesting for two reasons: Firstly from the theoretical point of view it is interesting to know whether there is a gap between constant ambiguity and the "lowest" possible divergent ambiguity as is the case between polynomially bounded and exponential ambiguity. Secondly it

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 794–806, 2004.
© Springer-Verlag Berlin Heidelberg 2004

is well known that the parsing algorithm of Early [6,1] parses general context-free grammars in $\mathcal{O}(n^3)$ time while unambiguous context-free grammars are parsed in $\mathcal{O}(n^2)$ time. The proof which shows the speed up for unambiguous grammars can easily be generalised to $\mathcal{O}(n^2 \cdot \hat{d}_G(n + k_G))$ time for an arbitrary reduced context-free grammar G where $\hat{d}_G$ is the ambiguity function of G and $k_G \in \mathbb{N}$ is a constant only depending on G.[1] Thus Earley's algorithm parses languages with sublinear ambiguity faster than in cubic time. Languages with inherent square-root and logarithmic ambiguity can be found in [16], respectively. In [18] the following theorem has been shown:

Theorem 1. *The set of ambiguity functions for cycle-free context-free grammars and the set of inherent ambiguity functions coincide.*

Thus in order to prove the existence of infinite but sublogarithmic inherent ambiguity it suffices to present a cycle-free context-free grammar with the corresponding ambiguity.

It is well known that each recursively enumerable language is the homomorphic image of the intersection of two context-free languages [2,7,8]. In this paper we use similar ingredients. For each Turing machine M we construct context-free grammars generating languages of the form $(L\{\#\})^*F(\{\#\}R)^*$, where F is an unambiguous context-free subset of M's configurations, while L and R generate pairs of configurations separated by a "#" symbol, such that the right configuration is obtained in one step of M from the left one. Moreover exactly one of the configurations of such a pair is written in reverse. In case of L the right configuration is reversed, while for R it is the left one. It turns out that the ambiguity function of these grammars is dominated by the ambiguity of the words which lie in $F(\{\#\}R)^* \cap (L\{\#\})^*F$. A word w within this set represents a segment of a computation of the Turing machine M which starts and ends in a configuration belonging to F. Moreover the ambiguity of w is the number of times a configuration which belongs to F and which is preceded by an even number of configurations occurs in w. Since we are free in the choice of the underlying Turing machine M and the unambiguous context-free set F we have a strong tool to design very low divergent ambiguity functions. Roughly speaking, it is sufficient to find candidates for M and F, such that computations of M containing many configurations in F cannot be to short.

2 Preliminaries

If not stated otherwise Σ is an arbitrary finite non-empty alphabet in the sequel. The symbol # is not in Σ. The empty word is denoted by ε. For $w \in \Sigma^*$ the *reversal* w^R of w is defined by $\varepsilon^R = \varepsilon$ and $(au)^R = u^R a$ for $u \in \Sigma^*$ and $a \in \Sigma$. The length of a word $w \in \Sigma^*$ is denoted by $|w|$. For all $n \in \mathbb{N}$ we define $\Sigma^n := \{w \in \Sigma^* \mid |w| = n\}$, $\Sigma^{\leq n} := \cup_{i \in \{0,\ldots,n\}} \Sigma^i$, and $\Sigma^{<n} := \Sigma^{\leq n} \setminus \Sigma^n$. A

[1] The constant k_G can be omitted if $\exists c \in \mathbb{N} : \forall n \in \mathbb{N} : \hat{d}_G(n+1) \leq c \cdot \hat{d}_G(n)$. An ambiguity function which violates this condition is not known.

context-free grammar is a quadruple $G = (N, \Sigma, P, S)$ where N and Σ are two disjoint alphabets of *nonterminals* and *terminals,* respectively, $P \subseteq N \times (N \cup \Sigma)^*$ is a finite set of *productions,* and $S \in N$ is the *start symbol.* The reader is assumed to be familiar with derivation trees and the definition of the language generated by G as defined in [8]. A context-free grammar G is *cycle-free* if a nonterminal cannot reproduce itself in a non-void derivation.

Let $G = (N, \Sigma, P, S)$ be a context-free grammar. The ambiguity of a word w is its number of derivation trees. The *ambiguity power series* of G is a function $d_G : \Sigma^* \to \mathbb{N}$ which maps each word to its ambiguity. This function is not well defined if a single word has infinite ambiguity. Let $G = (N, \Sigma, P, S)$ be a context-free grammar with a well defined ambiguity power series. The *ambiguity function* $\hat{d}_G : \mathbb{N} \to \mathbb{N}$ of G is $\hat{d}_G(n) = \max\{d_G(w) \mid w \in \Sigma^{\leq n}\}$, i.e., it maps each $n \in \mathbb{N}$ to the ambiguity of the most ambiguous word of length up to n. The grammar G is *unambiguous* if its ambiguity function is bounded by 1.

The grammar G is *reduced* if it does not contain useless symbols, i.e., symbols which does not occur in any derivation tree. Useless symbols can easily be eliminated. This elimination preserves the ambiguity power series. Therefore we do not need to consider non-reduced context-free grammars. It is easily seen that a reduced context-free grammar has a well defined ambiguity power series if and only if it is cycle-free.

A context-free language is unambiguous if it is generated by some unambiguous context-free grammar. It is ambiguous otherwise. For each context-free language L and each $n \in \mathbb{N}$ there is a context-free grammar G such that all the words of length up to n are generated unambiguously, i.e., $\hat{d}_G(n) \leq 1$. In case L is ambiguous to increase n one has to switch to context-free grammars with larger and larger pumping constants. But for an ambiguous language it may be possible to specify the least length of a word with a given ambiguity up to a constant depending only on the the pumping constant of the used context-free grammar. This leads us to the following definition taken from [18]:

Definition 1. *Let L be a context-free language and $f : \mathbb{N} \to \mathbb{N}$ a function. The language L is* f-ambiguous *if*

1. *there is a context-free grammar G such that $L = L(G)$ and $f = \hat{d}_G$ and*
2. *for each context-free grammar G such that $L = L(G)$ there exists a $c \in \mathbb{N}$ such that $f(n) \leq \hat{d}_G(cn)$ for all $n \in \mathbb{N} \setminus \{0\}$.*

A function $f : \mathbb{N} \to \mathbb{N}$ is an *inherent ambiguity function* if there is a context-free language L such that L is f-ambiguous. If L is f-ambiguous then L is f'-ambiguous for all non-decreasing functions f' such that f' agrees with f for all but a finite number of arguments. Note that the question whether each context-free language has an inherent ambiguity function is open. Now we have introduced all the notions needed to properly understand Theorem 1, which occurred in the introduction.

The reader is assumed to be familiar with single tape Turing machines as defined in [8]. Whenever we refer to a Turing machine in this paper we mean this type. Whether the Turing machine is deterministic or nondeterministic or

whether the tape is single or both sided infinite does not matter here. But it may help to think of deterministic machines. A *configuration* of a Turing machine consists of the tape content, the state of the Turing machine and the position of the head. It is denoted by a word consisting of the shortest string which represents the coherent portion of the tape which covers all the non blank cells and the position of the tape head. The head is denoted immediately to the left of the tape cell the machine reads in the next step. For each Turing machine M the corresponding set of configurations is denoted by ID_M. The relation $\vdash_M$ contains all the pairs of configurations $(id_a, id_b) \in ID_M \times ID_M$ where id_b is obtained from id_a by a single step of M.

3 Block Correlation Languages

Our aim is to find for each computable divergent non-decreasing function f an inherent ambiguity function g which falls below f. As a tool to design divergent ambiguity functions with a very slow growth rate we introduce block correlation languages.

Definition 2. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation. Then*

$$\ell(R) := \{u\#v^R \mid (u,v) \in R\} \text{ and } r(R) := \{u^R\#v \mid (u,v) \in R\}.$$

The relation R is (unambiguous) context-free *if $\ell(R)$ and $r(R)$ are both (unambiguous) context-free languages.*

It can be shown that $\ell(R)$ is (unambiguous) context-free if and only if $r(R)$ is (unambiguous) context-free. But instead to prove this statement for arbitrary R it is easier to check both languages for the relations considered below.

Definition 3. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be an unambiguous context-free relation, and $F \subseteq \Sigma^*$ an unambiguous context-free language. The* block correlation language *over the relation R and the set F is defined by:*

$$L(R,F) := \big(\ell(R)\{\#\}\big)^* F \big(\{\#\} r(R)\big)^*.$$

If L(R, F) is a block-correlation language then F is called the corresponding language of free blocks.

Definition 4. *A* canonical grammar *for a block correlation language L(R, F) over a relation $R \subseteq \Sigma^* \times \Sigma^*$ and a set $F \subseteq \Sigma^*$ is a context-free grammar:*

$$G := \big(\{S, A\} \dot\cup N_\ell \dot\cup N_r \dot\cup N_F,\ \Sigma \cup \{\#\},\ P \cup P_\ell \cup P_r \cup P_F,\ S\big).$$

where $G_\ell := (N_\ell, \Sigma \cup \{\#\}, P_\ell, S_\ell)$, $G_r := (N_r, \Sigma \cup \{\#\}, P_r, S_r)$, and $G_F := (N_F, \Sigma, P_F, S_F)$ are unambiguous context-free grammars generating $\ell(R)$, $r(R)$, and F, respectively. Moreover P is defined by:

$$P := \{S \to S_\ell\#S,\ S \to S_F A,\ A \to A\#S_r,\ A \to \varepsilon\}.$$

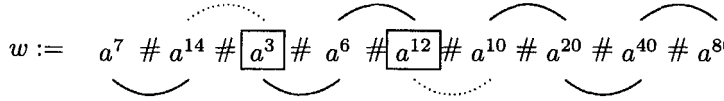
Note that G_ℓ, G_r, and G_F have pairwise disjoint sets of nonterminals. Moreover the nonterminal sets does not contain the symbols S and A, respectively. This is expressed by the dot on the union symbols.

Let G be a canonical grammar which generates a block correlation language $L(R,F)$ with all the sets and symbols named as above and let $G' := (\{S, A\}, \{S_\ell, S_r, S_F, \#\}, P, S)$. Obviously the grammar G' is unambiguous and generates the regular language $(S_\ell\#)^* S_F (\#S_r)^*$. A given derivation tree ρ of G generating a word w can always be trimmed in a unique way to obtain a derivation tree ρ' of G'. Let α be the frontier of ρ'. If we know α we can retrieve ρ' since G' is unambiguous. Moreover, except for the first and last symbol, each occurrence of S_ℓ, S_r and S_F in α is immediately preceded and followed by a "#" symbol. Furthermore each string of terminals generated by S_ℓ or S_r contains exactly one "#" symbol and a terminal string generated by S_F never generates a "#" symbol. Therefore each occurrence of the symbols S_ℓ, S_r and S_F in α can be uniquely matched with the infix of w it generates. This is sufficient to complete the remainder of ρ uniquely since the grammars G_ℓ, G_r, and G_F are unambiguous. Thus if we know w and α we can uniquely retrieve the whole derivation tree ρ. But this does not mean that G is necessarily unambiguous since w does not determine α in general. We can only deduce the length of $\alpha \in (S_\ell\#)^* S_F (\#S_r)^*$, but there may be several permissible position for S_F. We consider the "#" symbols as markers factorising w into blocks. Thus S_F generates exactly one block called the *free block* in the sequel. The free block is preceded and followed by strings of the form $\big(\ell(R)\{\#\}\big)^*$ and $\big(\{\#\}r(R)\big)^*$, respectively. (In particular this implies that the free block is preceded and followed by an even number of blocks.) The number of derivation trees for w coincides with the number of decompositions of w satisfying the requirements stated above. The result of our discussion is summarised in the following lemma:

Lemma 1. *Let G be a canonical grammar generating a block correlation language $L(R, F)$ over a relation $R \subseteq \Sigma^* \times \Sigma^*$ and a set $F \subseteq \Sigma^*$, and let $w \in \big(\Sigma \cup \{\#\}\big)^*$. Then the number of derivation trees for w is:*

$$d_G(w) = |\{i \in \mathbb{N} \mid w \in \big(\ell(R)\{\#\}\big)^i F \big(\{\#\}r(R)\big)^*\}|.$$

Example 1. Let $\Sigma = \{a\}$ and $R := \{(a^i, a^{2i}) \mid i \in \mathbb{N}\}$. Here $\ell(R) = r(R)$ since Σ is unary. To compute the ambiguity of a word w in a canonical grammar for the block correlation language $L(R, \Sigma^*)$ we consider each pair of consecutive blocks in w separated by a "#" symbol. From left to right we draw alternating arcs below and above consecutive pairs of blocks starting with an arc below the leftmost pair. An arc is drawn with a solid line if the pair is in relation, i.e., the number of a's in the right block is twice the number of a's in the left one. Otherwise the arc is dotted. Let us consider the following word:



By definition the free block is preceded and followed by an even number of blocks. Such a block is a candidate for the free block if all the arcs below the word to its left and all the arcs above the word to its right are solid. These criteria are satisfied for exactly those blocks of w written in boxes. Therefore the word w has exactly two derivation trees for any canonical grammar generating $L(R, \Sigma^*)$.

Note that a canonical grammar generating a block correlation language is not always the least ambiguous grammar generating it. For instance consider the unary alphabet $\Sigma = \{a\}$ and the unambiguous context-free relation $R := \{(a^i, a) \mid i \in \mathbb{N}\}$. Then it is easily seen that $L(R, \Sigma^*) = (a^*\#a\#)^*a^*(\#a^*\#a)^*$, which is regular, and therefore unambiguous context-free. Despite that, for each canonical grammar generating $L(R, \Sigma^*)$ and for each $i \in \mathbb{N}$ the word $(a\#)^{2^i}a$ has $i + 1$ derivation trees.

Definition 5. *Let G be a context-free grammar over Σ. Then the* support *of $\hat{d}_G$ is the set:*

$$support_G := \{w \in L(G) \mid \forall u \in \Sigma^{<|w|} : d_G(u) < d_G(w)\}$$

Thus a word w is in the support of the ambiguity function of a context-free grammar G if it is a shortest word with ambiguity at least $d_G(w)$. To determine the ambiguity function $\hat{d}_G$ of G it is sufficient to consider the words in $support_G$ and their corresponding ambiguities. More precisely the ambiguity function $\hat{d}_G$ is uniquely determined by the set:

$$\{(|w|, d_G(w)) \in \mathbb{N} \times \mathbb{N} \mid w \in support(G)\}.$$

But how do the words in the support of a canonical grammar for a block correlation language look like? It turns out to be necessary for them that each pair of consecutive blocks is correlated. In the notation of Example 1 this means that a word in the support of a canonical grammar never has a "dotted" arc connecting consecutive blocks. Before showing this formally, we define:

Definition 6. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation and $F \subseteq \Sigma^*$ a formal language. Then*

$$val(R, F) := \left\{ w_0\#w_1^R\#\cdots\#w_{2n} \;\middle|\; \begin{array}{l} w_0, w_{2n} \in F \wedge \\ \forall i \in \{0, \ldots, 2n-1\} : (w_i, w_{i+1}) \in R \end{array} \right\}.$$

It is easily seen that $val(R, F) = \big(\ell(R)\{\#\}\big)^* F \cap F\big(\{\#\}r(R)\big)^*$ for each relation R and each language F over Σ.

Theorem 2. *Let G be a canonical grammar of some block correlation language $L(R, F)$. Then*

$$support_G \subseteq val(R, F).$$

Proof. Let $w \in L(R, F) \setminus val(R, F)$. By definition $w \in \big(\ell(R)\{\#\}\big)^* F\big(\{\#\}r(R)\big)^*$. Since $val(R, F) = \big(\ell(R)\{\#\}\big)^* F \cap F\big(\{\#\}r(R)\big)^*$ we know that $w \notin \big(\ell(R)\{\#\}\big)^* F$ or $w \notin F\big(\{\#\}r(R)\big)^*$. If $w \notin F\big(\{\#\}r(R)\big)^*$ then $w \in \big(\ell(R)\{\#\}\big)^+ F\big(\{\#\}r(R)\big)^*$.

But then cancellation of the first two blocks in w yields a shorter word $w' \in L(R,F)$. Moreover the number of blocks of w' which belong to F and which are preceded by an even number of blocks equals the number of blocks in w with these properties. According to Lemma 1 this implies $d_G(w') = d_G(w)$. Thus w is not in the support of d_G. Analogously if $w \notin (\ell(R)\{\#\})^* F$ we can cancel the last two blocks to obtain a shorter word w' with the same ambiguity as w which implies that w is not in the support of d_G in this case either. Thus $(L(R,F) \setminus val(R,F)) \cap support_G = \emptyset$. But $support_G \subseteq L(R,F)$. Hence $support_G \subseteq val(R,F)$. □

Note that in the proof above we do neither state that the word w' obtained from the cancellation of a block pair is in $support_G$ nor that it is in $val(R, F)$. But since $w' \in L(R,F)$ we can iterate the cancellation of block pairs either from left or right until eventually a word in $val(R, F)$ with the same ambiguity as the original word is reached. (For the word w in Example 1 this procedure would yield $a^3\#a^6\#a^{12}$.) Since w has a finite number of blocks such an iteration terminates.

If we apply Theorem 2 to the language $L(R, \Sigma^*)$ of Example 1 we see that the support of each canonical grammar G for this language only contains words where the number of a's is doubled from block to block. That is the shortest word with ambiguity $i+1$ is $a^{2^0}\#a^{2^1}\#\cdots\#a^{2^{2i}}$ for an arbitrary $i \in \mathbb{N}$. Since the length grows exponentially with the ambiguity we see that $\hat{d}_G$ is logarithmic for each canonical context-free grammar generating $L(R, \Sigma^*)$. It can be shown that $L(R, \Sigma^*)$ is even inherently ambiguous of logarithmic degree. A proof for the existence of an inherent logarithmic ambiguity function can be found in [16], actually for a very similar language. The result there is slightly stronger, since by a reordering of the blocks a *linear* context-free language could be achieved. The reordering used there resembles the one in [2] to show that recursively enumerable sets can be characterised by the homomorphic image of the intersection of two linear context-free languages. Moreover by the example in [16] it can be shown that there is a regular trace language with a logarithmic ambiguity degree, which has been observed in [3].

How can we get a divergent ambiguity function with a sublogarithmic growth rate? One trial may be to force an even stronger growth of the length of related blocks. But this approach doesn't work as the following lemma shows:

Lemma 2. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation and $\ell(R)$ a context-free language with the pumping-constant n. Then*

$$\forall x, y \in \Sigma^* : \Big(\big((x,y) \in R \wedge \forall z \in \Sigma^{<|y|} : (x,z) \notin R \big) \Rightarrow n(|x|+1) \geq |y| \Big).$$

Proof. We prove this statement by induction on $|y|$. For $|y| \leq n$ it is trivial. Now assume for some $m \leq n$ the statement holds for all $y \in \Sigma^{\leq m}$. Let $y \in \Sigma^{m+1}$. For each $x \in \Sigma^*$ we have to check the implication stated above. The nontrivial case is the one where the left-hand side of the statement is satisfied. In this case $x\#y^R \in \ell(R)$ and we can mark the rightmost $|y|$ symbols of this word. According to Ogden's Lemma [11,8] we can pump down $x\#y^R$ into a word of

the form $x'\#y' \in \ell(R)$. Now $|y'| < |y| \leq |y'| + n$. Due to the minimality of y we obtain that $|x'| + 1 \leq |x|$. Hence we finally get:

$$|y| = |y| - |y'| + |y'| \leq n + |y'| \leq n + n(|x'| + 1) \leq n + n|x| = n(|x| + 1).$$

□

Since $2|x| \geq |x| + 1$ for $x \neq \varepsilon$ Lemma 2 immediately implies:

$$\forall x, y \in \Sigma^+ : \Big(\big((x, y) \in R \wedge \forall z \in \Sigma^{<|y|} : (x, z) \notin R\big) \Rightarrow 2n|x| \geq |y|\Big).$$

Hence we cannot force consecutive blocks to grow faster than by a constant factor, except for the very first step. Therefore sublogarithmic ambiguity cannot be obtained by this method. Obviously we can prove a version of Lemma 2 with $\ell(R)$ replaced by $r(R)$ in an analogous way.

4 Valid Computations

In example 1 the language for the free blocks is Σ^*. Therefore no candidate for the free block can be excluded in this case. As we have seen there is no hope to achieve sublogarithmic ambiguity just by increasing the growth rate of the blocks any further.

The new idea is to find an unambiguous context-free relation R and an unambiguous context-free language F such that in an infinite chain of words $w_0, w_1, \ldots$ such that $(w_i, w_{i+1}) \in R$ for each $i \in \mathbb{N}$ there are infinitely many words with even index belonging to the free block language F. But with rising index the blocks in F occur less frequent.

Let M be a Turing machine. For each $F \subseteq ID_M$ the words in $val(\vdash_M, F)$ represent computations which start and end in configurations belonging to F. It is easily seen that $\vdash_M$ is an unambiguous context-free relation for each Turing machine M (even if M is nondeterministic). In fact $\ell(\vdash_M)$ and $r(\vdash_M)$ are even deterministic and linear context-free languages. Therefore by application of Theorem 2 we obtain:

Corollary 1. *Let M be a Turing machine and let $F \subseteq ID_M$ be an unambiguous context-free language. Then $L(\vdash_M, F)$ is a block correlation language. Moreover if G is a canonical grammar generating $L(\vdash_M, F)$ then*

$$support_G \subseteq val(\vdash_M, F).$$

Even though $L(\vdash_M, F)$ is a large superset of $val(\vdash_M, F)$ we don't need to care for the words in $L(\vdash_M, F) \setminus val(\vdash_M, F)$ since they don't contribute to the ambiguity function of G. Therefore Corollary 1 provides a strong tool to design ambiguity functions. For instance let M be a Turing machine and $F \subseteq ID_M$ the set of configurations of M where M is in the initial state. Let G be a canonical grammar generating $L(\vdash_M, F)$. Then only the words in $val(\vdash_M, F)$ are relevant

for the computation of the ambiguity function, i.e., the words representing computations which start and end in configurations containing the initial state of M. The ambiguity of such a word is just the number of occurrences of the initial state in w at positions preceded by an even number of configurations. By the use of F we can induce an additional unambiguous context-free constraint on the initial configuration.

5 The Design of Slow Divergent Ambiguity Functions

Now we construct suitable Turing machines by the use of Corollary 1.

Lemma 3. *Let $f : \mathbb{N} \to \mathbb{N}$ be a computable divergent total non-decreasing function Then there is a Turing machine M and an unambiguous context-free set $F \subseteq ID_M$ with the following properties: For each $n \in \mathbb{N} \setminus \{0, 1\}$ their is a word $w \in val(\vdash_M, F)$ which contains n occurrences of configurations in F and the shortest word with this property has a length of at least $f(n)$. Moreover each occurrence of a configuration in F is preceded by an even number of configurations.*

Proof. Let $g : \mathbb{N} \to \mathbb{N}$ be defined by $g(n) := f(n + 2)$ for each $n \in \mathbb{N}$. Then g is obviously a computable divergent total non-decreasing function. Let M' be a Turing machine which computes g. Without loss of generality we assume that {0,1} is the input alphabet of M' and that non negative integers are encoded binary. Let q_0 be the initial state of M. We further assume that the state sets of M and M' are disjoint. The set F contains all the configurations of the form $q_0 q_0' n \$ n^R$. Here \$ is a tape symbol of M which is not a tape symbol of M' and q_0' is the initial state of M' and at the same time a tape symbol of M. Finally n is a binary encoded non negative integer. For convenience we identify binary encodings and the corresponding non-negative integers. Note that F is an unambiguous context-free language. The machine M never corrupts the initial format, i.e., if $\tau \in ID_M$ is reached from a configuration in F then by erasing the state of M from τ we obtain a string of the form $u_1 \$ u_2 \in ID_{M'} \{\$\} \mathbb{N}$, here $\mathbb{N}$ means the set of binary encodings of non negative integers. We refer to u_1 and u_2 by calling them the *first* or *second segment* of τ. Note that u_1 contains a state of M'. The string obtained from u_1 by erasing this state is called the *tape* of the first segment. Let M be in the configuration $q_0 q_0' n_1 \$ n_1^R \in F$. Then M goes through the following infinite loop:

1. Switch to a state $q_1 \neq q_o$ which starts the simulation of M'.
2. Simulate M' on the first segment until it halts. This eventually happens since the function g is computed by M' is total.
3. Decrement the tape of the first segment stepwise until it is 0. This is an idle loop (which loops $g(n_1)$ times when step 3 is called for the first time.)
4. Increment the second segment in its reverse coding (which yield $(n_1 + 1)^R$ when step 4 is called for the first time.)
5. Overwrite the first segment by $q_0' n$, where n is the reversal of the second segment, ($q_0' n = q_0'(n_1 + 1)$ when step 5 is called for the first time), place

the head at the position of q_0'. If the last time the machine M was in q_0 is an odd number of steps ago enter q_0 immediately, otherwise wait one step before entering q_0. The parity of a step can easily be stored within the finite control of the Turing machine M. This action returns M into the initial situation. Thus it performs a kind of "**goto** 1" command.

Moreover we require that the Turing machine M is programmed in such a way that it does not enter the state q_0 except for the cases where this is explicitly mentioned above.

If a word $w \in val(\vdash_M, F)$ contains n occurrences of configurations in F for some $n \in \mathbb{N} \setminus \{0,1\}$ then steps 1 to 5 have been called at least $n-1$ times each. The value computed in the last call of step 2 was $g(n_1 + n - 2)$, where n_1 is the argument for which g is computed the first time. Since g is non-decreasing we have $g(n_1 + n - 2) \geq g(n-2) = f(n)$, and the machine needs at least $g(n_1 + n - 2) \geq f(n)$ steps in the idle loop executed in the last call of point 3 which is denoted in w. Hence w contain at least $f(n)$ many configuration each of which requires at least one symbol to be denoted.

Finally since the first configuration of a word in $val(\vdash_M, F)$ is in F and M always makes an even number of steps before reentering a configuration in F each of these configurations are preceded by an even number of configurations. This completes the argument. □

Note that M started on a configuration in F runs forever and passes an infinite number of times through a configuration in F. The set $val(\vdash_M, F)$ contains finite infixes of infinite runs of M.

The estimation in the previous proof is rather wasteful, but simple to understand. Since we are not looking for a result on the density of ambiguity functions here, we can afford to use such a rough estimation.

Theorem 3. *Let $f : \mathbb{N} \to \mathbb{N}$ be a computable divergent total non-decreasing function then there is a context-free grammar such that for each $n \in \mathbb{N} \setminus \{0,1\}$ a shortest word with at least n derivation tree has length at least $f(n)$.*

Proof. By Lemma 3 there is a Turing machine M and an unambiguous context-free language F such that a shortest word in $val(\vdash_M, F)$ with n occurrences of a configuration in F has length at least $f(n)$ for $n \in \mathbb{N} \setminus \{0,1\}$. Moreover each of these occurrences is preceded by an even number of configurations. Therefore according to Lemma 1 these words have n derivation trees in a canonical grammar generating $L(\vdash_M, F)$ and by Corollary 1 we do not need to consider other words in $L(\vdash_M, F)$. □

Now we can guarantee that the length of a shortest word with ambiguity $n \in \mathbb{N} \setminus \{0,1\}$ is larger than $f(n)$ where f belongs to a set of function which allow huge growth rates. While we have considered the word length as a function of the ambiguity, the ambiguity function considers ambiguity as a function of the word length. Thus if the shortest word with ambiguity at least n exceeds a length of $f(n)$ for a given grammar G then roughly speaking $\hat{d}_G$ falls below f^{-1}. But we

have to take care of non injective functions which technically does not have an inverse. Therefore we define:

Definition 7. *Let* $f : \mathbb{N} \to \mathbb{N}$ *be a divergent total non-decreasing function. Then:*

$$f^{-1}(n) := \max\{x \in \mathbb{N} \mid f(x) = \min\{y \in f(\mathbb{N}) \mid y \geq n\}\}.$$

It can be easily shown that $(f^{-1})^{-1} = f$ holds for each divergent total non-decreasing function. Now let f be a computable divergent total non-decreasing function. Then f^{-1} is as well a computable divergent total non-decreasing function. Thus according to Theorem 3 there is a context-free grammar G such that for each $n \in \mathbb{N} \setminus \{0,1\}$ a shortest word with ambiguity of at least n has length at least $f^{-1}(n)$. But one can easily verify that this translates into $\hat{d}_G(n) \leq (f^{-1})^{-1}(n) = f(n)$ for all $n \geq f^{-1}(1)$. Now it is possible to construct a context-free grammar G' for $L(G) \setminus \Sigma^{\leq f^{-1}(1)}$ from G such that $\hat{d}_{G'}(n) = \hat{d}_G(n)$ for all $n > f^{-1}(1)$. Obviously $\hat{d}_{G'}(n) = 0$ for each $n \leq f^{-1}(1)$. Therefore $\hat{d}_{G'}(n) \leq f(n)$ for each $n \in \mathbb{N}$. Obviously G' has a well defined ambiguity power series. Moreover we can assume that G' is reduced. Hence G' is cycle-free. This leads us to:

Theorem 4. *If* f *is a computable divergent total non-decreasing function then there is a cycle-free context-free grammar* G *such that* $\hat{d}_G$ *is a divergent function satisfying* $\hat{d}_G(n) \leq f(n)$ *for all* $n \in \mathbb{N}$.

By the use of Theorem 1 and Theorem 4 we immediately obtain:

Theorem 5. *If* f *is a computable divergent total non-decreasing function then there is a context-free language* L *such that* L *has a divergent inherent ambiguity function* $\hat{d}_L$ *such that* $\hat{d}_L(n) \leq f(n)$ *for all* $n \in \mathbb{N}$.

6 Conclusion

We have seen that for each computable divergent total non-decreasing function there is a divergent inherent ambiguity functions which fall below f. We have not examined which functions are indeed ambiguity functions. Seemingly there are no substantial gaps below linear ambiguity, in contrast to the gap between exponential and polynomially bounded ambiguity. But how can we characterise the "density" of ambiguity functions formally? To examine this question one should improve the estimation in this paper. There is no need to use single steps of Turing machines as a means of computation. Instead we can allow unambiguous context-free relations to perform single steps. Clearly for the computational power this is unimportant but it provides more control over the length of the computations, which is crucial to control the ambiguity in our construction.

A characterisation of the set of inherent ambiguity functions is still a challenging problem, not only for sublogarithmic, but also for the whole class of context-free languages with polynomially bounded ambiguity.

By the result of this paper it is obvious that for each context-free grammar G_1 with a divergent ambiguity function we can find a not necessarily equivalent context-free grammar G_2 with a substantially lower ambiguity function. Substantially lower here means that for any $c \in \mathbb{N}$ we have $d_{G_1}(n) \geq d_{G_2}(cn)$ for all but finitely many $n \in \mathbb{N}$. Is there a context-free language L such that a similar property holds for all the context-free grammars generating L? In this case L would not have an inherent ambiguity function. Are there in fact context-free languages which does not have an inherent ambiguity function?

References

1. A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation, and Compiling,* volume I. Prentice-Hall, Englewood Cliffs, N.J., 1972.
2. B. S. Baker and R. V. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences,* 8:315–332, 1974.
3. A. Bertoni, C. Choffrut, M. Goldwurm, and V. Lonati. On the number of occurrences of a symbol in words of regular languages. *Theoretical Computer Science,* 302:431–456, 2002.
4. A. Bertoni, M. Goldwurm, and M. Santini. Random generation and approximate counting of ambiguously described combinatorial structures. In H. Reichel and S. Tison, editors, *Proceedings of the STACS 2000,* number 1770 in Lecture Notes in Computer Science, pages 567–580, Berlin-Heidelberg-New York, 2000. Springer.
5. J. Crestin. Un langage non ambigu dont le carré est d'ambiguité non bornée. In M. Nivat, editor, *Automata, Languages and Programming,* pages 377–390. Amsterdam, North-Holland, 1973.
6. J. C. Earley. *An efficient context-free parsing algorithm.* PhD thesis, Carnegie-Mellon University, 1968.
7. S. Ginsburg, S. A. Greibach, and M. A. Harrison. One-way stack automata. *Journal of the Association for Computing Machinery,* 14:389–418, 1967.
8. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
9. H. Maurer. The existence of context-free languages which are inherently ambiguous of any degree. Research series, Department of Mathematics, University of Calgary, 1968.
10. M. Naji. Grad der Mehrdeutigkeit kontextfreier Grammatiken und Sprachen, 1998. Diplomarbeit, FB Informatik, Johann–Wolfgang–Goethe–Universität.
11. W. Ogden. A helpful result for proving inherent ambiguity. *Mathematical Systems Theory,* 2(3):191–194, 1968.
12. R. J. Parikh. Language–generating devices. In *Quarterly Progress Report,* volume 60, pages 199–212. Research Laboratory of Electronics, M.I.T, 1961.
13. P. Rossmanith and W. Rytter. Observations on $\log(n)$ time parallel recognition of unambiguous cfl's. *Information Processing Letters,* 44:267–272, 1992.
14. E. Shamir. Some inherently ambiguous context-free languages. *Information and Control,* 18:355–363, 1971.
15. K. Wich. Exponential ambiguity of context-free grammars. In G. Rozenberg and W. Thomas, editors, *Proceedings of the DLT, 1999,* pages 125–138. World Scientific, Singapore, 2000.

16. K. Wich. Sublinear ambiguity. In M. Nielsen and B. Rovan, editors, *Proceedings of the MFCS 2000,* number 1893 in Lecture Notes in Computer Science, pages 690–698, Berlin-Heidelberg-New York, 2000. Springer.
17. K. Wich. Characterization of context-free languages with polynomially bounded ambiguity. In J. Sgall, A. Pultr, and P. Kolman, editors, *Proceedings of the MFCS 2001, Mariánské Lázně, (Czech Republic), 2001,* number 2136 in Lecture Notes in Computer Science, pages 703–714, Berlin-Heidelberg-New York, 2001. Springer.
18. K. Wich. Universal inherence of cycle-free context-free ambiguity functions. In P. Widmayer et al., editors, *Proceedings of the ICALP 2002,* number 2380 in Lecture Notes in Computer Science, pages 669–680, Berlin-Heidelberg-New York, 2002. Springer.

An Elementary Proof for the Non-parametrizability of the Equation $xyz = zvx$

Elena Petre*

Department of Mathematics, University of Turku and
Turku Centre for Computer Science
Turku 20520, Finland
epetre@it.utu.fi

Abstract. Although it is decidable whether or not a word equation is satisfiable, due to Makanin's results, the structure of all solutions in general is difficult to describe. In particular, it was proved by Hmelevskii that the solutions of $xyz = zvx$ cannot be finitely parametrized, contrary to the case of equations in three unknowns. In this paper we give a short, elementary proof of Hmelevkii's result. We also give a simple, necessary, non-sufficient condition for an equation to be non-parametrizable.

1 Introduction

The theory of *word equations,* a central subfield of *Combinatorics on Words* was initiated in 1954 by A. A. Markov. He proposed in [8] the problem of satisfiability of word equations: decide whether or not a given word equation has solutions. The problem was solved by Makanin who proved it to be decidable for free semigroups in [9], and for free groups in [10] and [11], see also [2] for a recent survey. Makanin's algorithm is considered as one of the most involved results in the field and it has exponential space complexity. This was recently improved by Plandowski who gave an algorithm with polynomial space complexity, see [13] and [14]. Nevertheless, neither Makanin's result, nor Plandowski's describe the general structure of the solutions of word equations; note however that there is an algorithm due to Razborov which generates all the solutions of a given equation, see [16] and also [2].

The solutions of constant-free equations in three unknowns can be expressed using only a finite number of *parametric formulas,* i.e., formulas involving word parameters and numerical parameters, as shown by Hmelevskii in [3]. Perhaps more importantly, this is a *boundary point*– Hmelevskii also proved that solutions of equations with four unknowns need not be finitely parametrizable and gave a concrete example of such an equation: $xyz = zvx$. Nevertheless, Hmelevskii did characterize all solutions of this equation, using however an infinite number of parameters. This characterization has been recently simplified by Weinbaum in [17]. Hmelevskii's results are also discussed in a chapter, [1], in Lothaire [6].

* This research was supported by Academy of Finland under grant 44087.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 807–817, 2004.
© Springer-Verlag Berlin Heidelberg 2004

In this paper, we present a short, elementary proof for the non-parametrizability of the equation $xyz = zvx$, simplifying Hmelevskii's proof. Our approach uses only elementary techniques on word equations and the well-known property of the Fibonacci word of being 4th power free, see [4].

Based on Levi's lemma, one can associate with each word equation a graph describing an exhaustive search for solutions. For equations with a finite associated graph, such as *quadratic equations,* we prove that a necessary condition for the non-parametrizability is to have in the graph two cycles sharing a vertex. However, this condition is not sufficient: the quadratic equation $xyz = zyx$ although it satisfies the condition, is finitely parametrizable, as shown by Hmelevskii.

The paper is organized as follows. In Sect. 2 we fix our terminology and introduce some basic notions and results. In Sect. 3 we present the main result of this paper, the fact that the equation $xyz = zvx$ is not finitely parametrizable. In Sect. 4 we consider the graph associated with an equation and its connection to the parametrizability of its set of solutions.

2 Preliminaries

Let Σ be a finite alphabet. We denote by Σ^* the set of all finite words over the alphabet Σ, by 1 the empty word, and by Σ^+ the set of all nonempty finite words over Σ, $\Sigma^+ = \Sigma^* \backslash \{1\}$. A word u is a *factor* (resp. *left factor* or *prefix, right factor* or *suffix*) of w if there are words x, y such that $w = xuy$ (resp. $w = uy$, $w = xu$). We use the notation $pref_k(w)$ (resp. $suf_k(w)$) to denote the prefix (resp. the suffix) of length k of the word w. For a word w let us denote by $Alph(w)$ the set of distinct letters appearing in it and by $|w|$ its *length,* i.e., the number of letters in w. We say that two words u and v are *conjugates* if there exist words $x, y \in \Sigma^*$ such that $u = xy$ and $v = yx$. The following lemma is a well-known characterization for the conjugacy of two words.

Lemma 1. *Let $u, v \in \Sigma^+$. The following conditions are equivalent :*

1. *u and v are conjugates,*
2. *there exists a word z such that $uz = zv$,*
3. *there exist words z, p, q and a positive integer i such that $u = pq$, $v = qp$, and $z = (pq)^i p$.*

We define the *Fibonacci word* as the limit of the sequence of words given by the following recurrence formula:

$$w_0 = b, \quad w_1 = a, \quad w_n = w_{n-1}w_{n-2}, \text{ for all } n \geq 2 \ . \tag{1}$$

One of the important properties of this word, see [4], is that it is 4-*free,* i.e., it does not contain any factor of the form w^4, with $w \in \Sigma^+$. In fact, in [12] is shown that Fibonacci word is $(2+\phi)^-$-free, where $\phi = \frac{1}{2}(\sqrt{5}+1)$ is the golden

number, but in our considerations we need only the 4-freeness property. We also notice that

$$suf_2(w_n) = \begin{cases} ba, & \text{if } n \text{ is odd} \\ ab, & \text{if } n \text{ is even} \end{cases} \quad \text{for all } n \geq 2 \ . \tag{2}$$

Let us consider next the words given by the following formula:

$$G_n = pref_{(f_{n+1}-2)}\, w_{n+1}, \text{ for all } n \geq 1, \tag{3}$$

where f_n are the Fibonacci numbers given by the recurrence formula:

$$f_0 = 1, \quad f_1 = 1, \quad f_n = f_{n-1} + f_{n-2}, \text{ for all } n \geq 2 \ .$$

Thus, for all indexes n, the words G_n are prefixes of the Fibonacci word. Notice also, that for every $n \geq 0$, $|w_n| = f_n$.

Now, let Σ be a finite alphabet and X a finite set of unknowns, with $\Sigma \cap X = \emptyset$. An *equation* over the alphabet Σ, with X as the set of unknowns is a pair

$$(u, v) \in (\Sigma \cup X)^* \times (\Sigma \cup X)^* \ .$$

Normally, an equation is written as $u = v$. We say that an equation is *constant-free* if both u and v contain only elements from X. The *total length* of an equation $u = v$ is the sum of the lengths of u and v. An equation $u = v$ is called *reduced* if $pref_1(u) \neq pref_1(v)$ and $suf_1(u) \neq suf_1(v)$. A *solution* of an equation $u = v$ is a morphism $\varphi : (X \cup \Sigma)^* \to \Sigma^*$ such that $\varphi(u) = \varphi(v)$ and $\varphi(a) = a$ for every $a \in \Sigma$. Consequently, a solution is a $|X|$-tuple of words over the alphabet Σ.

We define *word parameters* and *numerical parameters* as parameters whose values are words over the alphabet Σ, and positive integers, respectively. A *parametric word* over the alphabet Δ is defined as:

1. every letter in Δ is a parametric word;
2. if δ is a parametric word, and k is a numerical parameter, then δ^k is a parametric word;
3. if δ_1 and δ_2 are parametric words, then also $\delta_1\delta_2$ is a parametric word, where $\delta_1\delta_2$ is obtained by concatenating δ_1 and δ_2.

We define the concatenation of parametric words as the natural extension of concatenation of words.

Given a parametric word δ, every assignment φ of values in Σ^* to the letters of Δ, and of values in N to the numerical parameters, defines a unique word in Σ^*, called the *value* of δ under φ, and is denoted by $\varphi(\delta)$.

We say that an equation over Σ and with n unknowns is *parametrizable* if there exists a finite number of n-tuples of finite parametric words $F_1, \ldots, F_k$ over an alphabet Δ such that every value of these n-tuples is a solution of the equation, and every solution is a value of at least one of these n-tuples. Every F_i, $1 \leq i \leq k$, is a *parametric solution* of the considered equation.

The following technical result is useful for our later considerations.

Lemma 2. *Let $u = v$ be a constant-free equation with n unknowns over the alphabet Σ with $|\Sigma| \geq 2$ and $(T_1, \ldots, T_n)$ be a parametric solution. Let $(V_1, \ldots, V_n) \in (\Delta^*)^n$ be the n-tuple obtained from $(T_1, \ldots, T_n)$ by assigning fixed values to all numerical parameters. Then, $(V_1, \ldots, V_n)$ is a solution of the equation $u = v$ over Δ.*

Proof. Since $(T_1, \ldots, T_n)$ is a parametric solution of $u = v$, for any assignment φ, $(\varphi(T_1), \ldots, \varphi(T_n))$ is a solution of $u = v$ over Σ.

Let $u' = v'$ be the relation over Δ obtained by substituting $(V_1, \ldots, V_n)$ in $u = v$. Suppose now that $u' = v'$ is not an identity over Δ. Then, up to cancelling a common prefix, we may assume that

$$u' = \alpha u'' \text{ and } v' = \beta v'', \text{ with } u'', v'' \in \Delta^*, \ \alpha, \beta \in \Delta, \text{ and } \alpha \neq \beta \ . \qquad (4)$$

Consider now any assignment φ for $(T_1, \ldots, T_n)$ that assumes the numerical values fixed in $(V_1, \ldots, V_n)$ and takes $\varphi(\alpha) = a$ and $\varphi(\beta) = b$, where $a, b \in \Sigma$, $a \neq b$. Then $(\varphi(T_1), \ldots, \varphi(T_n)) = (\varphi(V_1), \ldots, \varphi(V_n))$ and consequently, it follows from (4) that $(\varphi(T_1), \ldots, \varphi(T_n))$ is not a solution of $u = v$ over Σ. But, this is a contradiction and so $u' = v'$ is an identity over Δ, i.e., $(V_1, \ldots, V_n)$ is a solution of the equation $u = v$ over Δ. □

There are several different approaches for solving equations. One of the most used is based on the idea of eliminating the leftmost (or rightmost) unknowns, see, e.g., [5] and [15].

This method, referred to as *Levi's lemma* or the *Nielsen transformation,* is based on the following lemma:

Lemma 3. *If words u, w, x and y over the alphabet Σ satisfy the relation $uw = xy$, then there exists the unique word t such that either*

1. *$u = xt$ and $y = tw$, or*
2. *$x = ut$ and $w = ty$.*

So, if we have an equation $x\alpha = y\beta$ with $\alpha, \beta \in (\Sigma \cup X)^*, x, y \in X$, then we can write $x = yt$ (or $y = xt$), and substitute this to the initial equation to obtain $yt\alpha = y\beta$, or equivalently $t\alpha = \beta$.

This substitution can simplify the equation, and thus make it easier to solve. However, this substitution should be done for every occurrence of the unknown x both in α and in β, so the total length of the equation may increase. It also means that the set of unknowns is changed from X into $X \cup \{t\} \setminus \{x\}$.

Levi's lemma gives a systematic way to search for the solutions of a given equation. This can be represented as a graph where each vertex is an equation $xu = yv$, with x and y unknowns or constants. From each vertex with x and y unknowns, we draw edges to three other equations derived from $xu = yv$ as follows, see also Fig. 1 (if either x or y is a constant, then we have only two equations derived from $xu = yv$):

1. if $|x| > |y|$, then $x = yx'$ and the equation $xu = yv$ on the set of unknowns X is equivalent to the equation $x'u' = v'$ on the set of unknowns $(X\backslash\{x\})\cup\{x'\}$, where u' and v' are obtained from u and v respectively by replacing every occurrence of x with yx';
2. if $|x| = |y|$, then $x = y$ and the equation $xu = yv$ on the set of unknowns X is equivalent to the equation $u' = v'$ on the set of unknowns $X\backslash\{x\}$, where u' and v' are obtained from u and v respectively by replacing every occurrence of x with y;
3. if $|x| < |y|$, then $y = xy'$ and the equation $xu = yv$ on the set of unknowns X is equivalent to the equation $u' = y'v'$ on the set of unknowns $(X\backslash\{y\})\cup\{y'\}$, where u' and v' are obtained from u and v respectively, by replacing every occurrence of y with xy'.

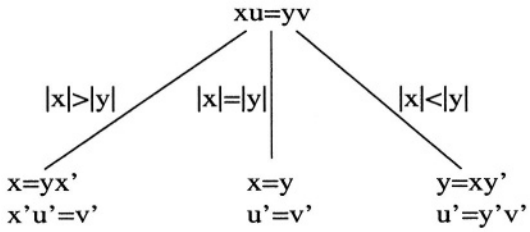


Fig. 1. The graph associated with equation $xu = yv$

Note that we only replace one unknown with another, or identify one with another, so we can choose not to rename it. In this case, the graph will be transformed as shown in Fig. 2.

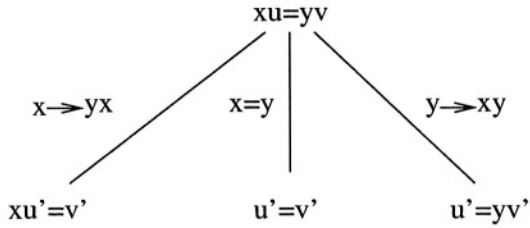


Fig. 2. In Fig. 1 we can use x, y instead of x', y' without loss of generality

Now, the initial equation has a solution with $|x| > |y|$ if and only if the equation $xu' = v'$ has a solution. On the other hand, if we have a solution for the equation $xu' = v'$, then we obtain a solution for the initial equation by replacing x with yx. Thus, all the solutions for the equation $xu = yv$ are found

by solving all the equations on the leaves of the graph and applying Levi's lemma in the reverse order.

An equation where each unknown occurs at most twice is called *quadratic*. Clearly, applying Levi's lemma to a quadratic equation does not increase the total length. Consequently, the graph associated with a quadratic equation is finite. In particular, it is decidable if a quadratic equation has a solution. All we have to do is to verify whether or not we get a solution on some leaf of the associated graph. Even more, if the equation has a solution, then we can actually find it using the graph.

3 Main Result

In [3] Hmelevskii proved that all constant-free equations in three unknowns are parametrizable. However, the situation changes when the number of unknowns increases. In the same paper, Hmelevskii gives also an example of an equation with four unknowns which is not parametrizable, i.e., the equation $xyz = zvx$. Here, we reprove this result. Our proof becomes shorter and easier to understand due to an efficient use of some basic techniques on word equations and of the properties of the Fibonacci words.

Theorem 1. *The set of solutions of the equation $xyz = zvx$ over an alphabet with at least two distinct letters is not parametrizable.*

Proof. Let us suppose that the equation $xyz = zvx$ is parametrizable. This means that we have a finite number of 4-tuples (T_1, T_2, T_3, T_4), where all T_i's, $1 \leq i \leq 4$ are finite parametric words, from which we can obtain all the solutions, and also any solution matches at least one of the patterns (T_1, T_2, T_3, T_4). Thus, we also have a finite number of word and numerical parameters.

Let (T_1, T_2, T_3, T_4) be a parametric solution of the equation $xyz = zvx$, and let Δ and Λ be the sets of word parameters and numerical parameters respectively, which appear in formulas T_i, $1 \leq i \leq 4$. Let (V_1, V_2, V_3, V_4) be the 4-tuple obtained from (T_1, T_2, T_3, T_4) by assigning some fixed values to all numerical parameters. Now, note that for every $1 \leq i \leq 4$, we have $V_i \in \Delta^*$ and we can define the length, prefix, suffix, and alphabet for such a word like in Sect. 2 and denote them as usual by $|V_i|$, $pref_k(V_i)$, $suf_k(V_i)$, and $Alph(V_i)$, respectively. Moreover, from Lemma 2, (V_1, V_2, V_3, V_4) is a solution of $xyz = zvx$ over Δ, i.e., $V_1V_2V_3 = V_3V_4V_1$ is an identity over the set of word parameters. In particular we see that $|V_2| = |V_4|$.

Note that if (V_1, V_2, V_3, V_4) is a solution of the equation $xyz = zvx$ over Δ, then (V_3, V_4, V_1, V_2) is also a solution over Δ. So, we can suppose without loss of generality that $|V_1| \geq |V_3|$.

We prove now that for such (V_1, V_2, V_3, V_4) we must have

$$V_2 = V_4 \text{ or } Alph(V_1V_3) \subseteq Alph(V_2V_4) \ .$$

We discuss here three cases, depending on the length of V_1.

Case 1: If $|V_1| = |V_3|$, since $V_1V_2V_3 = V_3V_4V_1$ is an identity over Δ, then we have $V_1 = V_3$ and $V_2 = V_4$. Thus, in this case (V_1, V_2, V_3, V_4) is such that $V_2 = V_4$.

Case 2: If $|V_3| < |V_1| \leq |V_3V_4| = |V_2V_3|$, then we can write $V_1 = V_3P$ and $V_1 = QV_3$, where $P \in \Delta^+$ is a prefix of V_4 and $Q \in \Delta^+$ is a suffix of V_2. So, $QV_3 = V_3P$, which implies (by Lemma 1) that there are two words $S, W \in \Delta^*$ and a positive integer $i \geq 0$ such that

$$Q = SW, \quad P = WS, \quad V_3 = (SW)^iS, \quad V_1 = (SW)^{i+1}S \ .$$

Now, let $\gamma \in Alph(V_1V_3) \subseteq \Delta$. This means that γ appears in at least one of the words S or W, so it appears both in P and in Q, i.e., $\gamma \in Alph(V_2V_4)$.

Thus, in this case, (V_1, V_2, V_3, V_4) is such that $Alph(V_1V_3) \subseteq Alph(V_2V_4)$.

Case 3: If $|V_1| > |V_3V_4|$, then we can write $V_1 = V_3V_4P$ and $V_1 = QV_2V_3$, with $P, Q \in \Delta^+$. Substituting these relations in the identity $V_1V_2V_3 = V_3V_4V_1$, we obtain that $P = Q$. So, in this case (V_1, V_2, V_3, V_4) is a solution of equation $xyz = zvx$ over Δ if and only if

$$PV_2V_3 = V_3V_4P \quad \text{and} \quad V_1 = PV_2V_3,$$

i.e., (P, V_2, V_3, V_4) is a solution of equation $xyz = zvx$ over Δ and $V_1 = PV_2V_3$. If $V_1 = P$, i.e., $V_2V_3 = 1$, then we immediately obtain that $V_2 = V_3 = V_4 = 1$. So, in this case the solution (V_1, V_2, V_3, V_4) is such that $V_2 = V_4$. Otherwise (i.e. $V_1 \neq P$), we reduced the solution (V_1, V_2, V_3, V_4) to (P, V_2, V_3, V_4), with $|P| < |V_1|$, and $V_1 = PV_2V_3$. Now, we can repeat this reduction step until $|P| \leq |V_3V_4|$, which means that we can apply *Case 1* or *Case 2* for the new solution (P, V_2, V_3, V_4). So, for (P, V_2, V_3, V_4) we have $V_2 = V_4$ or $Alph(PV_3) \subseteq Alph(V_2V_4)$. But, since $V_1 = PV_2V_3$, this implies that for the solution (V_1, V_2, V_3, V_4) we have $V_2 = V_4$ or $Alph(V_1V_3) \subseteq Alph(V_2V_4)$.

So, for any parametric solution (V_1, V_2, V_3, V_4) containing only word parameters, i.e., obtained from some parametric solution (T_1, T_2, T_3, T_4) by fixing the values of the numerical parameters, we must have $V_2 = V_4$ or $Alph(V_1V_3) \subseteq Alph(V_2V_4)$.

Now, we claim that the 4-tuple (G_k, ab, G_{k-1}, ba) is a solution over Σ of the equation $xyz = zvx$ for any odd index k, where words G_n, $n \geq 1$ are defined by the formula (3) in Sect. 2. To prove this claim it is enough to prove that $G_kabG_{k-1} = G_{k-1}baG_k$, for any odd index k. Using formulas (2) and (3), this identity is equivalent to $w_{k+1}G_{k-1} = w_kG_k$. Now, using formula (1), we obtain $w_{k-1}G_{k-1} = G_k$ and this can be proved by induction on k using formulas (1) and (3).

Consider an assignment φ and a parametric solution (T_1, T_2, T_3, T_4) such that $(\varphi(T_1), \varphi(T_2), \varphi(T_3), \varphi(T_4)) \in \{(G_k, ab, G_{k-1}, ba) \mid k \text{ odd }\}$. We prove now that the length of $\varphi(T_1)$ is bounded by some constant.

First, since every G_k is a prefix of the Fibonacci word, which is 4-free, and $\varphi(T_1) = G_k$ for some k odd, we must have that every power of a factor in $\varphi(T_1)$ is

less than 4. Consequently, for every numerical parameter λ appearing in formula T_1 we must have $\varphi(\lambda) < 4$.

Second, consider the 4-tuple (V_1, V_2, V_3, V_4) over Δ obtained from the parametric solution (T_1, T_2, T_3, T_4) by substituting every numerical parameter λ with its value $\varphi(\lambda)$. Since $\varphi(T_i) = \varphi(V_i)$ for every $1 \leq i \leq 4$, we obtain the following relations:

$$G_k = \varphi(V_1),\ G_{k-1} = \varphi(V_3), \tag{5}$$

$$ab = \varphi(V_2),\ ba = \varphi(V_4)\ . \tag{6}$$

Notice now that the values of formulas V_2 and V_4 under the assignment φ must be *ab* and respectively *ba,* so $V_2 \neq V_4$. Thus, this 4-tuple (V_1, V_2, V_3, V_4) must be such that $Alph(V_1V_3) \subseteq Alph(V_2V_4)$. Moreover, from relations (6) we observe that $|\varphi(\alpha)| \leq 2$, for any word parameter $\alpha \in Alph(V_2V_4)$, which implies that $|\varphi(\alpha)| \leq 2$ also for any word parameter $\alpha \in Alph(V_1V_3)$.

Consequently, $|\varphi(T_1)| = |\varphi(V_1)|$ is bounded by some positive constant, i.e., we cannot generate arbitrarily large solutions (G_k, ab, G_{k-1}, ba), with k odd. But this is a contradiction since words G_k can be arbitrarily large.

Thus, the equation $xyz = zvx$ is not parametrizable. □

4 Graphs Associated with Equations

With every word equation, we can associate a graph describing an exhaustive search for solutions, as presented in Sect. 2 of this paper. In this section we study the connection between the parametrizability of an equation and its associated graph.

Let us consider first the graph associated with the equation $xyz = zvx$, discussed in the previous section. Since every unknown appears at most twice in this equation, we know that the associated graph is finite (see Fig. 3).

This graph describes all solutions of the equation $xyz = zvx$. Any solution can be obtained by fixing some unknowns to be empty, solving the conjugacy equation and applying Levi's lemma in the reverse order. Notice that the graph associated with this equation contains two cycles having a common node (namely the root of the graph). Thus, when we reach that node, we can combine these two cycles in any way to obtain a new solution.

Let us consider, for example, the solutions obtained starting from the leftmost leaf (the solutions are underlined):

$$(x, y, z, v) \to (pq, qp, qpq, v) \to (v, qp, qpq, v) \to \underline{(qpqv, qp, qpq, v)} \to$$

$$(vqpqv, qp, qpq, v) \to \underline{(qpqvqpqv, qp, qpq, v)} \to (qpqvqpqv, qp, qpqpq, v)$$

$$\to \underline{(qpqvqpqv, qp, qpqvqpqvqpqpq, v)} \to \ldots$$

In this example, when we reached the root, i.e., the common node of the two cycles, we iterated first the left cycle and afterwards the right cycle. But these

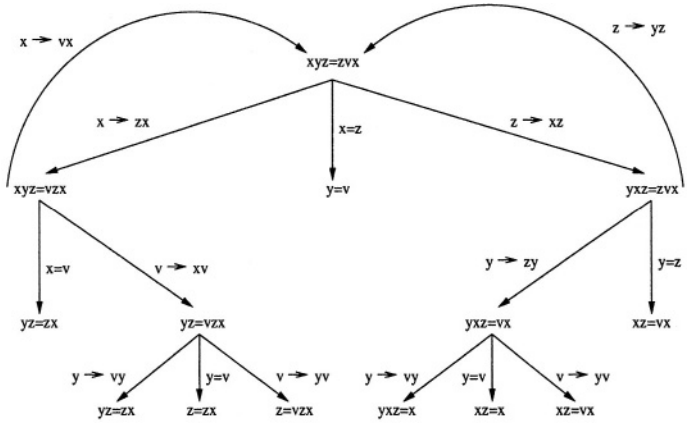


Fig. 3. The graph associated with the equation $xyz = zvx$

two cycles can be combined in any way and a solution is obtained every time the root of the graph is reached.

We also notice that, if there are two cycles with a common node, then on each cycle there must be an unknown which modifies its value depending on the value of another unknown which is modified in the other cycle. Thus, one may think that it is sufficient for an equation to have two cycles sharing a node in its graph, in order to be non-parametrizable. But this is not true in general.

Let us consider, for example, the graph associated with the equation $xyz = zyx$, which is also finite, see Fig. 4.

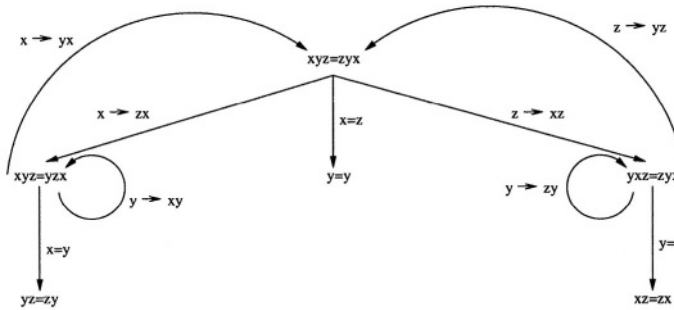


Fig. 4. The graph associated with the equation $xyz = zyx$

This is an equation with three unknowns for which we know that it has a finite parametric solution of the form:

$$x = (\alpha\beta)^i\alpha, \quad y = \beta(\alpha\beta)^j, \quad z = (\alpha\beta)^k\alpha,$$

where α, β are word parameters and i, j, k are numerical parameters, see [3].

We see in Fig. 4 that the graph associated with this equation has two cycles with a common node, i.e., the root of the graph. Moreover, the left cycle modifies the value of x using the current value of z, while the right cycle modifies the value of z using the current value of x. So, it is not sufficient to have two cycles with a common node in order to have a non-parametrizable equation.

On the other hand we notice that if the graph associated with an equation is finite and does not have two cycles sharing a node, then the equation is finitely parametrizable and, moreover, we obtain the general solution from the graph. If there are no cycles at all, then, to obtain the general solution, we just solve the equations on the leaves, and apply Levi's lemma in the reverse order until we reach the root of the graph. If there are some cycles, but not intersecting each other, then in the general solution we have some subwords of the form w^k, where k is a numerical parameter which represents the number of times we perform the cycles.

So, it is necessary to have two cycles with a common node in order to have a non-parametrizable equation.

Let us consider, for example, the equation $x^2y = u^2v$. We notice again that all the unknowns appear at most twice so the associated graph is finite, see Fig. 5.

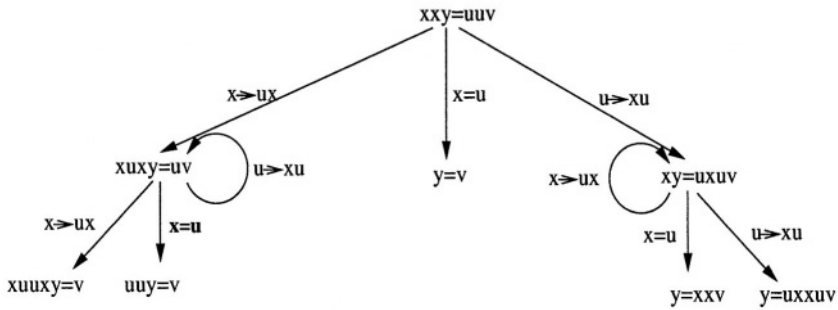


Fig. 5. The graph associated with the equation $x^2y = u^2v$

Although the graph contains two cycles, the one on the node labelled with *xuxy = uv* and the symmetric one on the right side of the graph, the general solution (x, y, u, v) of this equation has a finite expression using parametric words, and moreover it can be obtained from the graph:

$$x = (\alpha\beta)^t\alpha\alpha\beta, \quad y = \gamma, \quad u = (\alpha\beta)^t\alpha, \quad v = \beta\alpha^2\beta\gamma,$$

$$x = \alpha^{t+1}, \quad y = \beta, \quad u = \alpha^t, \quad v = \alpha^2\beta,$$

plus the symmetric ones (u, v, x, y) obtained from the right side of the graph, where $\alpha, \beta, \gamma \in \Sigma^*$ and $t \geq 0$. Iterating the cycles we only increase the value of the parameter t, but the expression of the general solution remains the same.

Acknowledgement. I am grateful to my supervisor, Prof. Juhani Karhumäki, for suggesting this problem and for his constant help with improving intermediary versions of this paper. I thank also Prof. Tero Harju for drawing my attention to the recent results in [17].

References

1. C. CHOFFRUT, Equations in Words. In: M. LOTHAIRE, *Combinatorics on words,* Chapter 9, Encyclopedia of Mathematics and its applications **17**, Addison-Wesley Publishing Co., 1983.
2. V. DIEKERT, Makanin's algorithm. In: M. LOTHAIRE, *Algebraic combinatorics on words,* Chapter 12, Encyclopedia of Mathematics and its applications **90**, Cambridge University Press, Cambridge, 2002.
3. J. I. HMELEVSKII, Equations in free semigroups, Translated by G. A. Kandall from the Russian original: Trudy Mat. Inst. Steklov. **107**, 1971, 1-270, American Mathematical Society, Providence, R.I., 1976.
4. J. KARHUMÄKI, *On cube-free ω-words generated by binary morphisms,* Discrete Appl. Math., **5,** 1983, 279-297.
5. A. LENTIN, *Equations in free monoids.* In M. Nivat, editor, Automata, languages and programming, 67–85, 1972.
6. M. LOTHAIRE, *Combinatorics on words,* Encyclopedia of Mathematics and its applications **17**, Addison-Wesley Publishing Co., 1983.
7. M. LOTHAIRE, *Algebraic combinatorics on words,* Encyclopedia of Mathematics and its applications **90**, Cambridge University Press, 2002.
8. A. A. MARKOV, *The theory of algorithms,* Trudy Math. Inst. Steklov, 42, 1954.
9. G. S. MAKANIN, *The problem of solvability of equations in a free semigroup,* Math, USSR Sbornik **32**, 129-198, 1977.
10. G. S. MAKANIN, *Equations in a free group,* Izv. Akad. Nauk SSSR Ser. Mat., **46**, 1199-1273, 1982. In Russian. English translation in: Math. USSR-Izv., **21**, 483-546, 1983.
11. G. S. MAKANIN, *Decidability of the universal and positive theories of a free group,* Izv. Akad. Nauk SSSR Ser. Mat., **48**, 735-749, 1984. In Russian. English translation in: Math. USSR-Izv., **25**, 75-88, 1985.
12. F. MIGNOSI, G. PIRILLO, *Repetitions in the Fibonacci infinite word,* RAIRO Inform. Theor. Appl. **26**, 1992, 199-204.
13. W. PLANDOWSKI, *Satisfiability of word equations with constants is in NEXPTIME,* Annual ACM Symposium on Theory of Computing, ACM, New York, 721-725, 1999.
14. W. PLANDOWSKI, *Satisfiability of word equations with constants is in PSPACE,* 40th Annual Symposium on Foundations of Computer Science, IEEE Computer Soc., Los Alamitos, CA, 495-500, 1999. Final version in JACM (to appear).
15. G. PLOTKIN, *Building in equational theories,* Machine Intelligence **7**, 73-90, 1972.
16. A. A: RAZBOROV, *On systems of equations in a free group,* Izv. Akad. Nauk SSSR Ser. Mat., **48**, 779-832, 1984. In Russian. English translation in: Math. USSR-Izv., **25**, 115-162, 1985.
17. C. M. WEINBAUM, *Word equation $ABC = CDA$, $B \neq D$,* Pacific J. Math., Vol. 213, **1,** 157-162, 2004.

A Generalization of Repetition Threshold

Lucian Ilie[1]*, Pascal Ochem[2], and Jeffrey Shallit[3]**

[1] Department of Computer Science, University of Western Ontario
London, ON N6A 5B7, CANADA
ilie@csd.uwo.ca
[2] LaBRI, Université Bordeaux I
351, cours de la Libération, 33405 Talence Cedex, FRANCE
ochem@labri.fr
[3] School of Computer Science, University of Waterloo
Waterloo, ON N2L 3G1, CANADA
shallit@graceland.uwaterloo.ca

Abstract. Brandenburg and (implicitly) Dejean introduced the concept of *repetition threshold:* the smallest real number α such that there exists an infinite word over a k-letter alphabet that avoids β-powers for all $\beta > \alpha$. We generalize this concept to include the lengths of the avoided words. We give some conjectures supported by numerical evidence and prove three of these conjectures. As a consequence of one of our results, we show that the pattern *ABCBABC* is 2-avoidable. This resolves a question left open in Cassaigne's thesis.

1 Introduction

In this paper we consider some variations on well-known theorems about avoiding repetitions in words.

A *square* is a repetition of the form xx, where x is a nonempty word; an example in English is `hotshots`. Let Σ_k denote the k-letter alphabet $\{0, 1, \ldots, k-1\}$. It is easy to see that every word of length ≥ 4 over Σ_2 must contain a square, so squares cannot be avoided in infinite binary words. However, Thue showed [16, 17,2] that there exist infinite words over Σ_3 that avoid squares.

Instead of avoiding *all* squares, one interesting variation is to avoid *all sufficiently large* squares. Entringer, Jackson, and Schatz [8] showed that there exist infinite binary words avoiding all squares xx with $|x| \geq 3$. Furthermore, they proved that every binary word of length ≥ 18 contains a factor of the form xx with $|x| \geq 2$, so the bound 3 is best possible. For some other papers about avoiding sufficiently large squares, see [7,13,9,14,15].

Another interesting variation is to consider avoiding fractional powers. For $\alpha \geq 1$ a rational number, we say that y is an α-power if we can write $y = x^n x'$ with x' a prefix of x and $|y| = \alpha|x|$. For example, the French word `entente` is a $\frac{7}{3}$-power and the English word `tormentor` is a $\frac{3}{2}$-power. For real $\alpha > 1$,

* Research supported in part by NSERC.
** Research supported in part by NSERC.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 818–826, 2004.
© Springer-Verlag Berlin Heidelberg 2004

we say a word *avoids α-powers* if it contains no factor that is a α'-power for any rational $\alpha' \geq \alpha$. Brandenburg [3] and (implicitly) Dejean [6] considered the problem of determining the *repetition threshold;* that is, the least exponent $\alpha = \alpha(k)$ such that there exist infinite words over Σ_k that avoid $(\alpha+\epsilon)$-powers for all $\epsilon > 0$. Dejean proved that $\alpha(3) = \frac{7}{4}$. She also conjectured that $\alpha(4) = \frac{7}{5}$ and $\alpha(k) = \frac{k}{k-1}$ for $k \geq 5$. In its full generality, this conjecture is still open, although Pansiot [12] proved that $\alpha(4) = \frac{7}{5}$ and Moulin-Ollagnier [11] proved that Dejean's conjecture holds for $5 \leq k \leq 11$. For more information, see [5].

In this paper we consider combining these two variations. We generalize the repetition threshold of Dejean to handle avoidance of all sufficiently large fractional powers. (Pansiot also suggested looking at this generalization at the end of his paper [12], but to the best of our knowledge no one else has pursued this question.) We give a large number of conjectures, supported by numerical evidence, about generalized repetition threshold, and prove three of them. Finally, some applications of our results to pattern avoidability are presented. In particular, we prove that the pattern *ABCBABC* is 2-avoidable, which resolves a question left open in Cassaigne's thesis [4], and implies that every ternary pattern is either unavoidable or 3-avoidable.

2 Definitions

Let $\alpha > 1$ be a rational number, and let $\ell \geq 1$ be an integer. A word w is a *repetition of order α and length ℓ* if we can write it as $w = x^n x'$ where x' is a prefix of x, $|x| = l$, and $|w| = \alpha|x|$. For brevity, we also call w a (α, ℓ)-*repetition.* Notice that an α-power is an (α, ℓ)-repetition for some ℓ. We say a word is (α, ℓ)-*free* if it contains no factor that is a (α', ℓ')-repetition for $\alpha' \geq \alpha$ and $\ell' \geq \ell$. We say a word is (α^+, ℓ)-*free* if it is (α', ℓ)-free for all $\alpha' > \alpha$.

For integers $k \geq 2$ and $\ell \geq 1$, we define the *generalized repetition threshold* $R(k, \ell)$ as the real number α such that either

(a) over Σ_k there exists an (α^+, ℓ)-free infinite word, but all (α, ℓ)-free words are finite; or
(b) over Σ_k there exists a (α, ℓ)-free infinite word, but for all $\epsilon > 0$, all $(\alpha - \epsilon, \ell)$-free words are finite.

Notice that $R(k, 1)$ is essentially the repetition threshold of Dejean and Brandenburg.

Theorem 1. *The generalized repetition threshold $R(k, \ell)$ exists and is finite for all integers $k \geq 2$ and $\ell \geq 1$. Furthermore, $1 + \ell/k^\ell \leq R(k, \ell) \leq 2$.*

Proof. Define S to be the set of all real numbers $\alpha \geq 1$ such that there exists a (α, ℓ)-free infinite word over Σ_k. Since Thue proved that there exists an infinite word over a two-letter alphabet (and hence over larger alphabets) avoiding $(2+\epsilon)$-powers for all $\epsilon > 0$, we have that $\beta = \inf S$ exists and $\beta \leq 2$. If $\beta \in S$, we are in case (b) above, and if $\beta \notin S$, we are in case (a). Thus $R(k, \ell) = \beta$.

For the lower bound, note that any word of length $\geq k^\ell + \ell$ contains $\geq k^\ell + 1$ factors of length ℓ. Since there are only k^ℓ distinct factors of length ℓ, such a word contains at least two occurrences of some word of length ℓ, and hence is not $(1 + \frac{\ell}{k^\ell}, \ell)$-free. □

Remarks

1. It may be worth noting that we know no instance where case (b) of the definition of generalized repetition threshold above actually occurs, but we have not been able to rule it out.

2. Using the Lovász local lemma, Beck [1] has proved a related result: namely, for all $\epsilon > 0$, there exists an integer n' and an infinite $(1 + n/(2-\epsilon)^n, n)$-free binary word for all $n \geq n'$. Thus our work can be viewed as a first attempt at an explicit version of Beck's result (although in our case the exponent does not vary with n).

3 Conjectures

In this section we give some conjectures about $R(k, \ell)$.

Figure 1 gives the established and conjectured values of $R(k, \ell)$. Entries in **bold** have been proved; the others (with question marks) are merely conjectured. However, in either case, if the entry for (k, ℓ) is α, then we have proved, using the usual tree-traversal technique discussed below, that there is no infinite (α, ℓ)-free word over Σ_k.

The proved results are as follows:

- $R(2,1) = 2$ follows from Thue's proof of the existence of overlap-free words over Σ_2 [16,17,2];
- $R(2,2) = 2$ follows from Thue's proof together with the observation of Entringer, Jackson and Schatz [8];
- $R(3,1) = \frac{7}{4}$ is due to Dejean [6];
- $R(4,1) = \frac{7}{5}$ is due to Pansiot [12];
- $R(k,1) = \frac{k}{k-1}$ for $5 \leq k \leq 11$ is due to Moulin-Ollagnier [11];
- $R(3,2) = \frac{3}{2}$, $R(3,3) = \frac{4}{3}$ and $R(2,4) = \frac{3}{2}$ are new and are proved in Section 4.

We now explain how the conjectured results were obtained. We used the usual tree-traversal technique, as follows: suppose we want to determine if there are only finitely many words over the alphabet Σ_k that avoid a certain set of words S. We construct a certain tree T and traverse it using breadth-first or depth-first search. The tree T is defined as follows: the root is labelled ϵ (the empty word). If a node w has a factor contained in S, then it is a leaf. Otherwise, it has children labelled wa for all $a \in \Sigma_k$. It is easy to see that T is finite if and only if there are finitely many words avoiding S.

We can take advantage of various symmetries in S to speed traversal. For example, if S is closed under renaming of the letters (as is the case in the

$R(k,\ell)$	ℓ							
	1	2	3	4	5	6	7	8
2	**2**	**2**	$\frac{8}{5}$?	$\frac{3}{2}$	$\frac{7}{5}$?	$\frac{4}{3}$?	$\frac{31}{24}$?	$\frac{24}{19}$?
3	$\frac{7}{4}$	$\frac{3}{2}$	$\frac{4}{3}$	$\frac{5}{4}$?	$\frac{6}{5}$?	$\frac{7}{6}$?	$\frac{8}{7}$?	$\frac{9}{8}$?
4	$\frac{7}{5}$	$\frac{5}{4}$?	$\frac{6}{5}$?	$\frac{7}{6}$?	$\frac{8}{7}$?	$\frac{9}{8}$?		
5	$\frac{5}{4}$	$\frac{6}{5}$?	$\frac{8}{7}$?	$\frac{9}{8}$?				
6	$\frac{6}{5}$	$\frac{36}{31}$?						
k 7	$\frac{7}{6}$	$\frac{9}{8}$?						
8	$\frac{8}{7}$							
9	$\frac{9}{8}$							
10	$\frac{10}{9}$							
11	$\frac{11}{10}$							
12	$\frac{12}{11}$?							
13	$\frac{13}{12}$?							

Fig. 1. Known and conjectured values of $R(k,\ell)$.

examples we study), we can label the root with an arbitrary single letter (instead of ϵ) and deduce the number of leaves in the full tree by multiplying by k.

Furthermore, if we use depth-first search, we can in some cases dramatically shorten the search using the following observation: if at any point some suffix of the current string strictly precedes the prefix of the same length of the same string in lexicographic order, then this suffix must have already been examined. Hence we can immediately abandon consideration of this node.

If the tree is finite, then certain parameters about the tree give useful information about the set of finite words avoiding S:

- If h is the height of the tree, then any word of length $\geq h$ over Σ_k contains a factor in S.
- If M is the length of longest word avoiding S, then $M = h - 1$.
- If I is the number of internal nodes, then there are exactly I finite words avoiding S. Furthermore, if L is the number of leaves, then (as usual), $L = 1 + (k-1)I$.
- If I' is the number of internal nodes at depth $h-1$, then there are I' words of maximum length avoiding S.

k	ℓ	α	L	I	h	$M{=}h{-}1$	I'
2	1	**2**	8	7	4	3	2
2	2	**2**	478	477	19	18	2
2	3	8/5	5196	5195	34	33	12
2	4	**3/2**	13680	13679	54	53	4
2	5	7/5	40642	40641	60	59	4
2	6	4/3	21476	21475	40	39	4
2	8	24/19	3480734274	3480734273	452	451	NR
3	1	**7/4**	6393	3196	39	38	18
3	2	**3/2**	11655	5827	31	30	6
3	3	**4/3**	4037361	2018680	228	227	6
3	4	5/4	188247	94123	63	62	24
3	5	6/5	493653	246826	63	62	12
3	6	7/6	782931	391465	60	59	24
3	7	8/7	2881125	1440562	68	67	24
3	8	9/8	6987903	3493951	62	61	24
4	1	**7/5**	709036	236345	122	121	48
4	2	5/4	10324	3441	17	16	24
4	3	6/5	153724	51241	24	23	96
4	4	7/6	2501620	833873	35	34	24
4	5	8/7	30669148	10223049	40	39	864
4	6	9/8	340760884	113586961	50	49	NR
5	1	**5/4**	1785	446	7	6	120
5	2	6/5	453965	113491	23	22	240
5	3	8/7	7497345	1874336	34	33	720
5	4	9/8	1521535445	380383861	52	51	NR
6	1	**6/5**	13386	2677	8	7	720
6	2	36/31	17372138466	3474427693	751	750	NR
7	1	**7/6**	112441	18740	9	8	5040
7	2	9/8	345508219	57584703	32	31	NR
8	1	**8/7**	1049448	149921	10	9	40320

Fig. 2. Tree statistics for various values of k and l

Figure 2 gives the value of some of these parameters. Here α is the established or conjectured value of $R(k, \ell)$ from Figure 1. "NR" indicates that the value was not recorded by our program.

We have seen how to prove computationally that only finitely many (α, ℓ)-free words exist. But what is the evidence that suggests we have determined the smallest possible α? For this, we explore the tree corresponding to avoiding (α^+, ℓ)-repetitions using *depth-first* (and not breadth-first) search. If we are able to construct a "very long" word avoiding (α^+, ℓ)-repetitions, then we suspect we have found the optimal value of α. For each unproven α given in Figure 1, we were able to construct a word of length at least 5000 avoiding the corresponding repetitions. This constitutes weak evidence of the correctness of our conjectures, but it is evidently not conclusive.

Based on the data in Figure 1, we propose the following conjectures.

Conjecture 1. $R(3,\ell) = 1 + \frac{1}{\ell}$ for $\ell \geq 2$.

Conjecture 2. $R(4,\ell) = 1 + \frac{1}{\ell+2}$ for $\ell \geq 2$.

These conjectures are weakly supported by the numerical evidence above.

4 New Results

In this section, we prove three results of the form $R(k,l) = \alpha$. From the numerical results reported in Figure 2, we know in each case that there exist no infinite (α, l)-free words over Σ_k. It therefore suffices to exhibit an infinite (α^+, l)-free word over Σ_k. A uniform morphism $h : \Sigma_i^* \to \Sigma_k^*$ is said to satisfy the *synchronizing property* if for any $a, b, c \in \Sigma_i$ and $s, r \in \Sigma_k$, if $h(ab) = rh(c)s$, then either $r = \varepsilon$ and $a = c$ or $s = \varepsilon$ and $b = c$. It is straightforward to verify that all the morphisms in this section satisfy the synchronizing property.

Lemma 1. *Let $\alpha, \beta \in \mathbb{R}$, $1 < \alpha < \beta < 2$. Let $h : \Sigma_i^* \to \Sigma_k^*$ be a synchronizing q-uniform morphism, for some $q \geq 1$. If t is a shortest α^+-free word in Σ_i^* such that $h(t)$ contains some β^+-repetition, then*

$$|t| < \max\left(\frac{2\beta}{\beta-\alpha}, \frac{2(q-1)(2\beta-1)}{q(\beta-1)}\right).$$

Proof. Suppose $h(t)$ contains a β^+-repetition, that is, a factor uvu such that $\frac{|uvu|}{|uv|} > \beta$. Denote $x = |u|$ and $y = |v|$.

If $x \geq 2q-1$, then each occurrence of u contains at least one full h-image of a letter. As h is synchronizing, the two occurrences of u in uvu contain the same h-images and in the same positions. Therefore, there is a factor UVU in t such that, denoting $X = |U|$ and $Y = |V|$, we have $Yq < y + 2q$ and $Xq > x - 2q$, or equivalently $x < (X+2)q$. (Each U is the factor of t that contains all letters whose h-images are contained in the corresponding u.)

We have then $\frac{2x+y}{x+y} > \beta$, which gives $y < \frac{2-\beta}{\beta-1}x$. The fact that t is α^+-free implies that $\frac{2X+Y}{X+Y} \leq \alpha$, which gives $X \leq \frac{\alpha-1}{2-\alpha}Y$. Now we have

$$Yq < y+2q < \frac{2-\beta}{\beta-1}x+2q < \frac{2-\beta}{\beta-1}(X+2)q+2q \leq \frac{2-\beta}{\beta-1}\left(\frac{\alpha-1}{2-\alpha}Y+2\right)q+2q,$$

implying that $Y < \frac{2(2-\alpha)}{\beta-\alpha}$. By the minimality of t we get

$$|t| \leq 2+Y+2X \leq 2+Y\left(1+2\frac{\alpha-1}{2-\alpha}\right) < 2+\frac{2(2-\alpha)}{\beta-\alpha}\frac{\alpha}{2-\alpha} = \frac{2\beta}{\beta-\alpha}.$$

Consider next the case when $x \leq 2q-2$. This implies $y < \frac{2-\beta}{\beta-1}(2q-2)$ and thus $2x+y < \frac{2\beta}{\beta-1}(q-1)$. The minimality of t implies that $(|t|-2)q \leq |uvu|-2 = 2x+y-2$. By the above we get that

$$|t| < \frac{2(q-1)(2\beta-1)}{q(\beta-1)},$$

which completes the proof. □

For convenience, let us denote the maximum in Lemma 1 by $\max_{\alpha,\beta,q}$.

Theorem 2. $R(3,2) = \frac{3}{2}$.

Proof. Consider the 3-uniform morphism $h : \Sigma_4 \longrightarrow \Sigma_3$ defined by

$$\begin{aligned} h(0) &= 021, \quad & h(1) &= 100, \\ h(2) &= 122, \quad & h(3) &= 201. \end{aligned}$$

By a result of Pansiot [12], there exist $\frac{7}{5}^+$-free infinite words over Σ_4. Consider one such word $\mathbf{x}$. A computer check shows that for every $\frac{7}{5}^+$-free word $t \in \Sigma_4^*$ such that $|t| < \max_{\frac{7}{5},\frac{3}{2},3} = 30$, $h(t)$ is $(\frac{3}{2}^+, 2)$-free. By Lemma 1, this proves that $h(\mathbf{x})$ is an infinite ternary $(\frac{3}{2}^+, 2)$-free word. □

Theorem 3. $R(3,3) = \frac{4}{3}$.

Proof. Consider the 14-uniform morphism $h : \Sigma_5 \longrightarrow \Sigma_3$ defined by

$$\begin{aligned} h(0) &= 00011112122220, \quad & h(1) &= 00101112202021, \\ h(2) &= 01012111102120, \quad & h(3) &= 10002212102020, \\ h(4) &= 10100222112020. \end{aligned}$$

By a result of Moulin-Ollagnier [11], there exist $\frac{5}{4}^+$-free infinite words over Σ_5. Consider one such word $\mathbf{x}$. A computer check shows that for every $\frac{5}{4}^+$-free word $t \in \Sigma_5^*$ such that $|t| < \max_{\frac{5}{4},\frac{4}{3},14} = 32$, $h(t)$ is $(\frac{4}{3}^+, 3)$-free. By Lemma 1, this proves that $h(\mathbf{x})$ is an infinite ternary $(\frac{4}{3}^+, 3)$-free word. □

Theorem 4. $R(2,4) = \frac{3}{2}$.

Proof. Consider the 19-uniform morphism $h : \Sigma_4 \longrightarrow \Sigma_2$ defined by

$$\begin{aligned} h(0) &= 0000110100100111110, \quad & h(1) &= 0000011011001010111, \\ h(2) &= 0000011010100111111, \quad & h(3) &= 0000010110111110010. \end{aligned}$$

We again consider an infinite $\frac{7}{5}^+$-free word $\mathbf{x}$ over Σ_4. A computer check shows that for every $\frac{7}{5}^+$-free word $t \in \Sigma_4^*$ such that $|t| < \max_{\frac{7}{5},\frac{3}{2},19} = 30$, $h(t)$ is $(\frac{3}{2}^+, 4)$-free. By Lemma 1, this proves that $h(\mathbf{x})$ is an infinite binary $(\frac{3}{2}^+, 4)$-free word. □

5 Applications to Pattern Avoidability

Our results on the repetition threshold have some interesting applications to pattern avoidability. This is due to the following observation: A word avoiding a repetition which appears in any image of a pattern, avoids the pattern itself.

For a pattern $p \in A^*$, its pattern language $p(\Sigma^+)$ is the language over Σ which contains all the words $h(p)$, where h is a non-erasing morphism from A^* to Σ^*. (For further notions and results on avoidability, we refer to Chapter 3 in [10].) We say that the pattern p has an *inherent* (α^+, ℓ)-*repetition with respect to* Σ if any word in $p(\Sigma^+)$ contains an (α', ℓ')-repetition for some $\alpha' > \alpha$ and $\ell' \geq \ell$.

We then have the following general result which can be used to prove avoidability for many patterns.

Lemma 2. *If there exists an* (α^+, ℓ)-*free infinite word over* Σ_k, *then any pattern that has an inherent* (α^+, ℓ)-*repetition is* k-*avoidable.*

According to Cassaigne [4], the pattern *ABCBABC* was the only avoidable ternary pattern not known to be 3-avoidable. The next result solves this open problem as well as some other open ones.

Corollary 1. *The patterns ABCBABC, ABBCBABBC, ABCCBABC, and ABCBAABC are simultaneously* 2*-avoidable.*

Proof. Any of the patterns in the given set has an inherent $(\frac{3}{2}^+, 4)$-repetition with respect to any alphabet. Theorem 4 gives a $(\frac{3}{2}^+, 4)$-free infinite word over Σ_2 which, by Lemma 2, avoids simultaneously all patterns in the set. □

Corollary 1 and the results of Cassaigne [4] give the following theorem.

Theorem 5. *Every ternary pattern is either unavoidable or* 3*-avoidable.*

Acknowledgements. We thank Narad Rampersad for a careful reading of this paper and for pointing out the remark of Pansiot.

References

1. J. Beck. An application of Lovász local lemma: there exists an infinite 01-sequence containing no near identical intervals. In A. Hajnal, L. Lovász, and V. T. Sós, editors, *Finite and Finite Sets,* Vol. 37 of *Colloq. Math. Soc. János Bolyai,* pp. 103–107. 1981.
2. J. Berstel. *Axel Thue's Papers on Repetitions in Words: a Translation.* Number 20 in Publications du Laboratoire de Combinatoire et d'Informatique Mathématique. Université du Québec à Montreal, February 1995.
3. F.-J. Brandenburg. Uniformly growing k-th power-free homomorphisms. *Theoret. Comput. Sci.* **23** (1983), 69–82.

4. J. Cassaigne. Motifs évitables et régularité dans les mots Thèse de Doctorat, Université Paris VI, Juillet 1994.
5. C. Choffrut and J. Karhumäki. Combinatorics of words. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages,* Vol. 1, pp. 329–438. Springer-Verlag, 1997.
6. F. Dejean. Sur un théorème de Thue. *J. Combin. Theory. Ser. A* **13** (1972), 90–99.
7. F. M. Dekking. On repetitions of blocks in binary sequences. *J. Combin. Theory. Ser. A* **20** (1976), 292–299.
8. R. C. Entringer, D. E. Jackson, and J. A. Schatz. On nonrepetitive sequences. *J. Combin. Theory. Ser. A* **16** (1974), 159–164.
9. A. S. Fraenkel and R. J. Simpson. How many squares must a binary sequence contain? *Electronic J. Combinatorics* **2** (1995), #R2.
10. M. Lothaire. *Algebraic Combinatorics on Words.* Cambridge Univ. Press, 2002.
11. J. Moulin-Ollagnier. Proof of Dejean's conjecture for alphabets with 5,6,7,8,9,10 and 11 letters. *Theoret. Comput. Sci.* **95** (1992), 187–205.
12. J.-J. Pansiot. A propos d'une conjecture de F. Dejean sur les répétitions dans les mots. *Disc. Appl. Math.* **7** (1984), 297–311.
13. H. Prodinger and F. J. Urbanek. Infinite 0–1-sequences without long adjacent identical blocks. *Discrete Math.* **28** (1979), 277–289.
14. N. Rampersad, J. Shallit, and M.-w. Wang. Avoiding large squares in infinite binary words. In *Proceedings of Words '03: 4th International Conference on Combinatorics on Words,* pp. 185–197. 2003. Turku Centre for Computer Science, TUCS General Publication #27. Paper available at `http://www.arxiv.org/abs/math.CO/0306081`.
15. J. Shallit. Simultaneous avoidance of large squares and fractional powers in infinite binary words. *Int. J. Found. Comput. Sci.* **15** (2004), 317–327. Preprint available at `http://www.arxiv.org/abs/math.CO/0304476`, 2003.
16. A. Thue. Über unendliche Zeichenreihen. *Norske vid. Selsk. Skr. Mat. Nat. Kl.* **7** (1906), 1–22. Reprinted in *Selected Mathematical Papers of Axel Thue,* T. Nagell, editor, Universitetsforlaget, Oslo, 1977, pp. 139–158.
17. A. Thue. Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. *Norske vid. Selsk. Skr. Mat. Nat. Kl.* **1** (1912), 1–67. Reprinted in *Selected Mathematical Papers of Axel Thue,* T. Nagell, editor, Universitetsforlaget, Oslo, 1977, pp. 413–478.

An Algorithmic Argument for Nonadaptive Query Complexity Lower Bounds on Advised Quantum Computation*

(Extended Abstract)

Harumichi Nishimura and Tomoyuki Yamakami

Computer Science Program, Trent University
Peterborough, Ontario, Canada K9J 7B8

Abstract. This paper employs a powerful argument, called an algorithmic argument, to prove lower bounds of the quantum query complexity of a multiple-block ordered search problem, which is a natural generalization of the ordered search problem. Apart from much studied polynomial and adversary methods for quantum query complexity lower bounds, our argument shows that the multiple-block ordered search needs a large number of nonadaptive oracle queries on a black-box model of quantum computation that is also supplemented with advice. Our argument is also applied to the notions of computational complexity theory: quantum truth-table reducibility and quantum truth-table autoreducibility.

Keywords: Algorithmic argument, query complexity, nonadaptive query, advice, quantum computation

1 An Algorithmic Argument for Query Complexity

A major contribution of this paper is the demonstration of a powerful argument, which we refer to *algorithmic argument,* to prove a lower bound of the nonadaptive query complexity for a multiple-block ordered search problem on advised quantum computation. In the literature, quantum query complexity lower bounds have been proven by classical adversary methods [8], polynomial methods [6], and quantum adversary methods [3,5,14]. Each method has its own strength and advantages over its simplicity, clarity, and dexterity. An algorithmic argument, however, is essentially different from these methods in its constructive manner. A basic scheme of our algorithmic argument is illustrated as follows: we (i) commence with the faulty assumption that a quantum algorithm $\mathcal{A}$ of low query complexity exists, (ii) define a compression scheme E that encodes each input s of fixed length into a shorter string $E(s)$, and (iii) prove the one-to-oneness of E by constructing a decoding algorithm from $\mathcal{A}$ that uniquely extracts s from

* This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 827–838, 2004.
© Springer-Verlag Berlin Heidelberg 2004

$E(s)$, which leads to an obvious contradiction against the pigeonhole principle. We therefore conclude that any quantum algorithm should require high query complexity. We apply our algorithmic argument to obtain a new nonadaptive query complexity lower bound on a query computation model, called a *black-box quantum computer* (sometimes called a *quantum network* [6]), in which a *query* is an essential method to access information stored outside of the computer. The minimal number of such queries, known as the *query complexity,* measures the smallest amount of information necessary to finish the desired computation. Query complexity on various quantum computation models has been studied for numerous problems, including unordered search, ordered search, and element distinctness.

A black-box quantum computer starts with a fixed initial state (e.g., $|0\cdots 0\rangle$), accesses a given source x (which is called an "oracle") by way of queries—"what is the binary value at location i in x?"—and outputs a desired solution with small error probability. If any query (except the first one) is chosen according to the answers to its previous queries, such a query pattern is conventionally referred to as *adaptive.* Adaptive oracle quantum computations have been extensively studied and have given rise to useful quantum algorithms. In general, an adaptive computation requires a large number of interactions between the computer and a given oracle. Since a quantum computer is known to be sensitive to any interaction with other physical systems, such as an oracle, it would be desirable to limit the number of interactions with the oracle. In contrast, the query pattern of which all the query words are prepared before the first query is referred to as *nonadaptive queries* (including *parallel queries* and *truth-table queries*). Recently, Buhrman and van Dam [10] and Yamakami [18] extensively studied the nature of parallel queries on quantum computation.

A black-box quantum computation can be further equipped with *advice,* which was first discussed by Karp and Lipton, to provide an additional source of information to boost the computational power. Lately, time-bounded advised quantum computation was introduced by Nishimura and Yamakami [16]. This notion has a close connection to nonuniform computation and also one-way communication.

Based on a black-box model of quantum computation with advice, we employ our algorithmic argument to prove a lower bound of the query complexity of a so-called *ordered search problem.* For simplicity, we focus our attention on the ordered search problem of the following kind: given an N-bit string x of the form $0^{N-j}1^j$ for certain positive integers N and j, we are to find the leftmost location s of 1 (i.e., $N-j+1$). Such a unique location s is called the *step* of x (since the input x can be viewed as a so-called *step function*). This ordered search problem is one of the well-studied problems for their quantum adaptive query complexity. Naturally, we can expand this problem into a "multiple-block" ordered search problem, in which we are to find the step in each block i when the block index i is given as an input. To avoid any confusion, we call the standard ordered search problem the *single-block ordered search problem.* In this paper, we present a new

query complexity lower bound for the multiple-block ordered search problem on a nonadaptive black-box quantum computer with the help of advice.

Recently, Laplante and Magniez [15] also found a similar algorithmic argument to demonstrate general lower bounds of randomized and quantum query complexity. Their argument, however, is meant for the adaptive query complexity without advice strings.

Finally, we note that an algorithmic argument is not new in classical complexity theory. Earlier, Feigenbaum, Fortnow, Laplante, and Naik [13] applied an algorithmic argument to show that the multiple-block ordered search problem is hard to solve on a classical Turing machine using nonadaptive queries. Their proof, nonetheless, cannot be directly applied to the case of black-box quantum computation since their proof exploits the fact that a probabilistic polynomial-time Turing machine with polynomial advice can be simulated by a certain deterministic polynomial-time Turing machine with polynomial advice. Our technique developed in this paper, to the contrary, enables us to show a desired quantum query complexity lower bound for the multiple-block ordered search problem.

In the subsequent sections, we will give the formal definition to a black-box quantum computer and to multiple-block ordered search problems. We will also present an overview of our lower bound results before proving the main theorems.

2 A Model of Nonadaptive Query Computation

We formally describe a black-box model of quantum nonadaptive query computation. The reader may refer to [6] for the formal description of quantum "adaptive" query computation. In particular, we use a "truth-table" query model rather than the "parallel" query model given in [10,18] to simplify our algorithmic argument. Note that these two models are fundamentally equivalent in the classical setting since there is no timing problem as it occurs in the quantum case (see, e.g., [18] for more details). In our truth-table query model, all queries are made at once at the first phase of computation and the second phase leads to a desirable solution without any query. This truth-table query model can be seen as a special case of the parallel query model of Yamakami [18].

In the rest of this paper, we assume the reader's familiarity with the fundamental concepts in computational complexity theory and quantum computing. Hereafter, we fix our alphabet Σ to be $\{0,1\}$ for simplicity. Let $\mathbb{N}$ be the set of all natural numbers (i.e., nonnegative integers) and set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. For any two integers m, n with $m < n$, let $[m,n]_{\mathbb{Z}} = \{m, m+1, m+2, \ldots, n\}$. For any $n \in \mathbb{N}^+$ and $i \in [1, 2^n]_{\mathbb{Z}}$, let $\mathrm{bin}_n(i)$ denote the lexicographically ith binary string in Σ^n (e.g., $\mathrm{bin}_n(1) = 0^n$ and $\mathrm{bin}_n(2^n) = 1^n$). All logarithms are to base two and all polynomials have integer coefficients.

Fix N as a power of 2, say, $N = 2^n$ for a certain $n \in \mathbb{N}^+$. A *(black-box) problem*[1] F_N over Σ is a function that maps each instance f to its value $F_N(f)$,

[1] To simplify the later arguments, we consider only the case where N is a power of 2 although it is possible to discuss the general case of N's taking any integer.

where f is any function from Σ^n to Σ. For convenience, f is often identified with its characteristic sequence[2] of length N.

To solve such a problem F_N, we carry out a nonadaptive query quantum computation on a black-box quantum computer. A *nonadaptive black-box quantum computer* is described as a series of pairs of unitary operators, say $\{(U_n, V_n)\}_{n\in\mathbb{N}}$. Let n be any fixed size of our instances. The quantum computer (U_n, V_n) consists of three registers. The first register is used to generate query words, the second register is used to receive the oracle answers to these query words, and the third register is used to perform non-query computation. Assume that an instance $x \in \Sigma^N$ of the problem F_N is given to the computer as an oracle and all the registers are initially set to be the quantum state $|0\rangle|0\rangle|0\rangle$. The computer first changes $|0\rangle|0\rangle|0\rangle$ to the *prequery quantum state* $\sum_{i_1,\ldots,i_T} |i_1,\ldots,i_T\rangle|0\rangle|\phi_{i_1,\ldots,i_T}\rangle$ by applying the unitary operator U_n, where $T \in \mathbb{N}$ and $i_1,\ldots,i_T \in [1,N]_{\mathbb{Z}}$. We now prepare the unitary operator O_x that represents x. The application of O_x results in the *postquery quantum state* $\sum_{i_1,\ldots,i_T} |i_1,\ldots,i_T\rangle|x_{i_1}\ldots x_{i_T}\rangle|\phi_{i_1,\ldots,i_T}\rangle$. Finally, the computer applies the unitary operator V_n and then halts. When the computer halts, its output state becomes $V_nO_xU_n(|0\rangle|0\rangle|0\rangle)$. The first $|F_N(x)|$ qubits of the third register are measured on the computational basis to obtain the outcome of the computation.

The *ϵ-error bounded quantum nonadaptive query complexity* of the problem F_N, denoted by $Q_\epsilon^{tt}(F_N)$, is defined to be the minimal number T of the nonadaptive queries made by any nonadaptive black-box quantum computer with oracle x such that $F_N(x)$ is observed with error probability at most ϵ by the measurement of the output state. Throughout this paper, we restrict all the amplitudes of a bounded-error black-box quantum computer to the amplitude set $\{0, \pm 3/5, \pm 4/5, \pm 1\}$. This restriction does not affect our results since bounded-error quantum algorithms are known to be robust against the choice of an amplitude set [1].

In case where advice h_x is given as a supplemental input, a nonadaptive black-box quantum computer starts with the initial quantum state $|0\rangle|0\rangle|h_x\rangle$ instead of $|0\rangle|0\rangle|0\rangle$. We denote by $Q_\epsilon^{k,tt}(F_N)$ the *ϵ-error bounded quantum nonadaptive query complexity of F_N given advice of length k*. For convenience, we often suppress the subscript ϵ if $\epsilon = 1/3$.

We further expand the problem F_N into a "multiple-block" problem in the following section.

3 Multiple Block Problems

We first give a general scheme of how to expand a (black-box) problem F_N into a "multiple-block" problem $F_{M,N}$. Formally, for any numbers $n, M \in \mathbb{N}$ and $N = 2^n$, we define an *(black-box) M-block problem* $F_{M,N}$ as follows. Let $\Gamma_{M,N} = [1,M]_{\mathbb{Z}} \times \Sigma^n$, where the first part of $\Gamma_{M,N}$ indicates "block numbers" and the second part indicates "locations." An instance to the problem $F_{M,N}$ is a pair

[2] A *characteristic sequence* of f is $x_1x_2\cdots x_N$, where the ith bit x_i is the value $f(\mathrm{bin}_n(i))$ for each $i \in [1,N]_{\mathbb{Z}}$.

of an integer $i \in [1, M]_{\mathbb{Z}}$ and a function f from $\Gamma_{M,N}$ to Σ. We abbreviate $f(i, x)$ as $f_i(x)$. A problem $F_{M,N}$ is a function mapping (i, f) to its value $F_{M,N}(i, f)$. The function f is given as an oracle and i is given as an input to a black-box quantum computer. To access the value of f, we need to query a pair $(i, y) \in \Gamma_{M,N}$ (called a *query word*) to the oracle f. Our task is to compute $F_{M,N}(i, f)$ when f is given as an oracle and i is given as an input. Given such a problem $F_{M,N}$, the quantum computer starts with the initial quantum state $|0\rangle|0\rangle|i\rangle$, where i represents a block number, and attempts to compute the value $F_{M,N}(i, f)$, which depends only on f_i, by making queries to f given as an oracle. (If an advice string h is augmented, the initial quantum state must be $|0\rangle|0\rangle|i, h\rangle$.) Obviously, the (classical/quantum and adaptive/nonadaptive) query complexity for any multiple-block problem $F_{M,N}$ is at most N. Later, we identify f_i with its characteristic sequence of length N.

The *M-block ordered search problem* $G_{M,N}$, where $N = 2^n$, is formally defined as follows. The domain of $G_{M,N}$ is the set $\{(i, x_1 \cdots x_M) \in [1, M]_{\mathbb{Z}} \times \Sigma^{MN} \mid \forall j \exists s_j \forall k [\, (x_j)_k = 1 \text{ if } k \geq s_j \text{ and } (x_j)_k = 0 \text{ if } k < s_j]\}$, where each x_j is in Σ^N. Each s_j is called the *step* of x_j. The output value $G_{M,N}(i, x_1 x_2 \cdots x_M)$ is the ith step s_i. For a later use, we define $G_{M,N,p}$ for each $p \in [1, n]_{\mathbb{Z}}$ as follows. The domain of $G_{M,N,p}$ is the same as that of $G_{M,N}$ but the outcome $G_{M,N}(i, x_1 x_2 \cdots x_M)$ is the last p bits of s_i. Note that $G_{M,N,n} = G_{M,N}$.

To solve the multiple-block ordered search problem $G_{M,N}$, our nonadaptive black-box quantum computer (U_n, V_n) operates in the following fashion. Given a pair (i, x) of a number $i \in [1, M]_{\mathbb{Z}}$ and an MN-bit string $x = x_1 x_2 \cdots x_M$ (where each x_i is in Σ^N), (U_n, V_n) starts with a block number i and an advice string g (which is given independent of i) of length k and attempts to compute the value $G_{M,N}(i, x)$ with small error probability. It is desirable in practice to minimize the number of queries and also the length of an advice string.

4 Query Complexity Lower Bounds: Overview

This section presents an overview of the adaptive and nonadaptive query complexity bounds for the multiple-block ordered search problem $G_{M,N}$. This problem can separate the power of quantum adaptive query computation and that of quantum nonadaptive query computation.

For its quantum adaptive query complexity, the single-block ordered search problem $G_{1,N}$ is one of the well-studied problems. In this adaptive query case, a simple binary search algorithm provides a trivial adaptive query complexity upper bound of $\log N$. The lower bound of the adaptive query complexity $Q(G_{1,N})$ was explored in [11,12], and the $\Omega(\log N)$-lower bound was recently given by Ambainis [2] and Høyer, Neerbek, and Shi [14]. For the multiple-block problem $G_{M,N}$, a trivial query complexity upper bound is also $\log N$.

On the contrary, the nonadaptive query complexity has a trivial upper bound of $N - 1$ for the multiple-block problem $G_{1,N}$. Similarly, in the presence of advice of length k, the query complexity $Q^{k,tt}(G_{1,N})$ is upper-bounded by $N/2^k - 1$. As for the lower bound, using our algorithmic argument, we can show in Theorem

Table 1. Quantum nonadaptive query complexity bounds of $G_{1,N}$ and $G_{M,N}$

problem	$G_{1,N}$		$G_{M,N}$	
w/o advice	no advice	advice length k	no advice	advice length k
upper bound	$N-1$	$N/2^k-1$	$N-1$	$N/2^{\lfloor k/M \rfloor}-1$
lower bound	$\Omega(N)$	$\Omega(N)/2^{2k}$	$\Omega(N)$	$\Omega(p(N,M,k))$

2 a lower bound $Q^{k,tt}(G_{1,N}) \geq \Omega(N)/2^{2k}$, which almost matches the above trivial upper bound for $G_{1,N}$. Turning to the multiple-block problem $G_{M,N}$, we can show in Theorem 1 a lower bound $Q^{k,tt}(G_{M,N}) \geq \Omega(p(N,M,k))$, where $p(N,M,k) = \min\left\{\frac{N}{M^2\cdot 2^{(k+2)/M^{1/3}}}, \frac{M-M^{1/3}}{(2M^{1/3}\log M+k+2)^2}\right\}$. These two lower bounds of the nonadaptive query complexity will be proven in §5 and §6. Moreover, a large gap between $Q(G_{M,N})$ and $Q^{k,tt}(G_{M,N})$ will be used in §7 to separate adaptive and nonadaptive complexity classes.

The aforementioned upper and lower bounds of the quantum nonadaptive query complexity of the multiple-block and single-block ordered search problems are summarized in Table 1.

5 Query Complexity for Multiple Block Ordered Search

We demonstrate how to use an algorithmic argument to obtain a new query complexity lower bound for the multiple-block ordered search problem $G_{M,N}$. As a special case, the query complexity for the single-block ordered search problem will be discussed later in §6. Now, we prove the following lower bound of $Q^{k,tt}(G_{M,N})$ as a main theorem.

Theorem 1. $Q^{k,tt}(G_{M,N}) \geq \Omega\left(\min\left\{\frac{N}{M^2\cdot 2^{(k+2)/M^{1/3}}}, \frac{M-M^{1/3}}{(2M^{1/3}\log M+k+2)^2}\right\}\right)$.

Theorem 1 intuitively states that multiple-block ordered search requires a large number of nonadaptive queries even with the help of a relatively large amount of advice. To show the desired lower bound, we employ an algorithmic argument that revolves around the incompressibility of instances.

We first prove a key proposition from which our main theorem follows immediately. For convenience, for any constant $\epsilon \in [0, 1/2)$, we define $d(\epsilon) = 1/(2\epsilon)-1$ if $\epsilon > 0$ and $d(\epsilon) = 1$ otherwise. Letting c be any constant satisfying $0 < c < d(\epsilon)$, we further define $\epsilon' = (1+c)\epsilon$ and $C_\epsilon = (1-2\epsilon')^2/16$. For any string y and any number $a \in [0, |y|]_{\mathbb{Z}}$, $\mathrm{First}_a(y)$ ($\mathrm{Last}_a(y)$, resp.) denotes the first (last, resp.) a-bit segment of y. Clearly, $y = \mathrm{First}_{|y|-a}(y)\mathrm{Last}_a(y)$.

To describe the key proposition, we need the notion of the *weight* of a query word and the function $C_{U,V}$. Fix $n, M \in \mathbb{N}$ and $p \in [1,n]_{\mathbb{Z}}$ and set $N = 2^n$. Consider the multiple-block problem $G_{M,N,p}$. Assume that a nonadaptive black-box quantum computer (U, V) solves $G_{M,N,p}$ with error probability $\leq \epsilon$ with advice of length k using T nonadaptive queries. Let s be any string of length Mn and assume that $s = s_1 s_2 \cdots s_M$, where each s_i is the ith block segment of s with

$|s_i| = n$. Let f be its corresponding k-bit advice string. For any $i, j \in [1, M]_{\mathbb{Z}}$ and any $z \in \Sigma^n$, the *weight* of the query word (j, z), denoted $wt(i : j, z)$, is the sum of all squared magnitudes of amplitudes of $|\boldsymbol{y}\rangle|0\rangle|\phi_{i,f,\boldsymbol{y}}\rangle$ such that the list $\boldsymbol{y} = (y_1, \ldots, y_T)$ of query words contains (j, z) in the prequery quantum state $U(|0\rangle|0\rangle|i, f\rangle) = \sum_{\boldsymbol{y}} |\boldsymbol{y}\rangle|0\rangle|\phi_{i,f,\boldsymbol{y}}\rangle$. Moreover, for each $i, j \in [1, M]_{\mathbb{Z}}$ and $z' \in \Sigma^{n-p}$, let $wt_p(i : j, z')$ be the sum of all the values $wt(i : j, z)$ over all $z \in \Sigma^n$ satisfying $z' = \mathrm{First}_{n-p}(z)$. An index i is called *good* if $wt_p(i : i, \mathrm{First}_{n-p}(s_i)) > C_\epsilon$. Any index that is not good is called *bad*. Let l'_s denote the total number of good indices; i.e., $l'_s = |\{i \in [1, M]_{\mathbb{Z}} \mid wt_p(i : i, \mathrm{First}_{n-p}(s_i)) > C_\epsilon\}|$. Note that $0 \le l'_s \le M$. At length, the function $C_{U,V}$ is introduced in the following way:

$$C_{U,V}(M, N, k, p, s, l) = \begin{cases} \frac{C_\epsilon N}{M^2 2^{p+1+(k+2)/l}} & \text{if } l \le l'_s, \\ \frac{C_\epsilon (M-l)p^2}{(2l \log M + k + 2)^2} & \text{if } l > l'_s, \end{cases}$$

where $l \ge 1$. The key proposition below relates to a relationship between the query complexity and the function $C_{U,V}$.

Proposition 1. *Let* $\epsilon \in [0, 1/2)$ *and* $c \in (0, d(\epsilon))$ *and set* $\epsilon' = (1 + c)\epsilon$ *and* $C_\epsilon = (1 - 2\epsilon')^2/16$. *Let* $n, M \in \mathbb{N}$ *and* $p \in [1, n]_{\mathbb{Z}}$ *and set* $N = 2^n$. *If a quantum computer* (U, V) *solves* $G_{M,N,p}$ *with error probability* $\le \epsilon$ *using* T *nonadaptive queries and advice of length* k, *then* $T \ge \max_{1 \le l \le M} \min_{s \in \Sigma^{Mn}} \{C_{U,V}(M, N, k, p, s, l)\}$.

Clearly, Theorem 1 follows from Proposition 1 by setting $p = 1$ and $l = M^{1/3}$ since $Q^{k,tt}(G_{M,N,1}) \le Q^{k,tt}(G_{M,N})$ for any constant k in $\mathbb{N}$.

Now, we detail the proof of Proposition 1 by employing our algorithmic argument. Assume to the contrary that Proposition 1 fails. Let (U, V) be a nonadaptive black-box quantum computer that solves $G_{M,N,p}$ with error probability $\le \epsilon$ with T nonadaptive queries using advice of length k. By our assumption, there exists a number $l \in [1, M]_{\mathbb{Z}}$ such that $T < C_{U,V}(M, N, k, p, s, l)$ for all strings s in Σ^{Mn}. It follows by a simple calculation that, for each $s \in \Sigma^{Mn}$,

$$l(2 \log M - n + \log(T/C_\epsilon) + p + 1) + k + 2 < 0 \quad \text{if } l \le l'_s, \tag{1}$$

$$2l \log M + k + 2 - p\sqrt{C_\epsilon (M - l)/T} < 0 \quad \text{if } l > l'_s. \tag{2}$$

Our goal is to define a compression scheme E working on all strings in Σ^{Mn} such that (i) E is one-to-one and (ii) E is length-decreasing[3]. These two conditions clearly lead to a contradiction since any length-decreasing function from Σ^{Mn} to Σ^* cannot be one-to-one (by the pigeonhole principle). More precisely, we define an "encoding" of s, denoted $E(s)$. We first show that $|E(s)| < |s|$. To show the one-to-oneness of E, we construct a "generic" deterministic decoding algorithm that takes $E(s)$ and outputs s for any $s \in \Sigma^{Mn}$ since this algorithm guarantees the uniqueness of the encoding $E(s)$ of s.

Let s be any string in Σ^{Mn} and let f be its corresponding advice string of length k. We split our proof into the following two cases: (1) $l \le l'_s$ and (2) $l > l'_s$.

[3] A function f from Σ^n to Σ^* is called *length-decreasing* if $|f(x)| < |x|$ for all $x \in \Sigma^n$.

(Case 1: $l \leq l'_s$) The desired encoding $E(s)$ contains the following four items: (i) the advice string f, (ii) the $2l'_s \log M$-bit string encoding in double binary all l'_s good indices, (iii) a separator 01, and (iv) all the strings $e(i)$ for each $i \in [1, M]_{\mathbb{Z}}$, where $e(i)$ is defined as follows. In case where i is good, $e(i)$ is of the form $(k_i, \mathrm{Last}_p(s_i))$ with $k_i = |\{a \in \Sigma^{n-p} \mid wt_p(i : i, a) > C_\epsilon \text{ and } a < \mathrm{First}_{n-p}(s_i) \text{ (lexicographically)}\}|$. If i is bad, then $e(i) = s_i$. These four items are placed in $E(s)$ in order from (i) to (iv).

The following lemma shows that $\lceil \log(T/C_\epsilon) \rceil$ bits are sufficient to encode k_i.

Lemma 1. *For each good i, $k_i < T/C_\epsilon$.*

By Lemma 1, each pair $(k_i, \mathrm{Last}_p(s_i))$ requires at most $\lceil \log(T/C_\epsilon) \rceil + p$ bits. The total length of $E(s)$ is thus bounded above by:

$$\begin{aligned} |E(s)| &\leq k + 2l'_s \log M + 2 + l'_s(\log(T/C_\epsilon) + p + 1) + (M - l'_s)n \\ &\leq Mn + l(2\log M - n + \log(T/C_\epsilon) + p + 1) + k + 2 \ < \ Mn, \end{aligned}$$

where the second inequality is obtained from Eq.(1) and our assumption $l \leq l'_s$, and the last inequality comes from Eq.(1). Since $|s| = Mn$, it follows that $|E(s)| < |s|$.

We next show that the encoding $E(s)$ is uniquely determined from s. To show this, we give a deterministic decoding algorithm that extracts s from $E(s)$ for all $s \in \Sigma^{Mn}$ with $l'_s \geq l$. The desired decoding algorithm $\mathcal{A}$ is described as follows.

Decoding Algorithm $\mathcal{A}$: For each $i \in [1, M]_{\mathbb{Z}}$, we compute s_i in the following manner. First, check whether i is good by examining item (ii) of $E(s)$. If i is bad, then find $e(i) = s_i$ directly from item (iv). The remaining case is that i is good. Note that $wt_p(i : i, \mathrm{First}_{n-p}(s_i)) > C_\epsilon$ and $e(i) = (k_i, \mathrm{Last}_p(s_i))$. Define A_i to be the set of all $(n-p)$-bit strings a with $wt_p(i : i, a) > C_\epsilon$. Find the lexicographically (k_i+1)st string in A_i. Obviously, this string equals $\mathrm{First}_{n-p}(s_i)$ by the definition of k_i. Use $\mathrm{Last}_p(s_i)$ to obtain the desired string $s_i = \mathrm{First}_{n-p}(s_i)\mathrm{Last}_p(s_i)$. Finally, output the decoded string $s = s_1 s_2 \cdots s_M$.

Since $\mathcal{A}$ does not involve the computation of (U, V), it is easy to show that $\mathcal{A}$ correctly outputs s from $E(s)$.

(Case 2: $l > l'_s$) Different from case (1), the encoding $E(s)$ includes the following six items: (i) the advice string f, (ii) the $2l'_s \log M$-bit string that encodes in double binary all l'_s good indices, (iii) a separator 01, (iv) all the strings s_i for each good index i, (v) all the strings $\mathrm{First}_{n-p}(s_i)$ for each bad index i, and (vi) an additional string r of length $\leq p((M-l'_s) - \sqrt{C_\epsilon(M-l'_s)/T})$, which will be defined later. These items are placed in $E(s)$ orderly from (i) to (vi).

We begin with the estimation of the length of $E(s)$. By summing up all the items of $E(s)$, we can upper-bound its length $|E(s)|$ by:

$$\begin{aligned} |E(s)| &\leq k + 2l'_s \log M + 2 + l'_s n + (M - l'_s)(n - p) \\ &\quad + p((M - l'_s) - \sqrt{C_\epsilon(M - l'_s)/T}) \\ &\leq Mn + 2l \log M + k + 2 - p(\sqrt{C_\epsilon(M - l)/T}) \ < \ Mn, \end{aligned}$$

where the second inequality comes from our assumption $l > l'_s$ and the fact that the derivative of the function $\mathcal{F}(z) = -p(\sqrt{C_\epsilon(M-z)/T}) + 2z \log M$ satisfies $\mathcal{F}'(z) \geq 0$ for any $z \in (0, M)$ and the last inequality follows from Eq.(2). Therefore, we obtain the desired inequality $|E(s)| < |s|$.

We still remain to define the string r. To describe it, we need to search for indices of light query weight. The following procedure, called the *lightly weighted step search* (abbreviated LWSS), selects a series of steps of light query weight. As we will show later, this series of steps are redundant and therefore, we can eliminate them, causing the compression of s. Let m be the positive solution of the equation $(T/C_\epsilon)m^2 - (T/C_\epsilon - 1)\,m - (M - l'_s) = 0$. (In case where m is a non-integer, we round it down.)

Procedure LWSS: Let $R_1 = \emptyset$ and $L_1 = \{i \in [1, M]_{\mathbb{Z}} \mid i \text{ is bad}\}$. Repeat the following procedure by incrementing i by one until $i = m$. At round i, choose the lexicographically smallest index w_i in the difference $L_i - R_i$. Simulate U deterministically on input (w_i, f) to generate the prequery quantum state $|\gamma_f(w_i)\rangle$. For each bad index $j \in [1, M]_{\mathbb{Z}}$, compute the weight $wt_p(w_i : j, \mathrm{First}_{n-p}(s_j))$ in $|\gamma_f(w_i)\rangle$. Define $R_{i+1} = R_i \cup \{w_i\}$ and $L_{i+1} = L_i \cap \{j \in [1, M]_{\mathbb{Z}} \mid wt_p(w_i : j, \mathrm{First}_{n-p}(s_j)) < C_\epsilon/m\}$. Finally, set $W = L_m$. Output all the elements in W.

We can prove the following lemma regarding LWSS.

Lemma 2. LWSS *produces a unique series of* m *distinct indices* $w_1, w_2, \ldots, w_m$ *such that, for any pair* (i, j) *of integers with* $1 \leq i, j \leq m$, *if* $i < j$ *then* $wt_p(w_i : w_j, \mathrm{First}_{n-p}(s_{w_j})) < C_\epsilon/m$.

The key element r in item (vi) is defined as follows. Let v_i be the lexicographically ith element in the set $\{j \in [1, M]_{\mathbb{Z}} \mid j \text{ is bad and } j \notin W\}$ and let r_i be the last p bits of s_{v_i}. The string $r = r_1 r_2 \cdots r_{M-l'_s-m}$ constitutes item (vi). Its length $|r|$ is clearly at most $p(M - l'_s - \sqrt{C_\epsilon(M - l'_s)/T})$ since $m = ((T/C_\epsilon - 1) + \sqrt{4(T/C_\epsilon)(M - l'_s) + (T/C_\epsilon - 1)^2})/(2T/C_\epsilon) \geq \sqrt{C_\epsilon(M - l'_s)/T}$.

Next, we want to show the uniqueness of our encoding $E(s)$ by constructing its decoding algorithm. First, we check which index i is good by simply examining item (ii). For any good index i, we immediately obtain s_i from item (iv). When i is bad, however, we obtain only $\mathrm{First}_{n-p}(s_i)$ from item (v). To obtain the last p bits of s_i, we need to exploit item (vi) of $E(s)$ and simulate (U, V) in a deterministic fashion. Since we cannot use the oracle $G_{M,N,p}$, we need to substitute its true oracle answers with their approximated values. The desired decoding algorithm $\mathcal{B}$ is given as follows.

Decoding Algorithm $\mathcal{B}$: 1) For any good index $i \in [1, M]_{\mathbb{Z}}$, obtain s_i directly from item (iv) of $E(s)$. For the other indices, run LWSS to compute $W = \{w_1, \ldots, w_m\}$. For any bad index e outside of W, item (vi) provides $\mathrm{Last}_p(s_e)$. Combining it with $\mathrm{First}_{n-p}(s_e)$ from item (v), we obtain s_e.

2) Let e be any bad index in W. First, obtain $\mathrm{First}_{n-p}(s_e)$ from item (v). The remaining part, $\mathrm{Last}_p(s_e)$, is obtained as follows. Repeat the following procedure starting at round 1 up to m. At round i $(1 \leq i \leq m)$, assume that the last p bits of $s_{w_1}, \ldots, s_{w_{i-1}}$ have been already obtained. Simulate U deterministically on input

(w_i, f) to generate the prequery quantum state $|\gamma_f(w_i)\rangle = \sum_{\boldsymbol{y}} |\boldsymbol{y}\rangle|0\rangle|\phi_{w_i,f,\boldsymbol{y}}\rangle$. Transform $|\gamma_f(w_i)\rangle$ into the state $|\gamma_f(w_i)^r\rangle = \sum_{\boldsymbol{y}} |\boldsymbol{y}\rangle|u_{1,\boldsymbol{y}}u_{2,\boldsymbol{y}}\cdots u_{T,\boldsymbol{y}}\rangle|\phi_{w_i,f,\boldsymbol{y}}\rangle$ using the string r, where each bit $u_{j,\boldsymbol{y}}$ is defined below. Choose any list $\boldsymbol{y}$ of query words. Note that $\boldsymbol{y}$ is of the form $(y_1, y_2, \ldots, y_T)$. Let $j \in [1,T]_{\mathbb{Z}}$ and assume that y_j is of the form (a, vz), where $a \in [1,M]_{\mathbb{Z}}$, $v \in \Sigma^{n-p}$, and $z \in \Sigma^p$. For simplicity, write u_j for $u_{j,\boldsymbol{y}}$.

a) If v is lexicographically smaller (larger, rep.) than $\mathrm{First}_{n-p}(s_a)$, then let $u_j = 0$ ($u_j = 1$, resp.). Next, assume $v = \mathrm{First}_{n-p}(s_a)$. In case where either a is good or $a \notin W$, obtain $\mathrm{Last}_p(s_a)$ from items (iv) and (vi) and let $u_j = 1$ if $z \geq \mathrm{Last}_p(s_a)$ and let $u_j = 0$ otherwise.

b) Consider the case where $a \in W$. Let b be the index satisfying $a = w_b$. There are two cases to consider.

b-i) Assume that $b < i$. Note that $\mathrm{Last}_p(s_a)$ has been obtained at an earlier round. Define $u_j = 1$ if $z \geq \mathrm{Last}_p(s_a)$ and $u_j = 0$ otherwise.

b-ii) If $b \geq i$, then set $u_j = 0$. In particular, if $b > i$, then Lemma 2 implies that $wt_p(w_i : w_b, \mathrm{First}_{n-p}(s_{w_b})) < C_\epsilon/m$ and $wt_p(w_i : w_i, \mathrm{First}_{n-p}(s_{w_i})) \leq C_\epsilon$.

3) Simulate V deterministically on input $|\gamma_f(w_i)^r\rangle$. Find its output that is obtained with probability $\geq 1/2$. Such a string must be $\mathrm{Last}_p(s_{w_i})$. With the known string $\mathrm{First}_{n-p}(s_i)$, this gives the entire string s_i, as required.

4) Output the decoded string $s = s_1 s_2 \cdots s_M$.

We need to verify that the decoding algorithm $\mathcal{B}$ correctly extracts s from $E(s)$. If i is good, then s_i can be correctly obtained from item (iv). If i is bad and not in W, then $\mathcal{B}$ computes $\mathrm{Last}_p(s_i)$. Henceforth, we assume that i is bad and in W. Let j be the index satisfying $i = w_j$. The operator U on input (w_j, f) generates the prequery quantum state $|\gamma_f(w_j)^r\rangle$. We need to prove that our approximation of the true oracle answers from $G_{M,N,p}$ suffices for the correct simulation of (U, V). Let $|\gamma_f(w_j)^G\rangle$ be the true postquery quantum state; namely, $\sum_{\boldsymbol{y}} |\boldsymbol{y}\rangle|G_{M,N,p}(y_1)\cdots G_{M,N,p}(y_T)\rangle|\phi_{w_j,f,\boldsymbol{y}}\rangle$, where $\boldsymbol{y} = (y_1, \ldots, y_T)$. Now, we claim that $|\gamma_f(w_j)^r\rangle$ is close to $|\gamma_f(w_j)^G\rangle$.

Lemma 3. $\| |\gamma_f(w_j)^r\rangle - |\gamma_f(w_j)^G\rangle \| \leq 2\sqrt{C_\epsilon}$.

Lemma 3 is shown as follows. The term $\| |\gamma_f(w_j)^r\rangle - |\gamma_f(w_j)^G\rangle \|^2$ is bounded above by $2\sum_{b:b\geq j} wt_p(w_j : w_b, \mathrm{First}_{n-p}(s_{w_b}))$, which is $\leq 2(|W| \cdot C_\epsilon/m + C_\epsilon)$ by Lemma 2. This last term clearly equals $4C_\epsilon$.

Using the approximated oracle answers $|\gamma_f(w_j)^r\rangle$, the operator V produces a wrong solution with probability at most $2\sqrt{C_\epsilon} + \epsilon \leq 1/2 - c\epsilon$. In other words, V outputs the correct string $\mathrm{Last}_p(s_{w_j})$ with probability at least $1/2 + c\epsilon$. By simulating V in a deterministic way, we can correctly find the correct outcome of V. Therefore, $\mathcal{B}$ correctly outputs s from $E(s)$. This ends case (2).

Combining cases (1) and (2), we conclude that E is a length-decreasing one-to-one function from Σ^{Mn} to Σ^*, contradicting the pigeonhole principle. This completes the proof of Proposition 1.

6 Query Complexity for Single Block Ordered Search

The single-block ordered search problem $G_{1,N}$ has been extensively studied in the literature for the lower bound of its quantum adaptive query complexity. Upon quantum nonadaptive query computation, we demonstrate a lower bound of the query complexity $Q^{k,tt}(G_{1,M})$ in the presence of advice. We follow the notations introduced in §5.

Theorem 2. *Let $n \in \mathbb{N}$ and set $N = 2^n$. For any $\epsilon \in [0, 1/2)$ and any $c \in (0, d(\epsilon))$, $Q_\epsilon^{k,tt}(G_{1,N}) \geq C_\epsilon N/2^{2k+2}$, where $\epsilon' = (1+c)\epsilon$ and $C_\epsilon = (1-2\epsilon')^2/16$.*

For Theorem 2, it suffices to show that $Q_\epsilon^{k,tt}(G_{1,N,p}) \geq C_\epsilon N/2^{2k+2}$ for $p = k+1$. For the special case $k = 0$ and $\epsilon = 1/3$, Theorem 2 gives a lower bound $Q^{tt}(G_{1,N}) \geq \Omega(N)$, which is *optimal* if we ignore its constant multiplicative factor since $Q^{tt}(G_{1,N}) \leq N - 1$.

For the single-block ordered search problem, we can also employ a quantum adversary argument to prove its quantum nonadaptive query complexity. Using an inner product method of Høyer et al. [14], we can obtain an $\Omega(N)$-lower bound of $Q^{k,tt}(G_{1,N})$. Note that it is not clear whether this inner product method can be extended to the multiple-block ordered search problem.

7 Other Applications of an Algorithmic Argument

We have shown in the previous sections how our algorithmic argument proves query complexity lower bounds. Hereafter, we apply our algorithmic argument to two notions of computational complexity theory: quantum truth-table reducibility and quantum truth-table autoreducibility.

The first example is a nonadaptive oracle separation between P and BQP/poly. To deal with the uniform class BQP, we introduce a "uniform" model of quantum truth-table query computation by simply replacing a black-box quantum computer (U, V) with a pair (M, N) of polynomial-time multi-tape well-formed quantum Turing machines (QTMs, in short). The relativized complexity class BQP_{tt}^A relative to oracle A is then defined as the collection of all sets recognized by pairs of polynomial-time QTMs in a manner similar to a black-box quantum computer with oracle A. The class $\mathrm{BQP}_{tt}^A/\mathrm{poly}$ in addition uses polynomial advice. Note that $\mathrm{BQP}_{tt}^A \subseteq \mathrm{BQP}^A$ for any oracle set A.

Applying the result in §5, we can show the following theorem.

Theorem 3. *There is a recursive oracle A such that $\mathrm{P}^A \not\subseteq \mathrm{BQP}_{tt}^A/\mathrm{poly}$.*

As the second example, we focus our interest on the notion of autoreducible sets. After Trakhtenbrot [17] brought in the notion of autoreduction in recursion theory, the autoreducible sets have been studied in, e.g., program verification theory. We define the notion of a BQP-*tt-autoreducible* set, which is obtained by replacing a polynomial-time PTM of the classical definition of "nonadaptive BPP-autoreducibility" in [13] by a pair of polynomial-time QTMs (M, N), provided that any prequery quantum state produced by M on each input x does

not include the query word x with nonzero amplitude. Let $\mathrm{BQP}_{tt}/\mathrm{poly}\text{-}\mathrm{AUTO}$ be the class of all BQP-tt-autoreducible sets with polynomial advice.

Similarly to Proposition 1, we can show the following statement.

Theorem 4. $\mathrm{BPP}\text{-}\mathrm{AUTO} \not\subseteq \mathrm{BQP}_{tt}/\mathrm{poly}\text{-}\mathrm{AUTO}$.

We apply our algorithmic argument to show the following, which is incomparable to the corresponding results in [7,16].

Theorem 5. $\mathrm{ESPACE} \not\subseteq \mathrm{BQP}_{tt}/\mathrm{poly}\text{-}\mathrm{AUTO}$.

References

1. L. M. Adleman, J. DeMarrais and M. A. Huang. Quantum computability. *SIAM J. Comput.* **26** (1997) 1524–1540.
2. A. Ambainis. A better lower bound for quantum algorithms searching an ordered list. In *Proc. 40th FOCS,* pp.352–357, 1999.
3. A. Ambainis. Quantum lower bounds by quantum arguments. *J. Comput. System Sci.* **64** (2002) 750–767.
4. A. Ambainis. Polynomial degree vs. quantum query complexity. In *Proc. 44th FOCS,* pp.230–240, 2003.
5. H. Barnum, M. Saks, and M. Szegedy. Quantum query complexity and semidefinite programming. In *Proc. 18th CCC,* pp.179–193, 2003.
6. R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM* **48** (2001) 778–797.
7. R. Beigel and J. Feigenbaum. On being incoherent without being very hard. *Comput. Complex.* **2** (1992) 1–17.
8. C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.* **26** (1997) 1510–1523.
9. M. Blum and S. Kannan, Designing programs that check their work, In *Proc. 21st STOC,* pp.86–97, 1989.
10. H. Buhrman and W. van Dam. Bounded quantum query complexity. In *Proc. 14th CCC,* pp.149–157, 1999.
11. H. Buhrman and R. de Wolf. A lower bound of quantum search for an ordered list. *Inform. Process. Lett.* **70** (1999) 205–209.
12. E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. A limit on the speed of quantum computation for insertion into an ordered list. See ArXiv quant-ph/9812057.
13. J. Feigenbaum, L. Fortnow, S. Laplante, and A. Naik. On coherence, random-self-reducibility, and self-correction. *Comput. Complex.* **7** (1998) 174–191.
14. P. Høyer, J. Neerbek, and Y. Shi. Quantum lower bound of ordered searching, sorting, and element distinctness. *Algorithmica* **34** (2002) 429–448.
15. S. Laplante and F. Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. To appear in *Proc. 19th CCC,* 2004.
16. H. Nishimura and T. Yamakami. Polynomial time quantum computation with advice. *Inform. Process. Lett.* **90** (2004) 195–204.
17. B. Trakhtenbrot. On autoreducibility. *Doklady Akad. Nauk. SSSR* **192** (1970) 1224–1227. English translation in *Soviet Math. Dokl.* **11** (1970) 814–817.
18. T. Yamakami. Analysis of quantum functions. *International Journal of Foundations of Computer Science* **14** (2003) 815–852.

Universal Test for Quantum One-Way Permutations

Akinori Kawachi[1,2], Hirotada Kobayashi[1], Takeshi Koshiba[1,3], and Raymond H. Putra[1,2]

[1] ERATO Quantum Computation and Information Project, Japan Science and Technology Agency, Tokyo/Kyoto, Japan
[2] Graduate School of Informatics, Kyoto University, Kyoto, Japan
[3] Secure Computing Laboratory, Fujitsu Laboratories Ltd., Kawasaki, Japan
{kawachi,hirotada,koshiba,raymond}@qci.jst.go.jp

Abstract. The next bit test was introduced by Blum and Micali and proved by Yao to be a universal test for cryptographic pseudorandom generators. On the other hand, no universal test for the cryptographic one-wayness of functions (or permutations) is known, though the existence of cryptographic pseudorandom generators is equivalent to that of cryptographic one-way functions. In the quantum computation model, Kashefi, Nishimura and Vedral gave a sufficient condition of (cryptographic) quantum one-way permutations and conjectured that the condition would be necessary. In this paper, we relax their sufficient condition and give a new condition that is necessary and sufficient for quantum one-way permutations. Our condition can be regarded as a universal test for quantum one-way permutations, since our condition is described as a collection of stepwise tests similar to the next bit test for pseudorandom generators.

1 Introduction

One-way functions are functions f such that, for each x, $f(x)$ is efficiently computable but, only for a negligible fraction of y, $f^{-1}(y)$ is computationally tractable. While the modern cryptography depends heavily on one-way functions, the existence of one-way functions is one of the most important open problems in theoretical computer science. Two of the most famous candidates for one-way functions are the RSA function and the discrete logarithm function, both of which are no longer one-way under the existence of quantum mechanical computers [13]. Nonetheless, some cryptographic applications based on quantum one-way functions have been considered (see, e.g., [1,6]).

As a cryptographic primitive other than one-way functions, pseudorandom generators have been studied well. Blum and Micali [3] proposed how to construct pseudorandom generators from one-way permutations and introduced the next bit test for pseudorandom generators. (They actually constructed a pseudorandom generator assuming the hardness of the discrete logarithm problem.) Yao [14] proved that the next bit test is a universal test for pseudorandom generators,

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 839–850, 2004.
© Springer-Verlag Berlin Heidelberg 2004

which accomplished the Blum–Micali's construction paradigm of pseudorandom generators from one-way permutations. In the case of pseudorandom generators based on one-way permutations, the next bit unpredictability can be proved by using the hard-core predicates for one-way permutations. After that, Goldreich and Levin [7] showed that there exists a hard-core predicate for any one-way function (and also permutation) and Håstad *et al.* [10] showed that the existence of pseudorandom generators is equivalent to that of one-way functions.

Yao's result on the universality of the next bit test assumes that any bits appeared in pseudorandom bits are computationally unbiased. Schrift and Shamir [12] extended Yao's result to the biased case and proposed universal tests for nonuniform distributions. On the other hand, no universal test for the one-wayness of a function (or a permutation) is known, although pseudorandom generators and one-way functions (or permutations) are closely related.

In the quantum computation model, Kashefi, Nishimura and Vedral [11] gave a necessary and sufficient condition for the existence of *worst-case* quantum one-way permutations. They also considered the *cryptographic (i.e., average-case)* quantum one-way permutations and gave a sufficient condition of (cryptographic) quantum one-way permutations. They also conjectured that the condition would be necessary. Their conditions are based on the efficient implementability of reflection operators about some class of quantum states. Note that the reflection operators are successfully used in the Grover's algorithm [9] and the quantum amplitude amplification technique [4]. To obtain a sufficient condition of cryptographic quantum one-way permutations, a notion of "pseudo identity" operators was introduced [11]. Since the worst-case hardness of reflection operators is concerned with the worst-case hardness of the inversion of the permutation f, we need some technical tool with which the inversion process of f becomes tolerant of some computational errors in order to obtain a sufficient condition of cryptographic quantum one-way permutations. Actually, pseudo identity operators permit of *exponentially* small errors during the inversion process [11].

In this paper, we give a necessary and sufficient condition of cryptographic quantum one-way permutations by relaxing the sufficient condition in [11]. We incorporate their basic ideas with the universal hashing technique [5] in order to obtain a technical tool to permit of *polynomially* small errors during the inversion process. Roughly speaking, pseudo identity operators are close to the identity operator in a sense. The similarity is defined by an intermediate notion between the statistical distance and the computational distance. In [11], it is "by upper-bounding the similarity" that the sufficient condition of cryptographic quantum one-way permutations was obtained. By using the universal hashing technique, we can estimate the expectation of the similarity and then handle polynomially small errors during the inversion of the permutation f.

Moreover, our necessary and sufficient condition of quantum one-way permutations can be regarded as a universal test for the quantum one-wayness of permutations. To discuss universal tests for the one-wayness of permutations, we briefly review the universality of the next bit test for pseudorandom generators. Let $g(x)$ be a length-regular deterministic function such that $g(x)$ is of

length $\ell(n)$ for any x of length n. The universality of the next bit test says that we only have to check a collection of stepwise polynomial-time tests $T_1, ..., T_{\ell(n)}$ instead of considering all the polynomial-time tests that try to distinguish the truly random bits from output bits of g. Here, each T_i is the test whether, given the $(i-1)$-bits prefix of $g(x)$ (and the value of $\ell(|x|)$), the i-th bit of $g(x)$ is predictable or not with probability non-negligibly higher than 1/2. Our necessary and sufficient condition of quantum one-way permutations says that the quantum one-wayness of a given permutation f can be checked by a collection of stepwise tests $T'_1, ..., T'_n$ instead of considering all the tests of polynomial-size quantum circuit. Here, each T'_i is the test whether, given some quantum state q_{i-1} that can be defined by using the $(i-1)$-bits prefix of $f(x)$, some other quantity t_i is computable with polynomial-size quantum circuit or not and the next state q_i can be determined from q_{i-1} and t_i. In this sense, our universal test for quantum one-way permutations is analogous to the universal test (i.e., the next bit test) for pseudorandom generators.

2 Preliminaries

We say that a unitary operator (on n qubits) is *easy* if there exists a quantum circuit implementing U of size polynomial in n and a set $\mathcal{F}$ of unitary operators is *easy* if every $U \in \mathcal{F}$ is easy. Throughout this paper, we assume that $f : \{0,1\}^* \to \{0,1\}^*$ is a length-preserving permutation unless otherwise stated. Namely, for any $x \in \{0,1\}^n$, $f(x)$ is an n-bit string and the set $\{f(x) : x \in \{0,1\}^n\}$ is of cardinality 2^n for every n. First, we mention some useful operators in describing the previous and our results. The tagging operators O_j are defined as follows:

$$O_j|x\rangle|y\rangle = \begin{cases} -|x\rangle|y\rangle & \text{if } f(y)_{(2j,2j+1)} = x_{(2j,2j+1)} \\ |x\rangle|y\rangle & \text{if } f(y)_{(2j,2j+1)} \neq x_{(2j,2j+1)} \end{cases}$$

where $y_{(i,j)}$ denotes the substring from the i-th bit to the j-th bit of the bit string y. Note that these unitary operators O_j are easy. Next, we consider the reflection operators. A reflection operator with respect to a quantum state $|\psi\rangle$ is defined by

$$S_{|\psi\rangle} = 2|\psi\rangle\langle\psi| - I.$$

In particular, we consider the operators $Q_j(f)$ defined as follows:

$$Q_j(f) = \sum_{x \in \{0,1\}^n} |x\rangle\langle x| \otimes S_{|\psi_{j,x}\rangle},$$

where $|\psi_{j,x}\rangle = \frac{1}{\sqrt{2^{n-2j}}} \sum_{y:f(y)_{(1,2j)}=x_{(1,2j)}} |y\rangle$. We allow a bit abuse of the term "reflection operator" and may also mention these $Q_j(f)$'s as reflection operators. The notation $Q_j(f)$ is often abbreviated to Q_j. In general, reflection operators are commonly and successfully used in the Grover's algorithm [9] and the quantum amplitude amplification technique [4].

Theorem 1. (Kashefi, Nishimura and Vedral [11]) *Let* $f : \{0,1\}^n \to \{0,1\}^n$ *be a permutation. Then* f *is worst-case quantum one-way if and only if the set* $\mathcal{F}_n = \{Q_j(f)\}_{j=0,1,\ldots,\frac{n}{2}-1}$ *of unitary operators is not easy.*

As a part of the proof of Theorem 1, Kashefi, Nishimura and Vedral [11] give a quantum algorithm (we call Algorithm INV in what follows) computing f^{-1} by using unitary operators O_j and Q_j. The initial input state to INV is assumed to be $\frac{1}{\sqrt{2^n}}|x\rangle \sum_{y\in\{0,1\}^n} |y\rangle$, where INV tries to compute $f^{-1}(x)$. Then INV performs the following steps:

foreach $j = 0$ to $\frac{n}{2} - 1$
(step W.j.1) Apply O_j to the first and the second registers;
(step W.j.2) Apply Q_j to the first and the second registers.

After each step, we have the following:

$$\begin{bmatrix}\text{the state after}\\ \text{step W.j.1}\end{bmatrix} = \frac{2^j}{\sqrt{2^n}}|x\rangle \left(\sqrt{2^{n-2j}}|\psi_{j,x}\rangle - 2 \sum_{y:f(y)_{(1,2j+2)}=x_{(1,2j+2)}} |y\rangle \right);$$

$$\begin{bmatrix}\text{the state after}\\ \text{step W.j.2}\end{bmatrix} = \frac{2^{j+1}}{\sqrt{2^n}}|x\rangle \sum_{y:f(y)_{(1,2j+2)}=x_{(1,2j+2)}} |y\rangle.$$

Before reviewing the known sufficient condition of cryptographic quantum one-way permutations, we define two types of cryptographic "one-wayness" in the quantum computational setting.

Definition 1. A permutation f is *weakly quantum one-way* if the following conditions are satisfied:

1. f can be computed by a polynomial size quantum circuit (and whenever inputs are classical the corresponding outputs must be classical).
2. There exists a polynomial $p(\cdot)$ such that for every polynomial size quantum circuit A and all sufficiently large n's, $\Pr[A(f(U_n)) \neq U_n] > \frac{1}{p(n)}$, where U_n is the uniform distribution over $\{0,1\}^n$.

Definition 2. A permutation f is *strongly quantum one-way* if the following conditions are satisfied:

1. f can be computed by a polynomial size quantum circuit (and whenever inputs are classical the corresponding outputs must be classical).
2. For every polynomial size quantum circuit A and every polynomial $p(\cdot)$ and all sufficiently large n's, $\Pr[A(f(U_n)) = U_n] < \frac{1}{p(n)}$.

As in the classical one-way permutations, we can show that the existence of weakly quantum one-way permutations is equivalent to that of strongly quantum one-way permutations (see, e.g., [8]). Thus, we consider the weakly quantum one-way permutations in this paper. While Theorem 1 is a necessary and sufficient condition of *worst-case* quantum one-way permutations, Kashefi, Nishimura and Vedral [11] also gave a sufficient condition of *cryptographic* quantum one-way permutations by using the following notion.

Definition 3. Let $d(n) \geq n$ be a polynomial in n and J_n be a $d(n)$-qubit unitary operator. J_n is called $(a(n), b(n))$-pseudo identity if there exists a set $X_n \subseteq \{0,1\}^n$ such that $|X_n|/2^n \leq b(n)$ and for any $z \in \{0,1\}^n \setminus X_n$,

$$|1 - (\langle z|_1\langle 0|_2) J_n (|z\rangle_1|0\rangle_2)| \leq a(n),$$

where $|z\rangle_1$ is the n-qubit basis state (in computational basis) for each z and $|0\rangle_2$ corresponds to the ancillae of $d(n) - n$ qubits.

The closeness between a pseudo identity operator and the identity operator is measured by a pair of parameters $a(n)$ and $b(n)$. The first parameter $a(n)$ is a measure of a statistical property and the second one $b(n)$ is a measure of a computational property. Note that we do not care where each $z \in X_n$ is mapped by the pseudo identity operator J_n. While we will give a necessary and sufficient condition of quantum one-way permutations by using the notion of pseudo identity, we introduce a new notion, which may be helpful to understand intuitions of our and previous conditions, in the following.

Definition 4. Let $d'(n) \geq n$ be a polynomial in n and P_n be a $d'(n)$-qubit unitary operator. P_n is called $(a(n), b(n))$-pseudo reflection (with respect to $|\psi(z)\rangle$) if there exists a set $X_n \subseteq \{0,1\}^n$ such that $|X_n|/2^n \leq b(n)$ and for any $z \in \{0,1\}^n \setminus X_n$,

$$\left| 1 - \left(\langle z|_1\langle w|_2 \Big(\sum_{y\in\{0,1\}^n} |y\rangle\langle y|_1 \otimes S_{|\psi(y)\rangle_2} \Big) \langle 0|_3 \right) P_n (|z\rangle_1|w\rangle_2|0\rangle_3) \right| \leq a(n).$$

The above definition of pseudo reflection operators may look complicated. Fig. 1 illustrates a geometrical intuition that may be helpful to understand the idea of pseudo reflection operators. The figure on the left shows that if x is not in the set X_n, then the image of the state $|x\rangle|\psi\rangle|0\rangle$ by the pseudo reflection operator P_n must be within the $a(n)$-ball in the ℓ_1-norm centered by the ideally reflected state (denoted by a vector in dashed line). We do not care the image of the state $|x\rangle|\psi\rangle|0\rangle$ by the pseudo reflection operator P_n if x is in X_n (the figure on the right).

Let J_n be a $d(n)$-qubit $(a(n), b(n))$-pseudo identity operator. Then $(I_n \otimes J_n)^\dagger (Q_j \otimes I_{d(n)-n})(I_n \otimes J_n)$ is a $(d(n)+n)$-qubit $(a'(n), b'(n))$-pseudo reflection operator with respect to $|\psi_{j,x}\rangle$, where $a'(n) \leq 2a(n)$ and $b'(n) \leq 2b(n)$. These estimations of $a'(n)$ and $b'(n)$ are too rough to obtain a necessary and sufficient condition. Rigorously estimating these parameters is a main technical issue in this paper.

Theorem 2. (Kashefi, Nishimura and Vedral [11]) *Let f be a permutation that can be computed by a polynomial-size quantum circuit. If f is not (weakly) quantum one-way, then for any polynomial p and infinitely many n, there exist a polynomial $r_p(n)$ and an $r_p(n)$-qubit $(1/2^{p(n)}, 1/p(n))$-pseudo identity operator J_n such that the family of pseudo reflection operators $\mathcal{F}_{p,n}(f) = \{(I_n \otimes J_n)^\dagger (Q_j(f) \otimes I_{r_p(n)-n})(I_n \otimes J_n)\}_{j=0,1,\ldots,\frac{n}{2}-1}$ is easy.*

They conjectured that the converse of Theorem 2 should still hold and proved a weaker version of the converse as follows [11].

Theorem 3. [11] *Let f be a permutation that can be computed by a polynomial-size quantum circuit. If for any polynomial p and infinitely many n there exist a polynomial $r_p(n)$ and an $r_p(n)$-qubit $(1/2^{p(n)}, p(n)/2^n)$-pseudo identity operator J_n such that the family of pseudo reflection operators $\mathcal{F}_{p,n}(f) = \{(I_n \otimes J_n)^\dagger (Q_j(f) \otimes I_{r_p(n)-n})(I_n \otimes J_n)\}_{j=0,1,\ldots,\frac{n}{2}-1}$ is easy, then f is not (weakly) quantum one-way.*

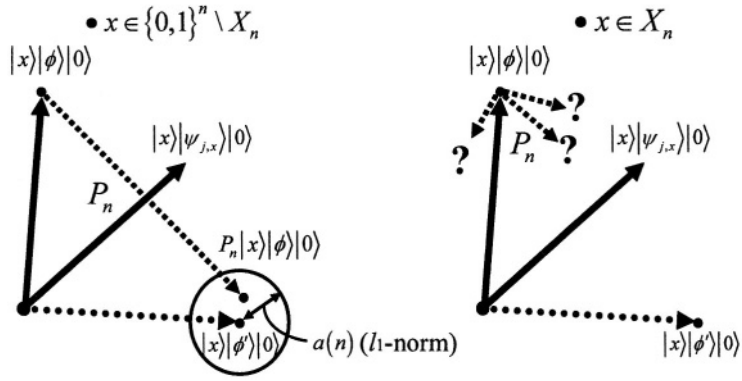


Fig. 1. Pseudo reflection operator

We mention why it is difficult to show the converse of Theorem 2. To prove it by contradiction, all we can assume is the existence of a pseudo identity operator. This means that we cannot know how the pseudo identity operator is close to the identity operator. To overcome this difficulty, we introduce a randomization technique and estimate the expected behavior of the pseudo identity operator. (Note that the converse looks difficult to be proved without some randomization techniques.) Together with the randomization technique, we give a necessary and sufficient condition of the existence of quantum one-way permutations in terms of reflection operators.

3 Universal Hashing Operator

We prepare a technical tool, which may be useful by itself. In general, random permutations can be a useful tool to the analysis of randomized algorithm. In the case of quantum computation, the efficient and faithful implementation of random permutations seems to be hard to realize. Instead, we introduce the universal hashing technique [5] to quantum computing and settle the conjecture by using the efficient implementability of the universal hash functions and the similar property to random permutations.

Let $r_1, r_2 \in \mathrm{GF}(2^n)$ and h_{r_1,r_2} be a function from $\mathrm{GF}(2^n)$ to $\mathrm{GF}(2^n)$ such that $h_{r_1,r_2}(x) = r_1 x + r_2$, where the addition and the multiplication are over $\mathrm{GF}(2^n)$ and their implementation can be realized over the ring of polynomials in one variable. Let $\mathcal{R} = \{h_{r_1,r_2} : r_1, r_2 \in \mathrm{GF}(2^n) \text{ with } r_1 \neq 0\}$. We identify $\mathrm{GF}(2^n)$ with $\{0,1\}^n$ by considering the standard conversion, and regard $\mathcal{R}$ as a family of functions from $\{0,1\}^n$ to $\{0,1\}^n$. Note that any function in $\mathcal{R}$ is a permutation but it is generally called a "hash" function because of the historical and conventional reason. Then, $\mathcal{R}$ is *2-univeral,* namely, for any $x_1 \neq x_2 \in \{0,1\}^n$ and for h chosen uniformly at random from $\mathcal{R}$, $\Pr[h(x_1) = h(x_2)] = 1/2^n$. In the quantum computation, the following operation is easy: $|z\rangle|0\rangle|0\rangle \mapsto \sum_{r_1,r_2\in\{0,1\}^n} |h_{r_1,r_2}(z)\rangle|z\rangle|r_1,r_2\rangle$. We call it *hashing* (with swap) operator and denote it by R_n. Since we use R_n only in the form of $R_n^\dagger(U \otimes I_{3n})R_n$ for some unitary operator U on the first n-qubit register, the swapped registers go back to the original positions. A few notes are in order:

- Instead of using the hashing operator R_n, we can choose polynomially many pairs of (r_1, r_2) beforehand and consider (r_1, r_2) being fixed during quantum computation. Namely, we can consider that the whole computation is performed by choosing a quantum circuit randomly and then feeding a given input to the quantum circuit. For simplicity, while we take this manner, we still describe the hashing computation just like the unitary operator. (Also note that the whole process can be done by unitary operators.)
- As mentioned above, we use R_n only in the form of $R_n^\dagger(U \otimes I_{3n})R_n$. We stress that the third register of R_n is discarded after each two applications of $R_n^\dagger(U \otimes I_{3n})R_n$. Every time we perform the first R_n of the four R_n's in this pair, we must prepare fresh ancillae of $2n$ qubits for the third register. Note that, if the total number of such executions is polynomially bounded, so is the total number of ancillae qubits necessary for the hashing operators.

4 Necessary and Sufficient Condition of Quantum One-Way Permutations

We have a necessary and sufficient condition of cryptographic quantum one-way permutations as follows.

Theorem 4. *The following statements are equivalent.*

1. *There exists a weakly quantum one-way permutation.*
2. *There exists a polynomial-time computable function f satisfying that there exists a polynomial p such that for all sufficiently large n's, any polynomial $r_p(n)$ and any $r_p(n)$-qubit $(1/2^{p(n)}, 1/p(n))$-pseudo identity operator J_n such that $\mathcal{F}_{n,p}(f) = \{(I_n \otimes (R^\dagger J_n R))^\dagger (Q_j(f) \otimes I_{r_p(n)-n})(I_n \otimes (R^\dagger J_n R))\}_{j=0,1,\ldots,\frac{n}{2}-1}$ is not easy, where $R = (R_n \otimes I_{r_p(n)-n})$*[1].

[1] Note that $3n$-qubit ancillae for R_n is omitted for the sake of simple description. The $3n$-qubit ancillae correspond to the second and third registers in the explanation of the hashing operator in Section 3. The unitary operator R_n is performed on the second register (in the statement of Theorem) and the $3n$-qubit ancillae.

Theorem 4 can be proved as the combination of the following two theorems. Exactly speaking, $\mathcal{F}_{n,p}(f)$ in the statement of Theorem 4 is not a family of pseudo reflection operators. However, as we discussed in Section 3, we may regard $\mathcal{F}_{n,p}(f)$ as a family of pseudo reflection operators by feeding $\mathcal{F}_{n,p}(f)$ random inputs for parameters of hashing operators beforehand.

Theorem 5. *Let f be a permutation that can be computed by a polynomial-size quantum circuit. If f is not (weakly) quantum one-way, then for any polynomial p and infinitely many n, there exist a polynomial $r_p(n)$ and an $r_p(n)$-qubit $(1/2^{p(n)}, 1/p(n))$-pseudo identity operator J_n such that the family $\mathcal{F}_{n,p}(f) = \{(I_n \otimes (R^\dagger J_n R))^\dagger (Q_j(f) \otimes I_{r_p(n)-n})(I_n \otimes (R^\dagger J_n R))\}_{j=0,1,\ldots,\frac{n}{2}-1}$ is easy.*

Theorem 6. *Let f be a permutation that can be computed by a polynomial-size quantum circuit. If for any polynomial p and infinitely many n there exist a polynomial $r_p(n)$ and an $r_p(n)$-qubit $(1/2^{p(n)}, 1/p(n))$-pseudo identity operator J_n such that the family $\mathcal{F}_{n,p}(f) = \{\tilde{Q}_j(f)\} = \{(I_n \otimes (R^\dagger J_n R))^\dagger (Q_j(f) \otimes I_{r_p(n)-n})(I_n \otimes (R^\dagger J_n R))\}_{j=0,1,\ldots,\frac{n}{2}-1}$ is easy, then f is not (weakly) quantum one-way.*

Since Theorem 5 can be proved by using a similar argument in the proof of Theorem 2 and the efficient implementability of hashing operators, we omit the proof. (Note that though Theorem 6 can be shown if we use random permutations instead of universal hashing operators, the efficiency of universal hashing operators is essential in the proof of Theorem 5.)

Due to space limitation, we give only a proof sketch of Theorem 6.

Proof Sketch. Suppose that for some fixed polynomial $p(n) \geq 4n+2$, infinitely many n, and some $(1/2^{p(n)}, 1/p(n))$-pseudo identity operator J_n, the family $\mathcal{F}_{p,n}$ of unitary operators is easy. We construct a polynomial-size algorithm av-INV to invert f by using unitary operations in $\mathcal{F}_{p,n}$. Algorithm av-INV is almost similar to Algorithm INV except that the operator Q_j is replaced with $\tilde{Q}_j$. The initial input state to av-INV is also assumed to be $\frac{1}{\sqrt{2^n}}|x\rangle_1 \sum_{y\in\{0,1\}^n} |y\rangle_2|0\rangle_3$, where $|z\rangle_1$ (resp., $|z\rangle_2$ and $|z\rangle_3$) denotes the first n-qubit (resp., the second n-qubit and the last $(r_p(n)-n)$-qubit) register. Algorithm av-INV performs the following steps:

foreach $j = 0$ to $\frac{n}{2} - 1$
(step j.1) Apply O_j to the first and the second registers;
(step j.2) Apply $\tilde{Q}_j$ to all the registers.

To analyze Algorithm av-INV, we use the following functionally equivalent description. (Note that the following procedure may not be efficient though the behavior is equivalent to Algorithm av-INV.)

foreach $j = 0$ to $\frac{n}{2} - 1$
(step A.j.1) Apply O_j to the first and the second registers;
(step A.j.2) Apply $R^\dagger J_n R$ to the second and third registers (and implicit ancillae);
(step A.j.3) Apply Q_j to the first and the second registers;

(step A.j.4) Apply $(R^\dagger J_n R)^\dagger$ to the second and third registers (and implicit ancillae).

Algorithm av-INV uses R before and after J_n. These application of R does not affect the expected performance of J_n in the following sense.

Lemma 1. *Fix parameters for R arbitrarily and let R' be the resulting operator. Suppose that J_n is an $(a(n), b(n))$-pseudo identity operator. Then operator $R'^\dagger J_n R'$ is also an $(a(n), b(n))$-pseudo identity operator.*

Lemma 1 can be easily shown by taking into account that h_{r_1,r_2} is a permutation for any choice of parameters (r_1, r_2) of the hashing operator. The justification of the replacement of the universal hashing operator R by a parameters-fixed universal hashing operator R' has been discussed in Section 3.

From the definition of pseudo identity operators, there exists a set $X_n \subseteq \{0,1\}^n$ with $|X_n| \leq 2^n/p(n)$ such that for any $y \in Y_n = \{0,1\}^n \backslash X_n$, $J_n|y\rangle_2|0\rangle_3 = \alpha_y|y\rangle_2|0\rangle_3 + |\psi_y\rangle_{23}$, where $|\psi_y\rangle_{23} \perp |y\rangle_2|0\rangle_3$ and $|1-\alpha_y| \leq \frac{1}{2^{p(n)}}$.

In Algorithm av-INV, we apply J_n (or, $R'^\dagger J_n R'$) before and after step A.j.3 for each j. The application of J_n makes an error in computation of f^{-1}. We call the vector $J_n|\psi\rangle - |\psi\rangle$ the *error* associated to $|\psi\rangle$. To measure the effect of this error, we use the following lemmas. (Lemma 3 itself was stated in [11].) The proof for Lemma 2 is given in Appendix. We note, in the sequel, the norm over vectors is Euclidean.

Lemma 2. *Assume that $T \subseteq S \subseteq \{0,1\}^n$. Then length $l(S,T)$ of the error associated to the state $|\psi(S,T)\rangle = \frac{1}{\sqrt{|S|}}\left(\sum_{y\in S\backslash T}|y\rangle|0\rangle - \sum_{y\in T}|y\rangle|0\rangle\right)$ satisfies that $l(S,T) \leq 2\sqrt{\frac{|S\cap X_n|}{|S|}} + \gamma(n)$, where $\gamma(n)$ is a negligible function in n.*

Lemma 3. *Let $J_n|\psi(S,T)\rangle = \alpha|\psi(S,T)\rangle + |\psi(S,T)^\perp\rangle$, where $|\psi(S,T)^\perp\rangle$ is a state orthogonal to $|\psi(S,T)\rangle$. Then, $||\psi(S,T)^\perp\rangle| \leq l(S,T)$.*

By using Lemma 2 and Lemma 3, we consider the effect of the additional applications of pseudo identity operators to INV in order to analyze Algorithm av-INV.

For each j, we let $S_j = \{y : f(y)_{(1,2j)} = x_{(1,2j)}\}$ and $T_j = \{y : f(y)_{(1,2j+2)} = x_{(1,2j+2)}\}$. We assume that the state before step A.j.2 is

$$|x\rangle_1|\psi(S_j,T_j)\rangle_{23} = |x\rangle_1 \frac{2^j}{\sqrt{2^n}}\left(\sum_{y\in S_j\backslash T_j}|y\rangle_2 - \sum_{y\in T_j}|y\rangle_2\right)|0\rangle_3.$$

Note that the above state is the same as the one before W.j.2 in Algorithm INV.

In step A.j.2, $R'^\dagger J_n R'$ is applied to the state. Since the hashing operator diffuses X_n and Y_n appeared in the definition of the pseudo identity operator, we have the following from Lemma 2 and the property of the universal hashing.

Lemma 4. *For each j, $\mathbf{E}[l(S_j, T_j)] \leq \frac{2}{\sqrt{p(n)}} + \gamma(n)$, where the expectation is over the random selection of parameters of hashing operators and $\gamma(n)$ is a negligible function in n.*

From Lemma 3 and Lemma 4, we obtain a vector $v = v_1 + v_2$ where $v_1/|v_1|$ is the unit vector corresponding to the state before step W.j.2 in Algorithm INV and v_2 is a vector of expected length at most $2/\sqrt{p(n)}$ orthogonal to v_1. (The negligible term $\gamma(n)$ is omitted for simplicity.) The vector v_2 corresponds to an error that happens when $R'^{\dagger} J_n R'$ is applied before step A.j.3.

Next, we consider the state after step A.j.3. We assume that the state after step A.j.3 is

$$|x\rangle_1 |\psi(S_{j+1}, \varnothing)\rangle_{23} = |x\rangle_1 \frac{2^j}{\sqrt{2^n}} \left(\sum_{y \in S_{j+1}} |y\rangle_2 \right) |0\rangle_3.$$

Note that the above state is the same as the one after step W.j.2 in Algorithm INV. In order to analyze the effect of the application of $(R'^{\dagger} J_n R')^{\dagger}$ after step A.j.3, we need another lemma similar to Lemma 4.

Lemma 5. *For each j, $\mathbf{E}[l(S_{j+1}, \varnothing)] \leq \frac{2}{\sqrt{p(n)}} + \gamma(n)$, where the expectation is over the random selection of parameters of hashing operators and $\gamma(n)$ is a negligible function in n.*

By a similar argument to the above, we obtain a vector $v = v_1 + v_2$ where $v_1/|v_1|$ is the unit vector corresponding to the state after step W.j.2 in Algorithm INV and v_2 is a vector of expected length at most $2/\sqrt{p(n)}$ orthogonal to v_1. (The negligible term $\gamma(n)$ is again omitted for simplicity.) The vector v_2 corresponds to an error that happens when $(R'^{\dagger} J_n R')^{\dagger}$ is applied after step A.j.3.

From the above analysis, we can see that Algorithm av-INV on input x terminates with the final state $v = v_1 + v_2$ where v_1 is parallel to $|x\rangle_1 |f^{-1}(x))_2 |0\rangle_3$ and v_2 is a vector of expected length at most $2 \cdot (n/2) \cdot (2/\sqrt{p(n)}) = 2n/\sqrt{p(n)}$ orthogonal to v_1. Thus, the expected length of v_2 is bounded by $1/q(n)$ for some polynomial q. Therefore, there exists a polynomial-size quantum circuit A and infinitely many n such that $\Pr[A(U_n) = f^{-1}(U_n)] > 1 - \frac{1}{(q(n))^2}$. This implies that f is not weakly quantum one-way. □

5 Conclusion

We have given a necessary and sufficient condition of cryptographic quantum one-way permutations in terms of pseudo identity and reflection operator. In order to prove that our condition is necessary and sufficient, we used a technique of universal hash functions. Thus, the operators corresponding to the universal hashing appear in the condition. However, since it is easy to see that J_n is an easy pseudo identity operator if and only if $R'^{\dagger} J_n R'$ is an easy pseudo identity operator, we may eliminate the universal hashing operators in the condition. If

possible, this implies that the conjecture given by Kashefi, Nishimura and Vedral [11] holds as it is. However, the notion of pseudo identity operators seems not to have a sufficient power to cope with probabilistic properties. Thus, we believe that a randomized version of pseudo identity operators might be indispensable in the condition.

Our necessary and sufficient condition of quantum one-way permutations can be regarded as a universal test for the quantum one-wayness of permutations. As long as the authors know, this is, classical or quantum, the first result on the universality for one-way permutations, though the next bit test is a universal test for pseudorandom generators in the classical computation. We believe that our universal test for quantum one-way permutations may help to find good candidates for them, which are currently not known.

Acknowledgments. We are grateful for valuable comments from anonymous referees. AK would like to acknowledge the financial support of the 21st COE for Research and Education of Fundamental Technologies in Electrical and Electronic Engineering, Kyoto University.

References

1. M. Adcock and R. Cleve, "A quantum Goldreich-Levin theorem with cryptographic applications", In *Proc. 19th Ann. Symp. Theoretical Aspects of Computer Science,* LNCS 2285, Springer, pp.323–334, 2002.
2. C. H. Bennett, E. Bernstein, G. Brassard and U. V. Vazirani, "Strengths and weaknesses of quantum computing", SIAM J. Comp. 26(5), pp.1510–1523, 1997.
3. M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits", SIAM J. Comp. 13(4), pp.850–864, 1984.
4. G. Brassard, P. Høyer, M. Mosca and A. Tapp, "Quantum amplitude amplification and estimation", In, S. J. Lomonaco, Jr. and H. E. Brandt (eds.), *Quantum Computation and Quantum Information,* AMS Contemp. Math. 305, 2002.
5. J. L. Carter and M. N. Wegman, "Universal classes of hash functions", J. Comp. Sys. Sci. 18(2), pp.143–154, 1979.
6. P. Dumais, D. Mayers and L. Salvail, "Perfectly concealing quantum bit commitment from any one-way permutations", In *Proc. EUROCRYPT 2000,* LNCS 1807, Springer, pp.300–315, 2000.
7. O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions", In *Proc. 21st ACM Symp. Theory of Computing,* pp.25–32, 1989.
8. O. Goldreich, *Foundations of Cryptography: Basic Tools,* Cambridge University Press, 2001.
9. L. K. Grover, "A fast quantum mechanical algorithm for database search", In *Proc. 28th ACM Symp. Theory of Computing,* pp.212–219, 1996.
10. J. Håstad, R. Impagliazzo, L. A. Levin and M. Luby, "A pseudorandom generator from any one-way function", SIAM J. Comp. 28(4), pp.1364–1396, 1999.
11. E. Kashefi, H. Nishimura and V. Vedral, "On quantum one-way permutations", Quantum Info. Comp. 2(5), pp.379–398, 2002.
12. A. W. Schrift and A. Shamir, "Universal tests for nonuniform distributions", J. Crypt. 6(3), pp.119–133, 1993.

13. P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", SIAM J. Comp. 26(5), pp.1484–1509, 1997.
14. A. C. Yao, "Theory and applications of trapdoor functions", In *Proc. 23rd IEEE Symp. Foundations of Computer Science,* pp.80–91, 1982.

Appendix — Proof of Lemma 2

First, we restate the property of the length of the error associated to the state $|y\rangle|0\rangle$ which was shown in [11]. The property is that the length is at most $\frac{2}{2^{p(n)/2}}$ if $y \in Y_n$ and at most 2 if $y \in X_n$. Using this property more carefully, we have a more tight bound of $l(S,T)$ as follows:

$$
\begin{aligned}
l(S,T) &= |J_n|\psi(S,T)\rangle - |\psi(S,T)\rangle| \\
&= \frac{1}{\sqrt{|S|}} \left| (J_n - I) \left(\sum_{y \in Y_n \cap (S \setminus T)} |y\rangle|0\rangle - \sum_{y \in Y_n \cap T} |y\rangle|0\rangle \right. \right. \\
&\qquad \left. \left. + \sum_{y \in X_n \cap (S \setminus T)} |y\rangle|0\rangle - \sum_{y \in X_n \cap T} |y\rangle|0\rangle \right) \right| \\
&\leq \frac{1}{\sqrt{|S|}} \left| (J_n - I) \left(\sum_{y \in Y_n \cap (S \setminus T)} |y\rangle|0\rangle - \sum_{y \in Y_n \cap T} |y\rangle|0\rangle \right) \right| \\
&\qquad + \frac{1}{\sqrt{|S|}} \left| (J_n - I) \left(\sum_{y \in X_n \cap (S \setminus T)} |y\rangle|0\rangle - \sum_{y \in X_n \cap T} |y\rangle|0\rangle \right) \right| \\
&\leq \frac{1}{\sqrt{|S|}} \left(\sum_{y \in Y_n \cap (S \setminus T)} |J_n|y\rangle|0\rangle - |y\rangle|0\rangle| + \sum_{y \in Y_n \cap T} |J_n|y\rangle|0\rangle - |y\rangle|0\rangle| \right) \\
&\qquad + \frac{1}{\sqrt{|S|}} \left(\left| J_n \left(\sum_{y \in X_n \cap (S \setminus T)} |y\rangle|0\rangle - \sum_{y \in X_n \cap T} |y\rangle|0\rangle \right) \right| \right. \\
&\qquad \left. + \left| \sum_{y \in X_n \cap (S \setminus T)} |y\rangle|0\rangle - \sum_{y \in X_n \cap T} |y\rangle|0\rangle \right| \right) \\
&\leq \frac{2}{2^{p(n)/2}} \frac{|S \cap Y_n|}{\sqrt{|S|}} + \frac{2}{\sqrt{|S|}} \sqrt{(|X_n \cap (S \setminus T)| + |X_n \cap T|)} \\
&= \frac{2}{2^{p(n)/2}} \frac{|S \cap Y_n|}{\sqrt{|S|}} + 2\sqrt{\frac{|S \cap X_n|}{|S|}}.
\end{aligned}
$$

Let $\gamma(n)$ be the former term in the above inequality. Then

$$\gamma(n) = \frac{2}{2^{p(n)/2}} \frac{|S \cap Y_n|}{\sqrt{|S|}} < \frac{2^{n+1}}{2^{p(n)/2}} < \frac{1}{2^n}$$

and is negligible. (Recall that we take $p(n) \geq 4n + 2$.) □

A Common Algebraic Description for Probabilistic and Quantum Computations*
(Extended Abstract)

Martin Beaudry[1], José M. Fernandez[2], and Markus Holzer[3]**

[1] Département de mathématiques et d'informatique, Université de Sherbrooke, 2500, boul. Université Sherbrooke (Québec), J1K 2R1 Canada
beaudry@dmi.usherb.ca
[2] Département d'I.R.O., Université de Montréal, C.P. 6128, succ. Centre-Ville, Montréal (Québec), H3C3J7 Canada
fernandez@iro.umontreal.ca
[3] Institut für Informatik, Technische Universität München, Boltzmannstraße 3, D-85748 Garching bei München, Germany
holzer@informatik.tu-muenchen.de

Abstract. Through the study of gate arrays we develop a unified framework to deal with probabilistic and quantum computations, where the former is shown to be a natural special case of the latter. On this basis we show how to encode a probabilistic or quantum gate array into a sum-free tensor formula which satisfies the conditions of the partial trace problem, and *vice-versa.* In this way complete problems for the classes pr-BPP (promise BPP) and pr-BQP (promise BQP) are given when changing the semiring from $(\mathbb{Q}^+, +, \cdot)$ to the field $(\mathbb{Q}, +, \cdot)$. Moreover, by variants of the problem under consideration, classes like $\oplus\mathrm{P}$, NP, $\mathrm{C}_=\mathrm{P}$, its complement co-$\mathrm{C}_=\mathrm{P}$, the promise version of Valiant's class UP, its generalization promise SPP, and unique polytime US are captured as problem property and the semiring varies.

1 Introduction

We formalize and confirm the intuition, that basic tensor calculus not only captures natural complexity classes in simple ways as shown in [5] and [3], but also yields a simpler and unified view on classical probabilistic and quantum computation, which gives probabilistic and quantum computations the exact same definition, up to the underlying algebra. Apart from offering a first application of the algebraic approach to quantum computing, our paper thus reasserts the point made in [10], that for the classes BPP and BQP, the jump from classical to

* Supported by the Québec FCAR, the NSERC of Canada, and by the Deutsche Forschungsgemeinschaft.

** Part of the work was done while the author was at Département d I.R.O. of the Université de Montréal.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 851–862, 2004.
© Springer-Verlag Berlin Heidelberg 2004

quantum polynomial-time computation consists in allowing negative matrix entries for the evolution operators, which means that different computations done in parallel may interfere destructively. Based on this unified framework, we define a meaningful computational problem on tensor formulas, called the *partial trace tensor formula problem,* which is fundamental to our studies, and allows us to capture important complexity classes. Our precise characterizations are as follows:

1. We present probabilistic computation as a natural special case of quantum computation using the unified framework on gate arrays, instead of presenting quantum as a more or less artificial extension of probabilistic computation.
2. The partial trace *sum-free* tensor formula problem enables us to capture the significant complexity classes (pr-)P (promise P), NP, pr-BPP (promise BPP), and PP and some of their quantum counterparts pr-EQP (promise EQP), NQP, and pr-BQP (promise BQP), by showing completeness results of the problem under consideration. This suggests that the difference between probabilistic and quantum polytime computers may ultimately lie in the possibility, in the latter case, of having destructive interference between computations occurring in parallel.
3. By giving up sum-freeness, we obtain completeness statements for further complexity classes like $\oplus$P, NP, $C_=$P, its complement co-$C_=$P, Valiant's class pr-UP (promise UP), its generalization pr-SPP (promise SPP), and unique polytime US.

Some of these classes are "semantic" classes, i.e., the underlying machine must obey a property for all inputs, which is not obvious to check, or even undecidable. An example would be UP, since for a nondeterministic machine to define a language in UP, it must have the property that for all inputs either exactly one accepting path exists or none. Therefore, the obtained completeness results are subject to a certain promise. Due to the lack of space the proofs are omitted. A complete presentation of these results will be found in the full paper.

2 Definitions

We use standard notation from computational complexity [11,12]. In particular we recall the inclusion chains: $\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{PP} \supseteq \mathrm{NP}$ and $\mathrm{EQP} \subseteq \mathrm{BQP} \subseteq \mathrm{PP} \supseteq \mathrm{NQP}$. Here P (NP, respectively) denotes the set of problems solvable by deterministic (nondeterministic) Turing machines in polytime, and the probabilistic class PP (BPP, respectively) is the set of all languages accepted by nondeterministic Turing machine in polytime with majority (strict majority, respectively). Moreover, EQP, NQP, and BQP denote the quantum analog of P, NP, and BPP, respectively. In the sequel, whenever we simultaneously deal with probabilism and quantum, we use the notations and vocabulary from the quantum case, in order to make the text easier to read.

A *semiring* [13] is a tuple $(\mathcal{S}, +, \cdot)$ with $\{0, 1\} \subseteq \mathcal{S}$ and two binary operations $+, \cdot : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ (sum and product), such that $(\mathcal{S}, +, 0)$ is a commutative monoid,

$(\mathcal{S}, \cdot, 1)$ is a monoid, multiplication distributes over sum, and $0 \cdot a = a \cdot 0 = 0$ for every a in $\mathcal{S}$. A semiring $\mathcal{S}$ is *commutative* if and only if $a \cdot b = b \cdot a$ for every a and b, it is *finitely generated* if there is a finite set $\mathcal{G} \subseteq \mathcal{S}$ generating all of $\mathcal{S}$ by summation, and is a *ring* if and only if $(\mathcal{S}, +, 0)$ is a group. The special choice of $\mathcal{G}$ has no influence on the complexity of problems we study in this paper. In this paper we consider the following semirings, which are finitely representable, i.e., every element from $\mathcal{S}$ can be encoded and easily manipulated over a finite alphabet: The field of rationals $(\mathbb{Q}, +, \cdot)$ and the commutative semiring of positive rationals $(\mathbb{Q}^+, +, \cdot)$. Moreover, we refer also to the field of complex numbers $(\mathbb{C}, +, \cdot)$.

Let $\mathbb{M}_{\mathcal{S}}^{k,\ell}$ denote the set of all *matrices* over $\mathcal{S}$ of order $k \times \ell$. For a matrix A in $\mathbb{M}_{\mathcal{S}}^{k,\ell}$ let $I(A) = [k] \times [\ell]$, where $[k]$ denotes the set $\{1, 2, \ldots, k\}$. The (i,j)th entry of A is denoted by $a_{i,j}$ or $(A)_{i,j}$. Addition and multiplication of matrices in $\mathbb{M}_{\mathcal{S}}^{k,\ell}$ are defined in the usual way. Additionally we consider the *tensor product* $\otimes : \mathbb{M}_{\mathcal{S}}^{k,\ell} \times \mathbb{M}_{\mathcal{S}}^{m,n} \to \mathbb{M}_{\mathcal{S}}^{km,\ell n}$ of matrices, also known as Kronecker product, outer product, or direct product, which is defined as follows. For $A \in \mathbb{M}_{\mathcal{S}}^{k,\ell}$ and $B \in \mathbb{M}_{\mathcal{S}}^{m,n}$ let $A \otimes B \in \mathbb{M}_{\mathcal{S}}^{km,\ell n}$ be defined as $(A \otimes B)_{i,j} = (A)_{q,r} \cdot (B)_{s,t}$, where $i = k \cdot (q-1) + s$ and $j = \ell \cdot (r-1) + t$. Moreover, let $\oplus : \mathbb{M}_{\mathcal{S}}^{k,k} \times \mathbb{M}_{\mathcal{S}}^{k,k} \to \mathbb{M}_{\mathcal{S}}^{2k,2k}$ be the direct sum of matrices which is defined as $A \oplus B = D_1^2 \otimes A + D_2^2 \otimes B$, for $A, B \in \mathbb{M}_{\mathcal{S}}^{k,k}$, where D_i^n is the order $n \times n$ "dot matrix" having one in position (i,i) and zero elsewhere. In the remainder of this section we define tensor formulas over semirings—see [5] for further information.

Definition 1. *The* tensor formulas *over a semiring $\mathcal{S}$ and their* order *are recursively defined as follows: (1) Every matrix F from $\mathbb{M}_{\mathcal{S}}^{k,\ell}$ with entries from $\mathcal{S}$ is a (*atomic*) tensor formula of order $k \times \ell$. (2) If F and G are tensor formulas of order $k \times \ell$ and $m \times n$, respectively, then (a) $(F + G)$ is a tensor formula of order $k \times \ell$ if $k = m$ and $\ell = n$, (b) $(F \cdot G)$ is a tensor formula of order $k \times n$ if $\ell = m$, and (c) $(F \otimes G)$ is a tensor formula of order $km \times \ell n$. (3) Nothing else is a tensor formula. Let $\mathbb{T}_{\mathcal{S}}$ denote the set of all tensor formulas over $\mathcal{S}$, and define $\mathbb{T}_{\mathcal{S}}^{k,\ell} \subseteq \mathbb{T}_{\mathcal{S}}$ to be the set of all tensor formulas of order $k \times \ell$.*

A tensor formula F is (1) *sum-free* if and only if none of F and its sub-formulas has the form $G + H$, for tensor formulas G and H, (2) *system-like* if and only if all sub-formulas of F evaluate to square matrices or column vectors, and (3) *orthogonal system-like*[1] if and only if is system-like and all sub-formulas of F evaluate to orthogonal square matrices or column vectors whose ℓ_2-norm equals 1.

We only consider semiring elements whose value can be given with a standard encoding over some finite $\mathcal{G}$. Hence, atomic tensor formulas, i.e., matrices, can be string-encoded using list notation such as "[[001][101]]." Non-atomic tensor formulas can be encoded over the alphabet $\Sigma = \{0\} \cup \mathcal{G} \cup \{[,], (,), \cdot, +, \otimes\}$. Strings over Σ which do not encode valid formulas are deemed to represent the trivial tensor formula 0 of order 1×1.

[1] We choose the term "orthogonal system-like" because as we will show, such a formula can be reorganized as a product of an orthogonal matrix with a column vector, i.e., as the specification of an orthogonal system of linear equations.

Observe, that by similar techniques as presented in [5] one can show that the following properties are deterministic polytime computable: Testing whether a string encodes a valid tensor formula and if so (a) to compute its order, (b) to check sum-freeness, (c) orthogonality on sum-free formulas, (d) the system-like property, and (e) whether all atomic sub-formula have orders, which are powers of two. Finally, the evaluation of tensor formulas is defined as follows:

Definition 2. *For each semiring $\mathcal{S}$ and each k and ℓ we define the mapping* $\mathrm{val}_{\mathcal{S}}^{k,\ell} : \mathbb{T}_{\mathcal{S}}^{k,\ell} \to \mathbb{M}_{\mathcal{S}}^{k,\ell}$, *as follows: (1)* $\mathrm{val}_{\mathcal{S}}^{k,\ell}(F) = F$, *if F is atomic, (2)* $\mathrm{val}_{\mathcal{S}}^{k,\ell}(F) = \mathrm{val}_{\mathcal{S}}^{k,\ell}(G) + \mathrm{val}_{\mathcal{S}}^{k,\ell}(H)$, *if $F = (G + H)$, (3)* $\mathrm{val}_{\mathcal{S}}^{k,\ell}(F) = \mathrm{val}_{\mathcal{S}}^{k,m}(G) \cdot \mathrm{val}_{\mathcal{S}}^{m,\ell}(H)$, *if $F = (G \cdot H)$ and $G \in \mathbb{T}_{\mathcal{S}}^{k,m}$, and (4)* $\mathrm{val}_{\mathcal{S}}^{k,\ell}(F) = \mathrm{val}_{\mathcal{S}}^{k/m,\ell/n}(G) \otimes \mathrm{val}_{\mathcal{S}}^{m,n}(H)$, *if $F = (G \otimes H)$ and $H \in \mathbb{T}_{\mathcal{S}}^{m,n}$. That is, we associate with each tensor formula F of order $k \times \ell$ its $k \times \ell$ matrix "value" in the natural way.*

The partial trace evaluation problem is defined as follows:

Definition 3. *Let $\mathcal{S}$ be a semiring. The partial trace evaluation problem means to determine the kth partial trace of* $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ *for a given tensor formula F over $\mathcal{S}$ of order $n \times 1$ and a natural number k, which is a power of two and is written in binary. Here the kth* partial trace *of an $n \times n$ matrix A, is the sum of its first k diagonal elements, counting downwards from the upper left corner.*

3 Background on Gate Arrays and Complexity

We introduce gate arrays in order to handle the two types of computations, i.e., probabilistic and quantum. It is useful to think of gate arrays as natural extensions of classical leveled Boolean circuits. The usual notion of depth and size on Boolean circuits naturally carries over to gate arrays. These consist of gates interconnected without fan-out[2] or feedback, by wires. Each wire represents a path of a single bit in time or space, forward from left to right, and it can be described by a state in a two dimensional space with orthonormal basis $|0\rangle$ and $|1\rangle$. The gates have the same number of inputs and outputs, and a gate of k inputs operates on the set of k-bit vectors mapping each of the 2^k possibilities of input values to a combination of output values, i.e., it can be specified by a square matrix over a certain semiring $\mathcal{S}$, which describes its action on the specified entries and may obey certain properties. Without loss of generality we may assume, that each gate acts on neighbouring wires. This requirement can easily be achieved at the cost of inserting a quadratic number of extra levels of "swap" gates, which interchange the values carried by two adjacent wires. Entries to the gate array are either input bits or non-input bits also called *ancilla* bits. Thus, an n-bit input to a gate array over semiring $\mathcal{S}$ can be seen as a formal sum of the form $|\varphi\rangle = \sum_{w \in \{0,1\}^n} \alpha_w |w\rangle$, where α_w is in $\mathcal{S}$ and $|\varphi\rangle$ may obey some additional

[2] Fan-in and fan-out are electrical engineering terms which refer to the *joining* and *branching* of wires; sometimes logical devices output k wires with the same signal, hence providing broadcasting, and this is known as fan-out k.

properties, and gates act on certain bits $|0\rangle$ and $|1\rangle$ of $|\varphi\rangle$ in the natural way. The vector of bits received as input by a gate array can be regarded as a linear combination of *(pure) states.* Finally, at the end of the gate array the decision whether the input is accepted or rejected is done by a particular observation on the output vector, which is also of the above given form. Next we compare quantum and probabilistic computation. We continue with the former one.

Quantum computation was originally defined by Deutsch [6] in terms of quantum Turing machines: Here the data (qubits) handled by this machine are formally represented as a vector whose complex components give the distribution of amplitudes for the probability that the qubits be in a certain combination of values and each transition of the machine acts as a unitary transformation on this vector. Later it was shown in [15] that polytime quantum Turing machines (and their inputs) can be encoded in deterministic polytime into an equivalent quantum gate array, if one allows a small probability of error. In our general view on gate arrays the properties of wires, gates, input and output vectors, and measurement at the very end of the gate array read as follows: (1) Wires in a quantum gate array carry qubits and they can be described by a state in a two dimensional Hilbert space with basis $|0\rangle$ and $|1\rangle$. Just as classical bit strings can represent the discrete states of arbitrary finite dimensionality, so a length n string of qubits can be used to represent quantum states in any Hilbert space of dimensionality up to 2^n. (2) The action of a k-input gate is a unitary operation of the group $U(2^k)$, i.e., a generalized rotation in a Hilbert space of dimension 2^k. The unitarity (orthogonality) property of the square matrices, which describe the performance of the gates, implies reversibility, i.e., computations where the input and output is uniquely retrievable from each other. In this way, it is always possible to un-compute or reverse the computation. It has been shown [2,7] that a small set of one- and two-qubit gates suffices to build quantum arrays, in that any k-qubit gate can be simulated by a gate array consisting of two-qubit gates, and the number thereof is at most an exponential in k. As two-qubit gates it suffices to take the controlled NOT-gate, which is defined as $x \mapsto x$ and $y \mapsto x \oplus y$, where $\oplus$ is the two-input one-output XOR function. Moreover, the power of quantum gate arrays remain unchanged if gates are restricted to implement unitary operations with entries taken form a small set of rationals [1]. (3) The coefficients α_x in vector $|\varphi\rangle = \sum_{w\in\{0,1\}^n} \alpha_x|x\rangle$ are called *amplitudes* and they satisfy Without loss of generality the input ancilla qubits are prepared to be in state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Assuming an even number of ancillae, we are back with the rationals since $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. Later, we will use a similar trick for probabilistic computations. (4) Finally, there is a measurement done on the array's output, which consists in projecting the output vector onto a subspace, usually defined by setting a chosen subset of the qubits to $|1\rangle$, the accepting subspace. If the qubits are numbered 1 to n, then a k-qubit accepting subset can be chosen to be qubits 1 to k, at the cost of inserting a quadratic number of extra swap gates. Thus, the probability of acceptance on input w equals the 2^kth partial trace of the matrix $|\psi\rangle\langle\psi|$, where the input is mapped to $|\psi\rangle$ by the gate array under consideration.

As quantum classes, also probabilistic complexity classes are usually defined in terms of Turing machines. Here the Turing machine picks one random bit at a time and acts deterministically otherwise[3]. In fact, deterministic computations, or to be more precise Boolean circuits, can be made reversible with little cost in efficiency [4], since there exists a three-bit universal gate for reversible computations, that is, a gate which when applied in succession to different triplets of bits in a gate array, could be used to simulate arbitrary reversible computations. This universal gate is called the Toffoli-gate, and is also known as the double-controlled NOT- or controlled-controlled NOT-gate and its behaviour is $x \mapsto x$, $y \mapsto y$, and $z \mapsto (x \wedge y) \oplus z$, where $\oplus$ is the two-input one-output XOR function. One can easily prove that the Toffoli gate is universal; by setting z to 1 at the input, the Toffoli gate produces the NAND function, which is a two-input one-output universal gate for classical irreversible computation. Thus, from a probabilistic Turing machine an equivalent circuit and in turn a gate array over the positive rationals can be built, in which an appropriate number of random bits are fed alongside the input bits. Whether the input belongs to the language specified by the Turing machine is verified by counting those combinations of random bits, for which the output bit takes value 1, assuming that all random bit combinations have equal length and are equally likely. In this way, the constraints of a gate array read as follows: (1) Wires carry bits $|0\rangle$ and $|1\rangle$. (2) Gates implement deterministic reversible computations, i.e., they carry out permutation operations thus can be described by matrices with 0-1 entries. (3) The coefficients α_x in vector $|\varphi\rangle = \sum_{w\in\{0,1\}^n} \alpha_x |x\rangle$ are called *probabilities* and they satisfy $\sum_{w\in\{0,1\}^n} \alpha_w = 1$. Moreover, input ancilla (probabilistic) bits are prepared to be equally likely, i.e., set to $\frac{1}{2}(|0\rangle + |1\rangle)$. (4) The measurement at the end of the gate array consists in determining the probability that the decision bits take some predefined values, usually set to $|1\rangle$, at the output level. Thus, the probability of acceptance equals the sum of some the coefficients α_w corresponding to the accepting subspace of the output vector $|\psi\rangle = \sum_{w\in\{0,1\}^n} \alpha_w |w\rangle$.

When restricting to rational numbers, the essential difference between probabilistic and quantum computation lies in the way, the probability of acceptance is determined. In most papers, quantum computation is presented as a natural extension of probabilistic computation. This is not convenient for us, therefore we go the other way around and want to explain how to see probabilistic computation as a natural special case of quantum. In this respect, we are already half the way towards this goal. Consider probabilistic gate arrays in more detail. Since all the gates in the array do classical reversible computations they only permute the different vector components without ever combining them, i.e., no interference ever takes place along the array's computation, so that it does not matter in terms of overall outcome, whether the vector entries are probabilities represented as such or as amplitudes. However, using amplitudes enables us to describe the

[3] When considering probabilistic Turing machines as Turing machines in which some transitions are random choices among finitely many alternatives, the below given argument results in gate arrays, where the gates can be described by stochastic matrices, the only ℓ_1-norm preserving linear mapping over the positive rationals.

measurement at the end of the computation as in the quantum case, by determining the partial trace of a matrix. Nevertheless, we face the problem, that the amplitudes compared to the probabilities in the original vector may not be rational anymore. For instance, the amplitudes to $\frac{1}{2}(|0\rangle+|1\rangle)$ are $\frac{1}{\sqrt{2}}$ for both $|0\rangle$ and $|1\rangle$. As argued in the quantum case, when considering an even number of ancilla bits, we can overcome this problem still staying rational. This observation allows us to see probabilistic computation as a natural special case of quantum computation as follows: (1) Wires carry bits $|0\rangle$ and $|1\rangle$ and (2) gates are describable by orthogonal matrices over the positive rationals. It is elementary to verify that these are exactly the permutation matrices. In other words, these gates are still classical reversible gates. (3) Instead of dealing with probabilities we compute with amplitudes, which implies that the vector $|\varphi\rangle = \sum_{w\in\{0,1\}^n} \alpha_x|x\rangle$ preserves ℓ_2-norm, and the even number of ancilla bits are prepared to be equally likely, which means, that they are set to $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. (4) The measurement at the end of the gate array is done as in the case of quantum gate arrays. The above given discussion motivates and satisfies the following definition and theorem.

Definition 4. *Let $\mathcal{S}$ be the set of positive rationals $\mathbb{Q}^+$ or the set of rationals $\mathbb{Q}$. Define R, $A \subseteq \mathcal{S}$ with $R\cap A = \emptyset$ and $R\cup A \subseteq [0,1]$. A logspace (polytime) uniform family of polynomial size gate arrays over $\mathcal{S}$ determines a language L as follows: Assume $1^{|w|} \mapsto C$ with an even number of ancilla bits, vector $|\varphi\rangle$ is built by input w and appropriately set ancilla bits, and $|\varphi\rangle \mapsto |\psi\rangle$ running C on input w. Then $w \in L$ implies $f_{C(w)} \in A$ and $w \notin L$ implies $f_{C(w)} \in R$, where $f_{C(w)}$ denotes the probability that $|\psi\rangle$ is projected onto the accepting subspace, i.e., equals the partial trace of $|\psi\rangle\langle\psi|$ restricted to the accepting subspace. The class $\mathcal{C}_{\mathcal{S}}(R,A)$ consists of all languages $L \subseteq \Sigma^*$ that can be accepted by gate arrays over $\mathcal{S}$ satisfying the above property.*

Theorem 1. *1. For the positive rationals we find:*

a) $\mathcal{C}_{\mathbb{Q}^+}(R,A) = \mathrm{P}$ *if* $R = [0]$ *and* $A=[1]$.

b) $\mathcal{C}_{\mathbb{Q}^+}(R,A) = \mathrm{NP}$ *if* $R = [0]$ *and* $A=(0,1]$.

c) $\mathcal{C}_{\mathbb{Q}^+}(R,A) = \mathrm{PP}$ *if* $R = [0,\frac{1}{2}]$ *and* $A = (\frac{1}{2},1]$.

d) $\mathcal{C}_{\mathbb{Q}^+}(R,A) = \mathrm{BPP}$ *if* $R = [0,\frac{1}{3}]$ *and* $A = [\frac{2}{3},1]$.

2. For the rationals we find:[4]

[4] Observe that the quantum analog to PP is in fact no different than PP itself. We recall the simple argument, which leads to this observation. An alternative characterization of PP reads as follows [8]: A language $L \subseteq \Sigma^*$ belongs to PP if and only if there is a GapP function f whose value on input w is positive, i.e., $f(w) > 0$ if and only if w is in L. Now given a quantum gate array, which checks membership of w in L with unique accepting and rejecting configurations, summing all the positive and negative contributions to the total amplitude for these configurations defines four #P functions. The difference between the probabilities of acceptance and rejection by this gate array is a quadratic polynomial in these four functions, which belongs to GapP by the closure of #P under finite sum and product. Thus, language L is a member of PP.

a) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{EQP}$ *if* $R = [0]$ *and* $A = [1]$.
b) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{NQP}$ *if* $R = [0]$ *and* $A = (0, 1]$.
c) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{PP}$ *if* $R = [0, \frac{1}{2}]$ *and* $A = (\frac{1}{2}, 1]$.
d) $\mathcal{C}_{\mathbb{Q}}(R, A) = \mathrm{BQP}$ *if* $R = [0, \frac{1}{3}]$ *and* $A = [\frac{2}{3}, 1]$. □

4 Gate Arrays to Sum-Free Tensor Formulas and Back

We show how gate arrays and sum-free tensor formulas are related to each other. In the forthcoming we say that a gate array is *reversible* if and only if all gates in the gate array can be described by orthogonal matrices. Thus, both quantum and probabilistic gate arrays are reversible gate arrays. The construction of a sum-free tensor formula from a given gate array is rather straightforward and is done as follows.

Lemma 1. *Let C be a (reversible) gate array operating on n wires, whose gates can be described by (orthogonal) square matrices over a semiring $\mathcal{S}$. Then there is a polytime computable function, which given a suitable encoding of C, computes a (orthogonal) system-like sum-free tensor formula F_C of order $2^n \times 2^n$ such that for each $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, if gate array C maps $|\varphi\rangle = |x_1 \ldots x_n\rangle$ to $|\psi\rangle$, then $|\psi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,2^n}(F_C) \cdot |\varphi\rangle$, and $|\varphi\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_x{}^{\mathsf{T}})$ for some polytime computable sum-free tensor formula d_x. Due to the linearty of the gate array "semantics" arbitrary input vectors are appropriately mapped to output vectors.*

In the formula to gate array part, we must deal with the fact that a sum-free tensor formula may contain matrices of various sizes and vectors at atypical locations. We first show how to transform a system-like sum-free tensor into an equivalent gate array, if the formula obeys some additional easily checkable properties.

Theorem 2. *Let F be a (orthogonal) system-like sum-free tensor formula of order $2^n \times 1$ over semiring $\mathcal{S}$, where the orders of all atomic atomic sub-formulas are powers of two. Then there is a polytime computable function, which given the tensor formula F, computes a (reversible) gate array C_F over $\mathcal{S}$ operating on n wires and an input $|\varphi_F\rangle$ (with $\langle\varphi_F|\varphi_F\rangle = 1$), such that $|\psi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(F)$, if gate array C_F maps $|\varphi_F\rangle$ to vector $|\psi_F\rangle$, and $|\varphi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^n,1}(d_F{}^{\mathsf{T}})$ for some sum-free tensor formula d_F.*

The following lemma, which is very technical and the main ingredient of the transformation to an equivalent gate array, shows how to pad matrices and vectors in order to turn their orders into powers of two.

Lemma 2. *Let F be a (orthogonal) system-like tensor formula F of order $n \times 1$ over semiring $\mathcal{S}$. Then there is a polytime computable function, which given the tensor formula F, computes a (orthogonal) system-like tensor formula G over the same semiring of order $m \times 1$ having only atomic sub-formulas whose orders are power of two, such that $\mathrm{val}_{\mathcal{S}}^{n,1}(F)$ appears in the upper left corner of $\mathrm{val}_{\mathcal{S}}^{m,1}(G)$, i.e., $\mathrm{val}_{\mathcal{S}}^{m,1}(G) = \begin{pmatrix} \mathrm{val}_{\mathcal{S}}^{n,1}(F) \\ (0)_{n-m}{}^{\mathsf{T}} \end{pmatrix}$, where $(0)_k$ denotes the all zero row vector of length k. If F is sum-free, then so is tensor formula G.*

As an immediate consequence of Theorem 2 and Lemma 2 we obtain the following theorem, which is the main result of this subsection.

Theorem 3. *Let F be a (orthogonal) system-like sum-free tensor formula F of order $n \times 1$ over semiring $\mathcal{S}$. Then there is a polytime computable function, which given the tensor formula F, computes a (reversible) gate array C_F over $\mathcal{S}$ operating on m wires, with $n \leq 2^m$, and an input $|\varphi_F\rangle$ (with $\langle\varphi_F|\varphi_F\rangle = 1$), such that $|\psi_F\rangle = \begin{pmatrix} \mathrm{val}_{\mathcal{S}}^{n,1}(F) \\ (0)_{2^m-n}^{\mathsf{T}} \end{pmatrix}$, if gate array C_F maps $|\varphi_F\rangle$ to vector $|\psi_F\rangle$, and $|\varphi_F\rangle = \mathrm{val}_{\mathcal{S}}^{2^m,1}(d_F{}^{\mathsf{T}})$ for some sum-free tensor formula d_F.* □

5 Complexity Results for Sum-Free Tensor Formulas

In this section we prove completeness results on variants of the partial trace problem for orthogonal system-like sum-free tensor formulas. These variants are defined as follows:

Definition 5. *Let $\mathcal{S}$ be a semiring. (1) The* one partial trace problem *over semiring $\mathcal{S}$ is the set of all tensor formulas F of order $n \times 1$ together with a natural number k, which is a power of two, given in binary, for which the kth partial trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ equals 1. (2) The* non-zero partial trace problem *over semiring $\mathcal{S}$ is the set of all tensor formulas F of order $n \times 1$ together with a natural number k, which is a power of two, given in binary, for which the kth partial trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ is non-zero.*

In order to obtain our completeness results we have to deal with promise versions of the above defined problems. Moreover, we also make use of promise complexity classes—see [14] for further details. The promise version of the one partial trace and non-zero partial trace problem restricted to tensor formulas of order $n \times 1$ such that the kth partial trace of the matrix $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ evaluates to either 0 or 1, will be called the 0-1-promise in the forthcoming. Moreover, we refer to the promise classes P and EQP, respectively, as pr-P and pr-EQP, respectively. Then the following first main theorem of this section reads as follows.

Theorem 4. *(1) The* 0-1-*promise version of the one partial trace problem over the positive rationals $\mathbb{Q}^+$ (rationals $\mathbb{Q}$, respectively), restricted to the domain of orthogonal system-like sum-free tensor formulas, is complete for* pr-P (pr-EQP, *respectively) under polytime many-one reductions. (2) The non-zero partial problem over the positive rationals $\mathbb{Q}^+$ (rationals $\mathbb{Q}$, respectively), restricted to the domain of orthogonal system-like sum-free tensor formulas, is complete for* NP (NQP, *respectively) under polytime many-one reductions*

In the remainder of this section we define meaningful problems, which capture PP, pr-BPP, and pr-BQP.

Definition 6. *Let $\mathcal{S}$ be either the commutative semiring of positive rationals $\mathbb{Q}^+$ or the field of rationals $\mathbb{Q}$. The* majority partial trace problem *over semiring $\mathcal{S}$ is*

the set of all orthogonal tensor formulas F of order $n \times 1$ together with a natural number k given in binary, for which the kth partial trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$ is superior to $\frac{1}{2}$.

Since BPP and BQP are defined *via* a semantic condition we need a promise version of the majority partial trace problem, which captures this condition in order to obtain completeness result. Let $\mathcal{S}$ be either $\mathbb{Q}^+$ or $\mathbb{Q}$. Restricting the majority partial trace problem to orthogonal tensor formulas of order $n \times 1$ such that the (partial) trace of $\mathrm{val}_{\mathcal{S}}^{n,n}(F \cdot F^{\mathsf{T}})$, belongs to the interval $[0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$, is called the *strict majority partial trace problem.*

Theorem 5. *(1) The majority partial trace problem over both, the positive rationals $\mathbb{Q}^+$ and rationals $\mathbb{Q}$ in general, restricted to the domain of orthogonal system-like sum-free tensor formulas is complete for* PP *under polytime many-one reductions. (2) The strict majority partial trace problem over the positive rationals $\mathbb{Q}^+$ (rationals $\mathbb{Q}$, respectively), restricted to the domain of orthogonal system-like sum-free tensor formulas is complete for* pr-BPP *(*pr-BQP*, respectively) under polytime many-one reductions.*

It is worth mentioning, that both the one partial trace and non-zero partial trace problem over the Boolean semiring $\mathbb{B}$, restricted to the domain of orthogonal system-like sum-free tensor formulas, is complete for P under polytime many-one reductions. Moreover, from the results presented in [9] one can deduce that the majority partial trace problem over the quaternions $\mathbb{H}$ is PP-complete under polytime many-one reduction, while the strict majority partial trace problem over the quaternions $\mathbb{H}$ becomes pr-BQP-complete.

6 Complexity Results for Tensor Formulas in General

To understand the statement of Lemma 3 below, keep in mind a situation in which it is required to determine the trace of say $A \oplus B \oplus C \oplus D$, where A, B, C, and D are $\ell n \times \ell n$ matrices. Lemma 3 describes a preliminary step which uses tensors to produce a large block matrix having $n \times n$ sub-matrices compatible to A, B, C, and D on its main diagonal, i.e., the new matrix has exactly the same diagonal elements as $A \oplus B \oplus C \oplus D$, but in permuted order. This particular application of Lemma 3 would require the parameter $m = 4$.

Lemma 3. *Let a sequence $\mathbf{A} = (A_i)$, with $1 \leq i \leq m$, of $\ell n \times \ell n$ matrices over a semiring $\mathcal{S}$ be given. Consider the $\ell mn \times \ell mn$ matrix $A = \bigoplus_{i=1}^{m} A_i$. Then there is a polytime Turing machine which computes on input $\mathbf{A}$, a tensor formula $F_{m,\mathbf{A}}$ evaluating to the $\ell mn \times \ell mn$ matrix $B = \bigoplus_{i=1}^{\ell m} B_i$, where $\mathbf{B} = (B_i)$ with $1 \leq i \leq \ell m$, of $n \times n$ matrices satisfying $diag(B) = (P_{\ell}^{\ell mn})^{-1} \cdot diag(A) \cdot P_{\ell}^{\ell mn}$. Here diag(A) denotes the matrix, which consists of the diagonal entries of A and is zero elsewhere.*

As an application we construct complete problems for some complexity classes. For a definition of the complexity classes #P, $\oplus$P, MOD_q-P, GapP, unique solutions US, $\mathrm{C}_{=}$P, UP, and SPP we refer to [12]. Finally, the promise

counterparts of the classes UP and SPP, respectively, are intuitively defined and denoted by pr-UP and pr-SPP, respectively. Now we are ready to state our main theorem, whose proof is heavily based on Lemma 3.

Theorem 6. *(1) The one partial trace problem is complete for* NP, $\oplus\mathrm{P}$, US, *and* $\mathrm{C}_{=}\mathrm{P}$, *respectively, with respect to polytime many-one reductions in case of Booleans* $\mathbb{B}$, *the field* $\mathbb{F}_2$, *the naturals* $\mathbb{N}$, *and the integers* $\mathbb{Z}$, *respectively. (2) The non-zero partial trace problem is complete for* NP, $\oplus\mathrm{P}$, NP, *and the class* co-$\mathrm{C}_{=}\mathrm{P}$, *respectively, with respect to polytime many-one reductions in case of Booleans* $\mathbb{B}$, *the field* $\mathbb{F}_2$, *the naturals* $\mathbb{N}$, *and the integers* $\mathbb{Z}$, *respectively.*

The above given proof can be rewritten in terms of the 0-1-promise version. Meanwhile the complexity of the one partial trace and the non-zero partial trace problem is obtained with a straightforward application of the above given argument. Thus, we state the below given corollary without proof. Observe, that the 0-1-promise is ridiculous in the case of Booleans $\mathbb{B}$ and the field $\mathbb{F}_2$, since by definition the promise condition is met.

Corollary 1. *The* 0-1*-promise versions of the below mentioned problems are complete w.r.t. polytime many-one reductions: Both the one partial trace problem and the non-zero partial trace problem are complete for* NP, $\oplus\mathrm{P}$, pr-UP, *and* pr-SPP, *respectively, in case of Booleans* $\mathbb{B}$, *the field* $\mathbb{F}_2$, *the naturals* $\mathbb{N}$, *and the integers* $\mathbb{Z}$, *respectively.*

7 Conclusions

Through the study of gate arrays, we have developed a common algebraic description for polytime complexity classes, where the choice of the semiring (plus a promise) determines the complexity class. In this way, characterizations of (pr-)P, NP, $\oplus\mathrm{P}$, pr-BPP and their quantum counterparts pr-EQP, NQP, and pr-BQP are obtained. In particular, for the inclusion $\mathrm{BPP} \subseteq \mathrm{BQP}$, the classical model of polytime probabilistic computation turns out to be a special case of polytime quantum computation where interference between computations is ruled out.

The definitions of variants of the partial trace problems on (sum-free) tensor formulas allowed us to obtain complete problems for the above mentioned polytime complexity classes in a very natural way. Moreover, by giving up sum-freeness, classes like $\oplus\mathrm{P}$, NP, $\mathrm{C}_{=}\mathrm{P}$, its complement co-$\mathrm{C}_{=}\mathrm{P} = \mathrm{NQP}$, the promise version of Valiant's class UP, its generalization promise SPP, and unique polytime US, were captured. It would be interesting, to see whether extending our work to other semirings would yield characterizations for further complexity classes.

Acknowledgments. We wish to thank Lance Fortnow and John Watrous for helpful discussion and pointers to useful references. Insightful comments by Gilles Brassard, Michele Mosca, and Pierre McKenzie were also appreciated.

References

1. L. M. Adleman, H. DeMarrais, and M.-D. A. Huang. Quantum computability. *SIAM Journal on Computing,* 26(5):1524–1540, October 1997.
2. A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A,* 52:3457–3467, 1995.
3. M. Beaudry and M. Holzer. The complexity of tensor circuit evaluation. In J. Sgall, A. Pultr, and P. Kolman, editors, *Proceedings of the 26th Conference on Mathematical Foundations of Computer Science,* number 2136 in LNCS, pages 173–185, Mariánské Lázně, Czech Republic, August 2001. Springer.
4. C. H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing,* 18(4):766–776, August 1989.
5. C. Damm, M. Holzer, and P. McKenzie. The complexity of tensor calculus. *Computational Complexity,* 11(1/2):54–89, 2003.
6. D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society London, Series A,* 400:97–117, 1985.
7. D. P. DiVincenzo. Two-bit gates are universal for quantum computation. *Physical Review A,* 51:1015–1022, 1995.
8. S. A. Fenner, L. Fortnow, and S. A. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences,* 48(1): 116–148, February 1994.
9. J. M. Fernandez and W. A. Schneeberger. Quaternionic computing. Technical Report quant-ph/0307017, Los Alamos Preprint Archiv, July 2003.
10. L. Fortnow. One complexity theorist's view of quantum computing. *Theoretical Computer Science,* 292(3):597–610, January 2003.
11. J. Gruska. *Quantum Computing.* Advanced Topics in Computer Science. McGraw Hill, 1999.
12. L. A. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion.* EATCS Monographs in Theoretical Computer Science. Springer, 2002.
13. W. Kuich and A. Salomaa. *Semirings, Automata, Languages,* volume 5 of *EATCS Monographs on Theoretical Computer Science.* Springer, 1986.
14. A. L. Selman. Promise problems complete for complexity classes. *Information and Computation,* 78:87–98, 1988.
15. A. C.-C. Yao. Quantum circuit complexity. In *Proceedings of the 34th Symposium on Foundations of Computer Science,* pages 352–361, Palo Alto, California, November 1993. IEEE Computer Society Press.

Extraction and Implication of Path Constraints*

Yves André, Anne-Cécile Caron, Denis Debarbieux, Yves Roos, and Sophie Tison

Laboratoire d'Informatique Fondamentale de Lille,
U.M.R. C.N.R.S. 8022
Université de Lille 1, 59655 Villeneuve d'Ascq Cedex. France.
{andre,caronc,debarbie,yroos,tison}@lifl.fr

Abstract. We consider semistructured data as rooted edge-labeled directed graphs, and path inclusion constraints on these graphs. In this paper, we give a new decision algorithm for the implication problem of a constraint $p \preceq q$ by a set of constraints $p_i \preceq u_i$ where p, q, and the p_i's are regular path expressions and the u_i's are non-empty paths, improving in this particular case, the more general algorithms of S. Abiteboul and V. Vianu, and Alechina et al. Moreover, in the case of a set of word equalities $u_i \equiv v_i$, we give a more efficient decision algorithm for the implication of a word equality $u \equiv v$, improving the more general algorithm of P. Buneman et al., and we prove that, in this case, the implication problem for non-deterministic models or for (complete) deterministic models are equivalent.

1 Introduction

The development of the World Wide Web has led to the birth of semistructured data models with languages adapted to these models. A lot of works have been done to define such models and to extend database techniques to them. In this paper, we see semistructured data as rooted edge-labeled directed graphs: indeed, we can model HTML pages as a graph (a page is a node, an hyper-link is an edge) or we can model XML documents as graphs. A presentation of this model and an overview of works done in this context can be found in [1].

Let us consider the datum figure 1 which represents a journal. This journal contains articles and each article is written by one or two authors. We can remark that some nodes have several outgoing edges with the same label (for example, the root has several "author" edges). In this case, the graph is said non-deterministic. In the deterministic case, the outgoing edges of a given node must have distinct labels. Note that XML documents are usually non-deterministic.

Path : Query languages proposed for semistructured data and querying the web are based on path expressions (see for example Lorel [2], UnSQL [7]). In particular, a regular path expression or regular query is a regular expression on the alphabet of labels appearing in the data. The result of the regular query q, is

* This research was partially supported by Inria (MOSTRARE team).

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 863–875, 2004.
© Springer-Verlag Berlin Heidelberg 2004

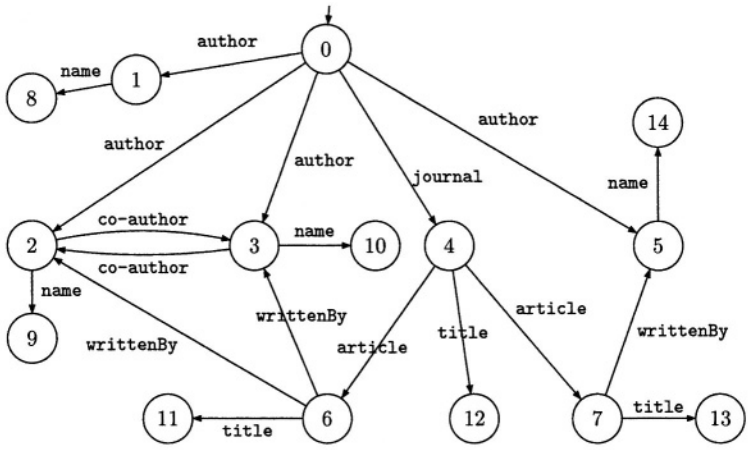


Fig. 1. Example of a semistructured datum

the set of nodes reached from the root by a path labeled by any word u of q. By extension, this word u is called *path.* For example, `author`, `author.name`, `journal.article.title` are paths of the datum D (figure 1). The regular expression `author.co-author*` is a regular query whose result on D is $\{1,2,3,5\}$.

Path inclusion: To optimize path queries, it can be useful to use structural informations about the data. Some of these are called path constraints since they give restrictions on the data paths. Certain kinds of integrity constraints found in object-oriented databases and also common in semistructured databases can be expressed with path constraints. These constraints have been introduced by Abiteboul and Vianu in [3]. See for instance [9], [17] or [5] where different classes of path constraints are analyzed. Here, we study path inclusion constraints. A path inclusion constraint is written $p \preceq q$ where p and q are regular path queries, and means that the set of nodes result of p is included in the set of nodes result of q. Continuing the example, since the result of $\mathbf{author.co\text{-}author}^+$ is $\{2, 3\}$ and the result of `journal.article.writtenBy` is $\{2, 3, 5\}$ the path inclusion $\mathbf{author.co\text{-}author}^+ \preceq \mathbf{journal.article.writtenBy}$ is satisfied. If we denote by $p \equiv q$ the conjunction $p \preceq q \wedge q \preceq p$, the datum D satisfies $\mathbf{author.co\text{-}author} \equiv \mathbf{author.co\text{-}author.co\text{-}author}$. On this example, the constraint is of the form $u \equiv v$ where u and v are paths. We call *word equality* this kind of constraint. Similarly, the constraint $u \preceq v$ is called *word inclusion.*

To take advantage of path inclusions, we must be able to reason about them. In this paper, we study the classical implication problem and the decidability of the boundedness property which is a decision problem close to the implication.

Implication problem: A set of path inclusions $\mathcal{C}$ implies a path inclusion $p \preceq q$ denoted $\mathcal{C} \models p \preceq q$ if every datum model of $\mathcal{C}$ is also a model of $p \preceq q$. Given a

set $\mathcal{C}$ of path inclusions, and two regular queries p, q, the implication problem for $\mathcal{C}$, p, q is to decide whether $\mathcal{C} \models p \preceq q$.

In this paper, we give a decision algorithm for the implication problem of a path inclusion $p \preceq q$ by a set of path inclusions $p_i \preceq u_i$, where p, q, the p_i's are regular path expressions, and the u_i's are non-empty paths. This problem is shown decidable in EXPSPACE in [3]. The authors of [5] give an EXPTIME decision algorithm and prove that this problem is PSPACE-hard. We give here a PSPACE algorithm for this decision problem. In the particular case of deciding if a word equality $u \equiv v$ is implied by a finite set of word equalities $u_i \equiv v_i$ we have an ad-hoc decision algorithm. In [8], the authors give a cubic decision algorithm in the case of the implication for deterministic models (with more general forward constraints). Here, we prove that, in this very particular case of word equalities, implication problem for non-deterministic models is equivalent to the implication problem for (complete and) determistic ones. We build a decision algorithm which complexity is quasi-linear.

Boundedness property:
A regular query p has the *boundedness property (strong boundedness property)* w.r.t a set $\mathcal{C}$ of path inclusions if there exists a regular query f such that $\mathcal{C} \models p \preceq f$ ($\mathcal{C} \models p \equiv f$) and $L(f)$, the language described by f, is finite. On the example, since `author.co-author` $\equiv$ `author.co-author.co-author`, the regular query `author.co-author`$^+$ has the strong boundedness property. Since it is easier to answer to a finite query, we can see the strong boundedness property as a query optimization method. More generally, if a query q has the boundedness property w.r.t $\mathcal{C}$, there exists a finite query f such that $q \preceq f$. So it is possible to approximate q with a finite query f since the answer of f is a superset of the answer of q.

The next section contains formal definitions related to path inclusions. In section 3, we use rewriting to define a decision algorithm for the implication problem of a constraint $p \preceq q$ by a set of constraints $p_i \preceq u_i$, where p, q, the p_i's are regular path expressions, and the u_i's are non-empty paths. In section 4, we give a decision algorithm for the implication problem of a word equality $u \equiv v$ by a set of word equalities $u_i \equiv v_i$. In section 5, we present an algorithm which computes from a finite datum D a set of word inclusions ; this set is finite and implies exactly every word inclusion satisfied by D.

2 Preliminaries

In the sequel we use the following notions which were introduced in [3]. Let A be a fixed finite alphabet of labels.

Definition 1. *A* semistructured datum *is a triple* $D =< N,$ `root`$, T >$ *where* N *is a set of nodes,* `root` $\in N$ *is called the* root *of datum* D *and* $T \subseteq N \times A \times N$ *is the set of* transitions. *If* N *is finite, the datum is said finite. If, for all* n *in* N*, for all* a *in* A*, there is at most (at least) one transition* (n,a,n') *in* T*, then the graph is* deterministic (complete).

Definition 2. *A* path *is a word over the alphabet A. A* regular query p *is a regular expression over A. The* result *of a query* p *over a datum D is the set* $\mathrm{acc}_D(p) = \cup_{u \in L(p)} \mathrm{acc}_D(u)$ *where* $L(p)$ *denotes the regular language described by* p *and* $\mathrm{acc}_D(u)$ *is defined by :*

1. *if* $u = \varepsilon$, *then* $\mathrm{acc}_D(u) = \{\mathtt{root}\}$
2. *if* $u = u'a$ $(a \in A)$, *then* $\mathrm{acc}_D(u) = \{n \in N \mid \exists n' \in \mathrm{acc}_D(u'), (n', a, n) \in T\}$

Definition 3. *A* path inclusion *is an expression of the form* $p \preceq q$ *where* p, q *are regular queries. A* path equality $p \equiv q$ *represents the conjunction* $(p \preceq q) \wedge (q \preceq p)$. *If* u *and* v *are paths,* $u \preceq v$ *is called a* word inclusion, *and* $u \equiv v$ *is called a* word equality. *A datum D* satisfies *(is a model of) a path inclusion* $p \preceq q$, *denoted* $D \models p \preceq q$, *if the set of nodes* $\mathrm{acc}_D(p)$ *is included in* $\mathrm{acc}_D(q)$. *D satisfies a set* $\mathcal{C}$ *of path inclusions, denoted* $D \models \mathcal{C}$, *if D satisfies each path inclusion of* $\mathcal{C}$. *A set* $\mathcal{C}$ *of path inclusions* implies *a path inclusion* $p \preceq q$, *denoted* $\mathcal{C} \models p \preceq q$, *if for each datum D such that* $D \models \mathcal{C}$, $D \models p \preceq q$. *Given a set* $\mathcal{C}$ *of path inclusions, and two regular queries* p, q, *the* implication problem *for* $\mathcal{C}$, p, q *is to decide whether* $\mathcal{C} \models p \preceq q$.

In [3], the authors prove that in the context of the implication problem we can restrict ourselves to finite models, since implication and finite implication of path inclusions are equivalent. It is stated in the following proposition.

Proposition 1. *A set* $\mathcal{C}$ *of path inclusions* implies *a path inclusion* $p \preceq q$, *denoted* $\mathcal{C} \models p \preceq q$, *if for each* ***finite*** *datum D such that* $D \models \mathcal{C}$, $D \models p \preceq q$.

Related to implication, another interesting problem, from a query optimization point of view, is the decidability of the boundedness property.

Definition 4. *A regular query* p *has the* boundedness property *(*strong boundedness property*) w.r.t a set* $\mathcal{C}$ *of path inclusions if there exists a regular query* f *such that* $\mathcal{C} \models p \preceq f$ *(*$\mathcal{C} \models p \equiv f$*) and* $L(f)$ *is finite.*

E.g., let $\mathcal{C}$ be $\{a^2 \preceq a\}$; w.r.t. $\mathcal{C}$, the query $a^\star$ is bounded (by a), whereas the query ba^* is not.

Definition 5. *The* size *of* $p \preceq q$, *denoted by* $|p \preceq q|$, *is the sum* $|p| + |q|$, *where* $|p|$ *is the length of the regular query* p, *and the size of the set of constraints* $\mathcal{C}$, *denoted by* $|\mathcal{C}|$, *is the sum of the sizes of its constraints.*

3 Solving Implication Problems with Rewriting

In order to define a decision algorithm for the implication problem, we introduce in this section a prefix rewrite system built from $\mathcal{C}$ such that u rewrites to v, if and only if $\mathcal{C}$ models $u \preceq v$. These techniques are also used in [3] or in [17].

First, remark that the empty word ε, in semistructured data, is a very particular word : by definition, for any datum D, $\mathrm{acc}_D(\varepsilon)$ is never empty since it

is equal to the singleton which contains the root of D. Hence, path constraints involving the empty word ε are particular too : for instance, if $D \models \varepsilon \preceq v$ for some word v then, for any prefix v' of v, $\mathrm{acc}_D(v')$ is not empty. When the empty word appears in the right-hand side of a path inclusion constraint, the consequences are more surprising: let $\mathcal{C}$ be any set of path constraints such that $\mathcal{C}$ contains $u \preceq \varepsilon$ and $\mathcal{C} \models v \preceq w$ for some words u, v and w. It is easy to see that it follows $\mathcal{C} \models uv \preceq uw$, since for any datum D, $D \models u \preceq \varepsilon$ means $\mathrm{acc}_D(u) = \emptyset$ or $\mathrm{acc}_D(u) = \mathrm{acc}_D(\varepsilon)$. Conversely, when a set $\mathcal{C}$ of word inclusion constraints does not contain any constraint where ε appears in the right-hand side, then all constraints implied by $\mathcal{C}$ can be obtained simply in term of reflexive, transitive or right-congruence closure as it is shown in this section. That is the reason why, from now on, we will suppose that any path constraint $p \preceq q$ satisfies $\varepsilon \notin L(q)$.

3.1 Bounded Path Constraints and Prefix Rewriting

In this subsection, we summarize some results we have shown in [14] and we will only consider the case of a finite set of inclusions of the form $p \preceq u$ where p is a regular path expression and u is a (non-empty) word : we call such path inclusions *bounded path inclusions.* In this case, following and slightly generalizing [3], we associate with a set $\mathcal{C}$ of bounded path inclusions a prefix rewrite system such that u rewrites to v, if and only if $\mathcal{C}$ models $u \preceq v$.

Definition 6. *Let $\mathcal{C} = \{p_1 \preceq u_1, \ldots, p_n \preceq u_n\}$ be a finite set of bounded path inclusions over an alphabet A. We consider the relation on paths defined by $u \xrightarrow[\mathcal{C}]{} v$ if and only if there exists i such that $u \in L(p_i)$ and $v = u_i$. By extension, we denote also $\xrightarrow[\mathcal{C}]{}$ its right congruence closure. Then $\xrightarrow[\mathcal{C}]{}^{\star}$ denotes the reflexive, transitive closure of $\xrightarrow[\mathcal{C}]{}$.*

We can remark that this relation is a prefix rewriting relation as defined in [10] based on an infinite rewrite system. Then using the hypothesis that $\mathcal{C}$ contains only bounded inclusions, we have shown that :

Proposition 2. *Let $\mathcal{C}$ be a set of bounded path inclusions, and q a regular query; the following properties are equivalent:*

- $\mathcal{C} \models u \preceq q$
- *there is some path v in $L(q)$ such that $\mathcal{C} \models u \preceq v$*
- *there is some path v in $L(q)$ such that $u \xrightarrow[\mathcal{C}]{}^{\star} v$*

The proof of that proposition uses a kind of (infinite) exact model $G_{\mathcal{C}}$ of $\mathcal{C}$ which is close to the model defined in [3]. So, using proposition 2, some path constraints properties can be reduced to properties of prefix rewriting relations. We will use extensively this correspondence in the following. The prefix rewriting relation that we use is defined by infinite rewrite systems whereas most known decidability results on prefix rewriting relations deal with finite rewrite systems (e.g. see [10]). However our rewrite systems are particular: they are defined by a

finite set of rules where the left-hand side are regular languages, the right-hand side are words. So one could simulate them by a finite rewrite system, expressing left-hand sides by the finite set of rules of the corresponding automata. An other approach would be to use ground tree transducers defined in [13]. Indeed, as this class of transducers is closed under transitive closure, it is easy to prove that the prefix left-rational rewrite relation can be realized by such a transducer and so that it is a recognizable relation, as defined in [11]. Then, by expressing the required properties in the theory of recognizable relations, we can get decision procedures for implication of constraints or boundedness properties.

We have chosen here a direct approach, even if the spirit of the techniques we use is the same, mainly computation of right congruence closure. This allows us to get simple and efficient constructions and to obtain tighter complexity results.

3.2 Solving Implication Problems

Now, we consider $\mathcal{C}$ a set of bounded path inclusions, p, q regular queries, and we study the implication problem $\mathcal{C} \models p \preceq q$. We obtain different complexity results when p and q are regular languages or simply paths. The algorithms use the computation of a set of ancestors by the rewrite system $\xrightarrow[\mathcal{C}]{\star}$. The main result of this section is the following theorem :

Theorem 1. *Let $\mathcal{C} = \{p_1 \preceq u_1, \ldots, p_n \preceq u_n\}$ be a finite set of bounded path inclusions, and p, q two regular queries. The implication problem $\mathcal{C} \models p \preceq q$ is PSPACE-complete.*

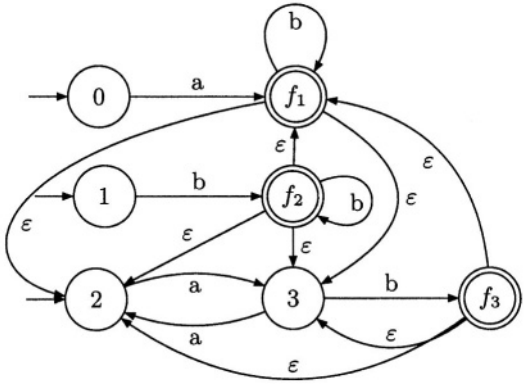
To prove this result, we use the fact that $\mathcal{C} \models p \preceq q$ if and only if $p \subseteq \text{ancestor}_{\mathcal{C}}(q)$, where $\text{ancestor}_{\mathcal{C}}(q)$ denotes the set $\{u \mid \exists\, w_q \in L(q),\ u \xrightarrow[\mathcal{C}]{\star} w_q\}$

In order to compute $\text{ancestor}_{\mathcal{C}}(q)$ for any regular query q, we will use the following construction, introduced in [14]. We build a finite automaton $\mathcal{A}_{\mathcal{C}}$ (with ε-moves) which recognizes the language $R_{\mathcal{C}} = \{v \in A^\star \mid \exists i, v \xrightarrow[\mathcal{C}]{+} u_i\}$. It is already known that $R_{\mathcal{C}}$ is a recognizable language from [6], [11], [10]. We give here a different construction : For each i with $1 \leq i \leq n$, let $\mathcal{M}_i = (A, Q_i, I_i, F_i, \delta_i)$ be an automaton recognizing the language $L(p_i)$. We can assume, without loss of generality, that for different subscripts i and j, the intersection $Q_i \cap Q_j$ is empty. Then we can define $\mathcal{A}_{\mathcal{C}} = (A, Q, I, F, \Delta)$ where $Q = \cup_{i=1}^n Q_i$, $I = \cup_{i=1}^n I_i$, $F = \cup_{i=1}^n F_i$ and $\Delta = \cup_{k \in \mathbb{N}} \Delta_k$ where Δ_k, for $k \in \mathbb{N}$ is defined inductively by :

- $\Delta_0 = \cup_{i=1}^n \delta_i$
- for $k > 0$, $\Delta_k = \Delta_{k-1} \cup \{(q, \varepsilon, q') \mid \exists i \leq n, q \in F_i, q' \in \Delta_{k-1}(I, u_i)\}$

Since Δ is included in $Q \times (A \cup \{\varepsilon\}) \times Q$, it is clear that there exists an integer K such that $\Delta_K = \Delta_{K+1} = \Delta$. Since we have $K \leq |Q|^2$, automaton $\mathcal{A}_{\mathcal{C}}$ can be built in polynomial time in $|\mathcal{C}|$.

Example 1. Let $\mathcal{C} = \{ab^\star \preceq ba, b^+ \preceq a, a(aa)^\star b \preceq a\}$. Automaton $\mathcal{A}_{\mathcal{C}}$ is the following :



It is proven in [14] that $\mathcal{A}_\mathcal{C}$ recognizes $R_\mathcal{C}$, and it is clear that, from automaton $\mathcal{A}_\mathcal{C}$, we easily obtain, for any word u_i, an automaton which recognizes $\text{ancestor}_\mathcal{C}(u_i)$ in PTIME in the size of $\mathcal{C}$. Now, in order to answer to the question $p \subseteq \text{ancestor}_\mathcal{C}(q)$, we have to compute the set of ancestors of q. Let us consider $\$_q$ a new letter (i.e. $\$_q \notin A$). We define a new finite set of bounded path inclusion $\mathcal{C}_q = \mathcal{C} \cup \{q \preceq \$_q\}$, and we can prove that a word u is in $\text{ancestor}_\mathcal{C}(q)$ if and only if u is in $\text{ancestor}_{\mathcal{C}_q}(\$_q) \cap A^*$. It follows that an automaton $\mathcal{A}^q_\mathcal{C}$ for $\text{ancestor}_\mathcal{C}(q)$ can be built in polynomial time in $|q| + |\mathcal{C}|$. In [4] the authors give a decision algorithm for the inclusion of two regular languages $\mathcal{L}_1$ and $\mathcal{L}_2$, given by two automata $\mathcal{A}_1$ and $\mathcal{A}_2$. Using this result, we can enounce :

Lemma 1. *For any set $\mathcal{C} = \{p_1 \preceq u_1, \ldots, p_n \preceq u_n\}$ of bounded path inclusions, and for any regular expressions p and q, the implication problem $\mathcal{C} \models p \preceq q$ is PSPACE.*

We are now able to end the proof of theorem 1, which is a consequence of the following lemma which states that, even when the regular expression q is reduced to a word u, the implication problem $\mathcal{C} \models p \preceq u$ is PSPACE-complete.

Lemma 2. *For any set $\mathcal{C} = \{p_1 \preceq u_1, \ldots, p_n \preceq u_n\}$ of bounded path inclusions, for any regular expressions p and for any word u, the implication problem $\mathcal{C} \models p \preceq u$ is PSPACE-complete.*

Nevertheless, for the implication problem of a constraint $u \preceq q$, we get a polynomial algorithm, since we only check whether $u \in \text{ancestor}_\mathcal{C}(q)$:

Proposition 3. *Let $\mathcal{C} = \{p_1 \preceq u_1, \ldots, p_n \preceq u_n\}$ a set of bounded path inclusions, u a word and q a regular query. We can decide the implication problem $\mathcal{C} \models u \preceq q$ in PTIME.*

4 Word Equality Constraints

In this section, we consider the case of a set of word equality constraints of the form $u \equiv v$ where u and v are paths. In this case, as in [3], we associate with a set C of word equality constraints a graph $G_\mathcal{C}$ such that for any paths u and v, $\mathcal{C} \models u \equiv v$ if and only if $G_\mathcal{C} \models u \equiv v$.

Definition 7. *Let C be a set of word equality constraints over an alphabet A. Clearly, C is a symmetric binary relation over A*. We denote by* $\equiv_C$ *the smallest equivalence relation, closed by right congruence, which contains C, and, for any path* $u \in A^\star$, *we denote by* $[u]_C$ *the equivalent class of the path u for* $\equiv_C$.

Let $G_C =< N, r, T >$ be the labeled rooted directed graph where $N = \{[u]_C \mid u \in A^\star\}$, $r = [\varepsilon]_C$ and $T = \{([u]_C, x, [ux]_C) \mid u \in A^\star, x \in A\}$. Clearly, G_C is deterministic and complete, and for any path $u \in A^\star$, we have $acc_{G_C}(u) = \{[u]_C\}$. It follows :

Proposition 4. *For any set C of word equality constraints over an alphabet A, the following properties are equivalent :*

1. $\mathcal{C} \models u \equiv v$
2. $G_C \models u \equiv v$
3. $u \equiv_C v$

From definition of G_C and from equivalence between 1 and 3 of proposition 4, we obtain that G_C is the unique (complete) deterministic exact model of $\mathcal{C}$. Moreover, since G_C is a complete deterministic graph, we have the following result :

Proposition 5. *For any set C of word equality constraints over an alphabet A, the following properties are equivalent :*

1. $\mathcal{C} \models u \equiv v$.
2. $\mathcal{C} \models u \equiv v$ *on the family of deterministic data.*
3. $\mathcal{C} \models u \equiv v$ *on the family of complete deterministic data.*

Generally, our model G_C is an infinite graph. Nevertheless, when C is a finite set of word equality constraints, it is possible to build a finite deterministic graph in order to decide implication of word equality constraints. A quite similar construction has been introduced by Buneman et al. in [8] :

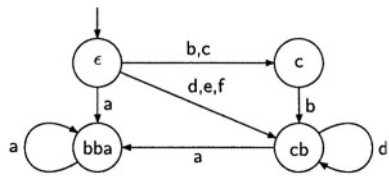


Fig. 2. graph G_C^f

Let $\mathcal{C}$ be a finite set of word equality constraints over A. Let us denote by W the set of all prefixes of $\{w \in A^\star \mid \exists w' \in A^\star, (w \equiv w') \in \mathcal{C}\}$. For any path in W, let us denote by $[w]$ the equivalence class of w for the restriction of $\equiv_C$ over W. Let us consider the finite deterministic graph $G_C^f =< N', r, T' >$ where $N' = \{[w] \mid w \in W\}$, $r = [\varepsilon]$ and $T' = \{([w], x, [wx]) \mid w \in W, wx \in W, x \in A\}$.

Then we can use G_C^f to decide if a finite set $\mathcal{C}$ of word equality constraints implies a word equality constraint. First, it is clear that, by construction, $G_C^f \models \mathcal{C}$ and, for any $w \in W$, $\mathrm{acc}_{G_C^f}(w) = \{[w]\}$. Then, for any path u in A^*, $\mathrm{acc}_{G_C^f}(u) = \{[w]\}$ if and only if $u \equiv_C w$. Let us consider now the application f_C, defined from the set of regular expressions over A to $2^{N' \times A^\star}$, where N' is the set of nodes of G_C^f, by : $\forall p, f_C(p) = \{(\mathrm{acc}_{G_C^f}(u_1), u_2) \mid u_1 u_2 \in L(p) \wedge u_1$ is the least prefix of $u_1 u_2$ such that $\mathrm{acc}_{G_C^f}(u_1) \neq \emptyset\}$.

Example 2. Let $A = \{a, b, c, d, e, f\}$ and $\mathcal{C} = \{a \equiv bba, b \equiv c, cb \equiv dd, d \equiv e, fa \equiv aa, ed \equiv f, e \equiv f, aa \equiv bba\}$. Figure 2 gives the graph $G_{\mathcal{C}}^{f}$ for this set of constraints. On this example, $f_{\mathcal{C}}(a^3) = ([bba], \epsilon)$, $f_{\mathcal{C}}(a^3c) = ([bba], c)$.

Proposition 6. *For any regular expressions p and q, $\mathcal{C} \models p \equiv q$ iff $f_{\mathcal{C}}(p) = f_{\mathcal{C}}(q)$ and p has the strong boundedness property w.r.t. $\mathcal{C}$ iff $f_{\mathcal{C}}(p)$ is finite.*

We will now give the complexity of the decision algorithms which correspond to proposition 6. We will first give an algorithm which computes the graph $G_{\mathcal{C}}^{f}$ with a lower complexity than the algorithm given in [8].

This algorithm uses the union-find principle to compute the set of equivalent classes, in a way similar to Shostak algorithm [18]. The union-find principle manages a set of classes (see [12] for more details). We first represent W by a prefix tree and consider one class by path of W. We compute the equivalent classes : if $(u \equiv v) \in \mathcal{C}$ then compute the union between the class of u and the class of v, and compute recursively the union between the class of ux and the class of vx, for each label x.

With an appropriate data structure (see example 2) we can obtain an $O(|A|)$ algorithm. We prove that our algorithm computes $\{[w], w \in W\}$ and then it is easy to compute $G_{\mathcal{C}}^{f}$. As result of [4] and [19], our algorithm which computes $G_{\mathcal{C}}^{f}$ is better than $O(|\mathcal{C}| \times lg^*(|\mathcal{C}|))$, so is in quasi linear time in $|\mathcal{C}|$.

Example 3. Since $(b \equiv c) \in \mathcal{C}$, b and c are in the same class. Then bb and cb are in the same class. Since $bba \in W$ and $cba \notin W$, we add an edge from the class of cb to the class of bba labeled by a. Finally, after merging the equivalent nodes, we obtain the graph $G_{\mathcal{C}}^{f}$ shown in figure 2.

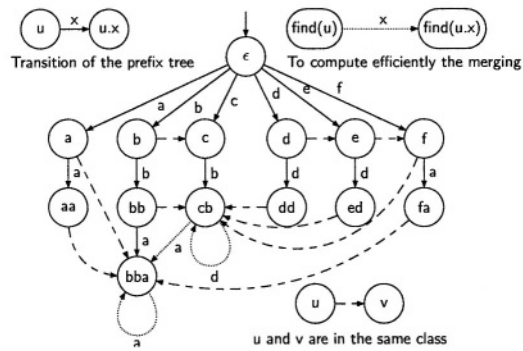


Fig. 3. Steps of the union-find algorithm

Clearly, for any path u, if $G_{\mathcal{C}}^{f}$ is given, the computation of $f_{\mathcal{C}}(u)$ has a linear complexity in the length of u. If we compute $f_{\mathcal{C}}(p)$ for some regular path expression p, we obtain :

Lemma 3. *For any regular path expression p and given the graph $G_{\mathcal{C}}^{f}$, the test of finiteness of $f_{\mathcal{C}}(p)$ can be done in PTIME in the sum of the size of $G_{\mathcal{C}}^{f}$ and the size of p. For any regular path expression p and q a comparison between $f_{\mathcal{C}}(p)$ and $f_{\mathcal{C}}(q)$ for some regular path expressions p and q can be done in PSPACE in the sum of the size of $G_{\mathcal{C}}^{f}$ and the size of the two regular expressions.*

Remark 1. For the comparison of $f_C(p)$ and $f_C(q)$ for some regular path expressions p and q, we cannot obtain a better complexity, since if we consider an empty set C of word equality constraints, we have $\mathcal{C} \models p \equiv q$ if and only if the language described by p is equal to the language described by q, and it is known from [15] that this problem is PSPACE-complete in the sum of the size of the two regular expressions p and q. It follows that the problem to know whether, given a finite set C of word equality constraints, we have $\mathcal{C} \models p \equiv q$ for some regular expressions p and q is PSPACE-complete.

Then, summarizing the complexity results of this section, we obtain :

Proposition 7. *For any finite set of word equality constraints $\mathcal{C}$, it is decidable to know whether $\mathcal{C} \models u \equiv v$ for some paths u and v in quasi-linear time in the sum of $|\mathcal{C}|$ and the size of the constraint $u \equiv v$; the problem to know whether $\mathcal{C} \models p \equiv q$ for some regular path expressions p and q is PSPACE-complete, in the sum of $|\mathcal{C}|$ and the size of the constraint $p \equiv q$; it is decidable to know whether some regular path expression p has the strong boundedness property w.r.t. $\mathcal{C}$ in PTIME in the sum $|\mathcal{C}| + |p|$.*

We can deduce from proposition 5 :

Corollary 1. *For any finite set C of word equality constraints over an alphabet A, and for any regular path expressions p and q, $\mathcal{C} \models p \equiv q$ if and only if $\mathcal{C} \models p \equiv q$ on the family of finite (complete) deterministic data.*

It follows from the proposition 7 that it is possible to check if a regular query q has the strong boundedness property w.r.t $\mathcal{C}$. But we are also able to compute f such that $L(f)$ is finite and $\mathcal{C} \models q \equiv f$.

Proposition 8. *Let $\mathcal{C}$ be a non-empty finite set of word equalities over an alphabet A. We compute, in quasi linear time, a transducer τ_C such that, for any regular query p over A, $\mathcal{C} \models p \equiv \tau_C(L(p))$ and $\tau_C(L(p))$ is finite iff p has the strong boundedness property w.r.t. $\mathcal{C}$*

5 Extraction of a Finite Set of Constraints

Generally, a datum satisfies an infinite number of different path inclusion constraints and, all the more, an infinite number of word inclusion constraints. The goal of this section is to show that one can finitely generate the set of all word inclusion constraints that are satisfied by a given finite datum. More precisely, from a finite datum $D =< N, \mathtt{root}, T >$, we want to define a finite set of word inclusion constraints $\mathcal{C}_f(D)$ such that for any words u and v, $\mathcal{C}_f(D) \models u \preceq v$ if and only if $D \models u \preceq v$.

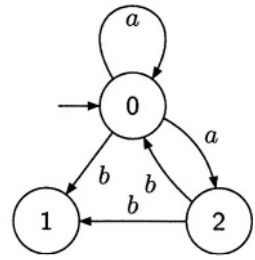
First, let us denote by $S(D)$ the set of all results of word queries over D, that is $S(D) = \{\mathrm{acc}_D(u) \mid u \in A^*\}$ and, for any member s of $S(D)$, let us denote by $\mathrm{lex}(s)$ the smallest word in the lexicographic order of the set $\{u \in A^* \mid \mathrm{acc}_D(u) = s\}$. By definition, for any datum D, we always have $\mathrm{lex}(\{\mathtt{root}\}) = \varepsilon$. Observe that, if we consider a finite datum D as a finite automaton, the set $S(D)$

corresponds with the set of states of the deterministic automaton equivalent to D that is obtained by the well known subset construction.

We are now able to define the following set of word inclusion constraints : $\mathcal{C}_f(D) = \mathcal{C}_{\preceq}(D) \cup \mathcal{C}_{\equiv}(D)$ where $\mathcal{C}_{\preceq}(D) = \{\text{lex}(s) \preceq \text{lex}(s') \mid s, s' \in S(D), s \subsetneq s'\}$ and $\mathcal{C}_{\equiv}(D) = \{\text{lex}(s)x \equiv \text{lex}(s') \mid x \in A, s' = \{n' \in N \mid \exists n \in s, (n, x, n') \in T\}, \text{lex}(s)x \neq \text{lex}(s')\}$.

Example 4. Let us consider the following semistructured datum D with root 0 :

$S(D) = \{\{0\}, \{0,2\}, \{0,1\}, \{1\}, \emptyset\}$,
$\text{lex}(\{0\}) = \varepsilon$, $\text{lex}(\{0,2\}) = a$, $\text{lex}(\{0,1\}) = ab$,
$\text{lex}(\{1\}) = b$, $\text{lex}(\emptyset) = ba$ and
$\mathcal{C}_f(D) = \{aba \equiv a, a \equiv aa, abb \equiv b, bb \equiv ba, \varepsilon \preceq a, \varepsilon \preceq ab, b \preceq ab, ba \preceq \varepsilon, ba \preceq a, ba \preceq ab, ba \preceq b\}$



Clearly, for any finite datum D, $\mathcal{C}_f(D)$ is finite, since the cardinality of $\mathcal{C}_f(D)$ is bounded by 2^{2n}, where n is the number of nodes in datum D. We obtain the following proposition :

Proposition 9. *For any datum D, for any words u and v, $D \models u \preceq v$ if and only if $\mathcal{C}_f(D) \models u \preceq v$. Moreover, if D is finite, then $\mathcal{C}_f(D)$ is finite.*

Observe that the set $\mathcal{C}_f(D)$ may contain word constraints with the empty word as right-hand side, as it is the case in example 4. Nevertheless, it is possible to use rewriting techniques, as presented in section 3, thanks to the following proposition :

Proposition 10. *For any finite datum D over an alphabet A, there exists a finite set of word constraints, $F_D \subseteq A^+ \times A^+$ which satisfies : for any paths u and v in A^+, $D \models u \preceq v$ if and only if $F_D \models u \preceq v$.*

6 Conclusion

The different results concerning the implication problem and the boundedness property are summarized bellow :

bounded path inclusions	new results	already known	word equality constraints	new results	already known
$\mathcal{C} \models p \preceq q$	PSPACE (lemma 1)	EXPSPACE [3] EXPTIME, PSPACE hard [5]	$\mathcal{C} \models p \equiv q$	PSPACE -comp. (prop. 7)	
$\mathcal{C} \models p \preceq u$	PSPACE-comp. (lemma 2)		$\mathcal{C} \models u \equiv v$ (prop. 7)	quasi linear	cubic time [8]
$\mathcal{C} \models u \preceq q$	PTIME (prop. 3)		boundedness property	PTIME (prop. 7)	

When the set of path inclusions $\mathcal{C}$ contains unbounded path inclusions, the problem of deciding whether a regular query p has the boundedness property w.r.t. $\mathcal{C}$ is still open.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web.* Morgan Kaufmann Publishers, 2000.
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The lorel query language for semistructured data. *Journal of Digital Libraries,* 1(1):68–88, 1997.
3. S. Abiteboul and V. Vianu. Regular path queries with constraints. In *Proc. of ACM Symposium on Principles of Database Systems,* 1997.
4. A. Aho, J. Hopcroft, and J. Ullman. *The design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.
5. N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation,* to appear, 2004.
6. J. Richard Büchi and W.H. Hosken. Canonical systems which produce periodic sets. *Mathematical Systems Theory,* 4(1), 1970.
7. P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *SIGMOD,* pages 505–516, Montreal, 1996.
8. P. Buneman, W. Fan, and S. Weinstein. Query optimization for semistructured data using path constraints in a deterministic data model. In *Lecture Notes in Computer Science 1949,* pages 208–223. 7th International Workshop on Database Programming Languages, 1999.
9. P. Buneman, W. Fan, and S. Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences,* 61(2), 2000.
10. D. Caucal. On the regular structure of prefix rewritings. pages 87 – 102, Copenhagen, Denmark, May 1990.
11. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.
12. T. Cormen, C. Leiserson, and R.Rivest. *Introduction to Algorithms.* MIT Press, 1990.
13. M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. pages 242–248. LICS 90, 1990.
14. D. Debarbieux, Y. Roos, S. Tison, Y. Andre, and A.C. Caron. Path rewriting in semistructured data. In *proceedings of words '03: 4th International Conference on Combinatorics on Words,* pages 358–369, Turkü, Finland, 2003. TUCS General Publication.
15. M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the Theory of NP-completeness.* Freeman, 1978.
16. V.M. Gluskov. the abstract theory of automata. In *Russian mathematical survey,* volume 16, pages 1–53, 1961.
17. G. Grahne and A. Thomo. Query containment and rewriting using views for regular path queries under constraints. In *proceedings of PODS'03,* pages 111–122. Symposium on Principles of Database Systems, ACM, 2003.
18. R.E. Shostak. An algorithm for reasoning about equality. *Commun. ACM,* 21(7):583–585, 1978.

19. R. Tarjan. Efficiency of a good but non linear set union algorithm. In *Journal of the ACM,* volume 22, n°2, pages 215 – 225, 1975.

Schema Evolution for XML: A Consistency-Preserving Approach

Béatrice Bouchou[1], Denio Duarte[1*], Mírian Halfeld Ferrari Alves[1], Dominique Laurent[2], and Martin A. Musicante[3**]

[1] Université François Rabelais - LI/Antenne de Blois, France
{bouchou, mirian}@univ-tours.fr, denio.duarte@etu.univ-tours.fr
[2] Université de Cergy-Pontoise - LICP, France
dominique.laurent@dept-info.u-cergy.fr
[3] Universidade Federal do Paraná - Departamento de Informática, Brazil
mam@inf.ufpr.br

Abstract. This paper deals with updates of XML documents that satisfy a given schema, *e.g.,* a DTD. In this context, when a given update violates the schema, it might be the case that this update is accepted, thus implying to change the schema. Our method is intended to be used by a data administrator who is an expert in the domain of application of the database, but who is not required to be a computer science expert. Our approach consists in proposing different schema options that are derived from the original one. The method is consistency-preserving: documents valid with respect to the original schema remain valid. The schema evolution is implemented by an algorithm (called GREC) that performs changes on the graph of a finite state automaton and that generates regular expressions for the modified graphs. Each regular expression proposed by GREC is a choice of schema given to the administrator.

1 Introduction

We consider an XML-based data-exchange environment in which exist both ordinary users and administrators. We are interested in updates to *valid* XML documents, *i.e.,* those that satisfy some schema constraints. When a valid XML document is updated, we have to verify that the new document still conforms to the imposed constraints. *Invalid updates, i.e.,* updates resulting in invalid XML documents, can be treated in different ways, according to the kind of user performing them. Invalid updates performed by ordinary users are rejected, whereas invalid updates performed by administrators can be accepted, thus provoking changes on the schema. We propose a method to enforce the validity of an update by means of changing the schema. Our approach aims to deal with the increasing demand for tools specially designed for administrators not belonging to the computer science community, but capable of making decisions on the

* Supported by CAPES (Brazil) BEX0353/01-9.
** Part of this work was done while the author was visiting Université François Rabelais. Supported by CAPES (Brazil) BEX1851/02-0.

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 876–888, 2004.
© Springer-Verlag Berlin Heidelberg 2004

evolution of an application [10]. This kind of user needs a system that assures a consistent evolution of the schema in a incremental and interactive way. The main features of our method are: (*a*) If an update violates schema constraints then corrective actions are taken to restore validity. This is done by computing all relevant schema changes. Each option is obtained from the characteristics of the schema and documents being updated. (*b*) Valid documents w.r.t. the original schema remain valid w.r.t. the new one. (*c*) Different choices of schema are given to the administrator. The administrator can decide which schema is to be adopted, based on their knowledge about the semantics of the documents.

To our knowledge, our approach adopts a new strategy to deal with schema evolution. Research papers in a similar domain (such as [8,12,13]) propose to change XML documents but not the schema, in case of invalid updates. Notice that our aim is much less ambitious than automatic learning of automata: we propose GREC as a simple and directly usable solution to an interesting problem. Nevertheless, as much work has been done in the area of inference of regular grammars from examples [2,9], we are considering the comparison between our approach and these ones.

An XML document is seen as an unranked labeled tree t having different kinds of nodes *(data, elements* and *attributes).* We assume a schema (defining some element and attribute constraints) specified by an unranked bottom-up tree automaton $\mathcal{A}$ [4]. Checking if an XML document respects the constraints established by the schema is equivalent to run $\mathcal{A}$ with input t. Updates are seen as changes to be performed on XML tree representations and invalid updates are those that produce XML trees which cannot be recognized by $\mathcal{A}$. We focus on (changing the schema after invalid) insertions, since deletions are easy to treat.

An insert operation consists in the insertion of a sub-tree t' into a given position p of t. Before accepting an insertion, we have to test if the new tree respects the constraints established by the associated tree automaton $\mathcal{A}$. These tests are incremental [5]. If the tests fail, changes on the schema are proposed to the administrator. Changing the schema means changing $\mathcal{A}$. The following example illustrates (*i*) how to validate an XML document using $\mathcal{A}$ and (*ii*) how an insertion requested by an administrator can lead to changes to $\mathcal{A}$.

Example 1. Fig. 1(a) shows the labeled tree t, representing part of an XML document. Each node has a label *(e.g., Author)* and a position (like 00, for an *Author* node and ϵ for the root). Let $\mathcal{A}$ be a tree automaton representing a schema. The execution of $\mathcal{A}$ with input t is represented by the labeled tree r (Fig. 1(b)). To illustrate how we obtain r, suppose that $\mathcal{A}$ contains the transition rule

$$Production, < \emptyset, \emptyset >, q_{Sub}\ (q_{Year}\ {q_{JPaper}}^{+})^{*} \rightarrow q_{Production} \qquad (1)$$

The intended semantics of the regular expression $E = q_{Sub}\ (q_{Year}\ {q_{JPaper}}^{+})^{*}$ is that the production of a given author is stated by the area (or subject) of their research, followed by a list of journal papers, presented by year.

Rule (1) states that a position p, labeled *Production* in t, can be associated with the state $q_{Production}$ in r if the constraints established by Rule (1) are

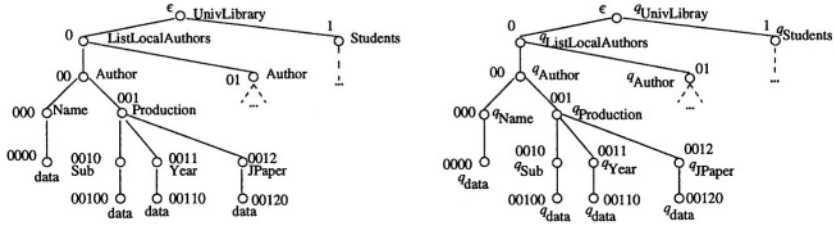


Fig. 1. (a) Labeled tree t. (b) Labeled tree r

respected[1]. As position 001 in t respects these constraints then node 001 in r is assigned to the corresponding state.

The tree automaton executes bottom-up by considering each position and the transition rule that applies to it. A tree automaton accepts a document tree if and only if the state associated to the root is *final*. In this case, we say that t is valid.

We consider now an insertion of a tree t_a at position 0013 (*i.e.,* on the right of position 0012) of the valid tree t. We suppose that the execution of $\mathcal{A}$ over t_a results in a labeled tree whose root is associated to $q_{ConfPaper}$.

In our case, as the update concerns position 0013, we should check if the subtree rooted at position 001 (0013's father) still respects the validity conditions. In other words, we should verify whether the state associated to position 001 after the update is still $q_{Production}$. This is done by analyzing the behavior of Rule (1). As the transition rule defines the possible children of a node using regular expressions we should check if the word $q_{Sub}q_{Year}q_{JPaper}q_{ConfPaper}$, corresponding to the concatenation of the states associated to children of position 001 after the update, matches the regular expression E. In our case, this does not happen and thus we have an invalid update.

As our user is an administrator, the requested update will be taken as a request to change the schema. To understand this request we consider the new word $q_{Sub}q_{Year}q_{JPaper}q_{ConfPaper}$ and the original regular expression $E = q_{Sub}\ (q_{Year}\ q_{JPaper}{}^{+})^{*}$ appearing in (1). The goal is to replace $\mathcal{A}$ by a new tree automaton $\mathcal{A}'$ having the following characteristics: (*i*) every XML document valid with respect to $\mathcal{A}$ is also valid with respect to $\mathcal{A}'$ and (*ii*) $\mathcal{A}'$ differs from $\mathcal{A}$ only in the regular expression affected by the update. The options are:

1. $(q_{Sub}(q_{Year}\ q_{JPaper}{}^{+})^{*}q_{ConfPaper}?)$ and $(q_{Sub}(q_{Year}\ q_{JPaper}{}^{+})^{*}(q_{ConfPaper})^{*})$: Choices allowing the insertion of one or several conference papers to a given domain (not organized by year).
2. $(q_{Sub}(q_{Year}\ q_{JPaper}{}^{+}q_{ConfPaper}?)^{*})$ and $(q_{Sub}(q_{Year}\ q_{JPaper}{}^{+}q_{ConfPaper}{}^{*})^{*})$: These options allow the insertion of one or several conference papers to a given domain per year.

[1] Here, the constraints are: p should have no attribute children (due to the tuple $<\emptyset,\emptyset>$) and the word formed by the concatenation of the states associated to the element children of p should match the regular expression $q_{Sub}\ (q_{Year}\ q_{JPaper}{}^{+})^{*}$.

3. $(q_{Sub}(q_{Year}(q_{JPaper}\ q_{ConfPaper}?)^+)^*)$ and $(q_{Sub}(q_{Year}(q_{JPaper}\ q_{ConfPaper}^*)^+)^*)$: Choices allowing the insertion of several conference papers to a given domain every year. In the first case, each conference paper (if it exists) should be preceded by a journal paper while, in the second case, this restriction is dropped. In both cases, a conference paper exists only if at least one journal paper exists.
4. $(q_{Sub}(q_{Year}(q_{JPaper} \mid q_{ConfPaper})^+)^*)$:
Journal and conference papers can exist alone. However, at least one of them should exist per year, *i.e.,* authors must have one publication per year.

Given these options, the administrator can choose the one that fits the best the application. □

From the above example, we can notice that our goal is to propose several choices of regular expressions, trying to foresee the needs of an application. Indeed, each candidate regular expression E' corresponds to a language $L(E')$ more general than $L(E) \cup \{w'\}$ (where E is the original regular expression and $w' \notin L(E)$). We are neither interested in the candidate $E|w'$ that adds just one word to *L(E),* nor in candidates too general allowing any kind of updates[2]. Our interest concerns candidates E' such that they are as similar to E as possible. Notice that in the previous example, each proposed regular expression have just one alphabet symbol inserted, in relation to the original one. This condition will be reflected by a very simple notion of distance, defined in Section 2.

Each transition rule of $\mathcal{A}$ has the general form $a, S, E \rightarrow q$ where a is a label, S is a tuple of two disjoint sets of states establishing attribute constraints ($S = <S_{optional}, S_{compulsory}>$ with $S_{optional}$ for optional attributes and $S_{compulsory}$ for compulsory ones), E is a regular expression establishing element constraints and q is a state. A run of $\mathcal{A}$ on a tree starts its computation at the leaves and then simultaneously works up the paths of the tree. The tree automaton accepts a tree t if all the attributes and element constraints defined in $\mathcal{A}$ (via the transition rules) hold in t. The insertion of a labeled tree t_a at position p of tree t can provoke changes on S (if the root of t_a is an attribute) or on E (if the root of t_a is an element or data). We concentrate on the changes occurring on E, *i.e.,* we only consider the insertion of sub-elements. Moreover, we only consider the insertion of one sub-element at a time.

Given a transition rule $a, S, E \rightarrow q$. Let $w = \alpha\beta$ be the word formed by the concatenation of the states associated to the element children of the position p, in a valid XML tree t. Thus, w belongs to the language *L(E).* The insertion of t_a as a child of position p, in t, corresponds to the construction of a new word $w' = \alpha n \beta$ (always associated to the sub-elements of p). Notice that n corresponds to the state that $\mathcal{A}$ associates to the root of t_a. If w' is not in *L(E)* then the rule $a, S, E \rightarrow q$ cannot be applied after the update. Our approach consists in computing new regular expressions to replace E according to the structure imposed by E *(i.e.,* number of starred sub-expression, disjoint symbols, etc.) and the characteristics of the unrecognized word w'. Thus, we propose a method that,

[2] As, for instance, a method that gives $E' = a^*b^*$ as the result for $E = ab$, $w = ab$ and $w' = aab$.

given a regular expression E and a new word w' to accept, *(i)* computes a finite state automaton M_E associated to *E; (ii)* performs some modifications on M_E to obtain new automata M'_E that accepts w' and *(iii)* finds regular expressions E' associated to each M'_E. To avoid important changes in E, we propose new regular expressions that have the smallest distance from *E,* that preserve the syntactic nesting of E and that respect the structure of w'. To this end, we introduce an algorithm called GREC which is an extension of the transformation of a Glushkov automaton into a regular expression presented in [7].

Section 2 presents some theoretical notions necessary to understand the schema evolution method implemented by GREC which is introduced in Section 3. Proofs are omitted due to lack of space (see [3]).

2 Theoretical Background

In this section, we consider the method proposed in [7] to obtain a homogeneous[3] finite state automaton, called Glushkov automaton, that recognizes the language associated to a given regular expression. Given a regular expression, a Glushkov automaton is built by subscribing each alphabet symbol in this regular expression with its position. In a Glushkov automaton, each non initial state corresponds to a position in the regular expression. For instance, given the regular expression $E = (a(b|c)^*)^*d$, the subscribed regular expression is $\overline{E} = (a_1(b_2|c_3)^*)^*d_4$. The Glushkov automaton $M = (\Sigma, Q, \Delta, q_0, F)$, built from *E,* is such that: the alphabet is $\Sigma = \{a, b, c, d\}$, the set of states is $Q = \{0,1,2,3,4\}$, the initial state is $q_0 = 0$, the set of final states is $F = \{4\}$ and the transition relation Δ is defined by the edges of the graph in Fig. 2(a).

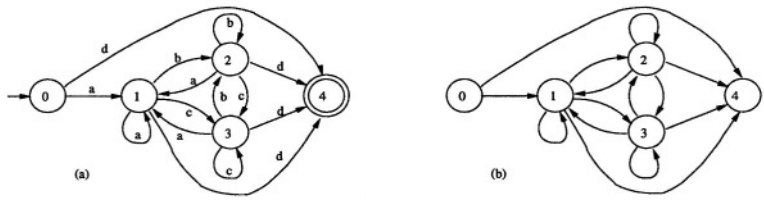


Fig. 2. (a) Pictorial representation of a FSA for $(a(b|c)^*)^*d$. (b) Its Glushkov graph

A *Glushkov graph* is the directed graph *G = (X, U)* obtained from a Glushkov automaton such that each node in *X* corresponds to a state and each edge in *U* to a transition. Since Glushkov automata are homogeneous, their edges are not decorated as shown in Fig. 2(b).

Let *G = (X, U)* be a graph. An edge between nodes r and s, denoted by $u = (r, s)$, is a *loop* iff $r = s$. A *path* is a sequence of nodes $x_0, \ldots, x_n$ such that,

[3] A finite state automaton is said to be *homogeneous* [7] if one always enters a given state by the same symbol.

for every $0 \leq i < n$, (x_i, x_{i+1}) is an edge in G. A path with no edges is said to be trivial. A graph has a *root* node r (resp. an *antiroot*) if there exists a path from r to any node in the graph (resp. from any node in the graph to r). A graph is a *hammock* if it has both a root (r) and an antiroot (s), with $r \neq s$.

Now we consider the graph properties taken from [7], that will be used later on in this work. A set $\mathcal{O} \subseteq X$ is said to be an *orbit* if for all x and x' in $\mathcal{O}$ there exists a non trivial path from x to x'. An orbit is *maximal* if for each node x of $\mathcal{O}$ and for each node x' out of $\mathcal{O}$, there does not exist a path from x to x' and a path from x' to x. Notice that an orbit is maximal if it is not contained in any other orbit. Let $\mathcal{O}$ be an orbit, we define: $In(\mathcal{O})=\{x \in \mathcal{O} \mid \exists x' \in (X \setminus \mathcal{O}), (x', x) \in U\}$ as the *input* of $\mathcal{O}$ and $Out(\mathcal{O})=\{x \in \mathcal{O} \mid \exists x' \in (X \setminus \mathcal{O}), (x, x') \in U\}$ as the *output* of $\mathcal{O}$. An orbit $\mathcal{O}$ is *stable* if $\forall x \in Out(\mathcal{O})$ and $\forall y \in In(\mathcal{O})$, the edge (x, y) exists. An orbit $\mathcal{O}$ is *transverse* if $\forall x, y \in Out(\mathcal{O})$, $\forall z \in (X \setminus \mathcal{O}), (x, z) \in U \Rightarrow (y, z) \in U$, and if $\forall x, y \in In(\mathcal{O})$, $\forall z \in (X \setminus \mathcal{O}), (z, x) \in U \Rightarrow (z, y) \in U$. An orbit $\mathcal{O}$ is *strongly stable* (resp. *strongly transverse*) if it is stable (resp. transverse) and if after deleting the edges in $Out(\mathcal{O}) \times In(\mathcal{O})$, every suborbit is strongly stable (resp. strongly transverse).

Example 2. The graph (a hammock) of Fig. 2(b) has 7 orbits. The orbit $\mathcal{O}_1 = \{1, 2, 3\}$, with $In(\mathcal{O}_1) = \{1\}$ and $Out(\mathcal{O}_1) = \{1, 2, 3\}$, is maximal. Orbits $\mathcal{O}_2 = \{1, 2\}$ and $\mathcal{O}_3 = \{2, 3\}$ are not maximal. Orbit $\mathcal{O}_3$ is stable since all the edges in $Out(\mathcal{O}_3) \times In(\mathcal{O}_3)$ are in $\mathcal{O}_3$. It is transverse since all the edges (1, 2), (1, 3), (2,4) and (3,4) exist in the graph. In fact, $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ are strongly stable and strongly transverse. □

Given a graph G in which all orbits are strongly stable, we build a *graph without orbits SO(G)* by recursively deleting, for each maximal orbit $\mathcal{O}$, all edges (x, y) such that $x \in Out(\mathcal{O})$ and $y \in In(\mathcal{O})$. The process ends when there are no more orbits. Notice that *SO(G)* is defined in a unique way [7].

Let x be a node in a graph without orbits $G = (X, U)$. We denote by $Q^-(x) = \{y \in X \mid (y, x) \in U\}$ the set of immediate predecessors of x and by $Q^+(x) = \{y \in X \mid (x, y) \in U\}$ the set of immediate successors of x. The graph G is reducible if it is possible to reduce it to one node by successive applications of any of the rules $\mathbf{R_1}$, $\mathbf{R_2}$ and $\mathbf{R_3}$ below (as illustrated in Fig. 3).

Rule $\mathbf{R_1}$: If two nodes x and y are such that $Q^-(y) = \{x\}$ and $Q^+(x) = \{y\}$, then replace node x by node xy and delete node y.

Rule $\mathbf{R_2}$: If two nodes x and y are such that $Q^-(x) = Q^-(y)$ and $Q^+(x) = Q^+(y)$, then replace node x by node $x|y$ and delete node y.

Rule $\mathbf{R_3}$: If a node x is such that $y \in Q^-(x) \Rightarrow Q^+(x) \subset Q^+(y)$, then replace node x by node $x?$ and delete the edges going from $Q^-(x)$ to $Q^+(x)$.

Notice that each node of the graph has a regular expression (which, initially, is just the position identifying the node). At the end of the process, the graph has just one node whose content is the regular expression corresponding to the original Glushkov automaton. In this way, we obtain a regular expression from a Glushkov automaton.

If x is optional, by $\mathbf{R_3}$ we build a regular expression in the following way: If the original regular expression associated to x is of the form E^+ (resp. E) then

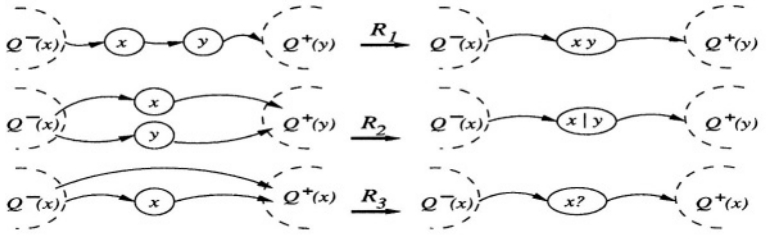


Fig. 3. Rules $\mathbf{R_1}$, $\mathbf{R_2}$ and $\mathbf{R_3}$

the new one will be E^* (resp. $E?$). From now on, we use the notation $E!$ to stand both for $E?$ and E^*. The node resulting from the application of one rule keeps the identities of its origins, responding for them. Notice that the above rules do not account to the construction of E^+ expressions. This kind of expression will appear by considering the orbits existing in the original graph. It is important to remark that the reduction process works from inside to outside of the nested maximal orbits. Maximal orbits are built during the construction of *SO(G)* and they are hierarchically organized according to the set-inclusion relation.

The characterization[4] of Glushkov FSA is given by the following theorem. Moreover, from Lemma 1 below, we can associate an orbit in a Glushkov graph to a Kleene closure in the corresponding regular expression.

Theorem 1. *[7] G = (X, U) is a Glushkov graph iff the following conditions are satisfied:* (1) *G is a hammock,* (2) *each maximal orbit in G is strongly stable and strongly transverse and* (3) *the graph without orbit of G is reducible.*

Lemma 1. *[7] Let G = (X,U) be a graph that satisfies the properties* (2) *and* (3) *of Theorem 1. Let $\mathcal{O}$ be a maximal orbit in G. By iteration of $\boldsymbol{R_1}$, $\boldsymbol{R_2}$ and $\boldsymbol{R_3}$ in SO(G), the orbit $\mathcal{O}$ is reduced to a unique node, under the assumption that $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ are only applied to pairs $(x,y) \in (\mathcal{O} \times \mathcal{O})$ or $(x,y) \in [(X \setminus \mathcal{O}) \times (X \setminus \mathcal{O})]$.*

We define now a very simple notion of the distance between two regular expressions, based on the number of positions of the subscribed expressions:

Definition 1. *Let E and E' be regular expressions. Let $\overline{E}$ and $\overline{E'}$ be subscripted expressions built from E and E', respectively, by using the Glushkov method. Let $S^{\overline{E}}$ (resp. $S^{\overline{E'}}$) be the set of positions of $\overline{E}$ (resp. $\overline{E'}$). The* distance *between E and E', denoted by $\mathcal{D}(E,E')$, is $\mathcal{D}(E,E') = || S^{\overline{E}} | - | S^{\overline{E'}} ||$.*

3 Schema Evolution by Changing Glushkov Graphs

We recall that, in our work, a schema is defined by an unranked bottom-up tree automaton $\mathcal{A}$ obtained from an (unambiguous) DTD [4] and that updates

[4] To characterize a Glushkov FSA, add an end mark (#) to every string [7].

are seen as changes to be performed on an XML tree as shown in Example 1. Invalid updates produce XML trees which cannot be recognized by $\mathcal{A}$. To deal with this kind of updates we present a method that proposes changes to the schema. If the update corresponds to the insertion of an attribute then the new schema is obtained just by allowing the existence of a new optional attribute. No attributes can be inserted as compulsory, otherwise the validity of documents with respect to the original schema is not preserved. Schema evolution due to a delete operation is also straightforward to define, since it consists in rendering optional the deleted attribute or sub-element. The challenge in this context is to consider the evolution of the schema caused by the insertion of an element. In this case, the schema evolution is achieved by changing the regular expression E that constraints the sub-elements of a given node in the XML tree. In this context, our problem can be expressed as follows: (1) The insertion of a labeled tree t_1 as a child of node a in tree t means changing the original word w obtained from the children of a. The new word w' is obtained from w by inserting the new state (associated to the root of t_1).

(2) Given the regular expression E (with $w \in L(E)$) and a word w', our problem is to propose new regular expressions E', such that $\mathcal{D}(E, E') = 1$, and whose languages contain, at least, the word w' and $L(E)$.

To work on words w and w', we use a Glushkov automaton M_E. This automaton is built by applying the method of [7], mentioned in Sect. 2, over each E that appears in the transition rules of $\mathcal{A}$. In M_E each state (but the initial one) corresponds to a position in the subscribed regular expression $\overline{E}$. The only final state of M_E is subscribed with the position of the end mark (#).

We consider now the execution of M_E over the new word w'. Let p be the position of w' where the new symbol is inserted. We define the *nearest left state* (s_{nl}) as a state in M_E reached after reading the first $p-1$ symbols in w' (or in w). Similarly, we define the *nearest right state* (s_{nr}) as a state in M_E that succeeds s_{nl} when reading the p-th position of w. Notice that to determine nodes s_{nl} and s_{nr} (to be passed to GREC), we scan w' using M_E. If the inserted symbol already belongs to the alphabet of M_E, a simple backtracking technique may be used to identify s_{nl} and s_{nr} (see [3] for details). Notice that both s_{nl} and s_{nr} exist and, when M_E is deterministic (as usually recommended in XML domain), they are unique.

Without loss of generality, we assume that an insertion operation always corresponds to the insertion of a new position in E. Thus, to accept the new word, we should insert a new state in M_E. This new state (s_{new}) should be added to M_E and there should exist a transition from s_{nl} to s_{new}. However this is not the only change to be performed on M_E. Other changes are needed in order to keep the graph associated to the automaton as a Glushkov graph. These changes depend on the situation of s_{nl} and s_{nr} in the Glushkov graph.

For a general regular expression E, the task of finding the places where the new symbol may be added is not trivial. There is a great variety of possible solutions and it is hard to find those that fit the best in a given context. As shown in Example 1, we want that the candidates respect the nesting of sub-

expressions of the original regular expression. The reduction process of [7] is well adapted to our goal of proposing solutions that preserve the general structure of the original regular expression E, since it follows the syntactic nesting of E using the orbits. Moreover, inserting a new state in M_E means inserting just one new position in the corresponding $\overline{E}$. In other words, our approach proposes only new regular expressions E' such that $\mathcal{D}(E, E') = 1$.

Each reduction step in [7] consists in replacing part of the automaton graph by nodes containing more complex regular expressions. The reduction finishes when the automaton graph is formed by just one node containing one regular expression, which corresponds to the original regular expression. Our goal is to build a new graph G' from a Glushkov graph G by preserving the Glushkov properties (Theorem 1).

Fig. 4 presents a high level algorithm for the procedure GREC (**G**enerate **R**egular **E**xpression **C**hoices), which is an extension of the method of [7] (explained in Sect. 2). GREC generates a list of regular expressions, each of which corresponds to a solution obtained by the insertion (in different places) of s_{new} in the original graph. The generated list contains the options we give to the administrator. GREC takes five input parameters: a graph without orbits G_A, a hierarchy of orbits O_A, two nodes of the graph, corresponding to s_{nl} and s_{nr}, and the new node s_{new} to be inserted.

```
(1) procedure GREC(G_A, O_A, s_nl, s_nr, s_new) {
(2)  if graph G_A has only one node
(3)    then stop
(4)    else{
(5)     R_i := ChooseRule(G_A, O_A);
(6)     for each (G_B,O_B):
        =LookForGraphAlternative(G_A,O_A,R_i,s_nl,s_nr,s_new)do
(7)         GraphToRegExp(G_B, O_B);
(8)     G_C := ApplyRule(R_i, G_A);
(9)     GREC(G_C,O_A,s_nl,s_nr,s_new);
(10)   }     }
```

Fig. 4. Algorithm to generate regular expression choices from a Glushkov graph

The Procedure ChooseRule uses the information concerning orbits to select a rule to be applied in the reduction of the graph. The Procedure ApplyRule computes a new graph resulting from the application of the selected rule, and the Procedure GraphToRegExp computes a regular expression from a given graph.

At each step of the reduction, GREC checks whether the chosen rule affects nodes s_{nl} or s_{nr} and, in this case, it modifies the graph to take into account the insertion of the new node s_{new}. The modifications to the graph being reduced are driven by $\mathbf{R_1}$, $\mathbf{R_2}$ and $\mathbf{R_3}$ and by the information concerning the orbits of the original graph. Each of these modifications is performed by the iterator LookForGraphAlternative (line (6) of Fig. 4). The iterations stop when no

more alternatives are found. The role of iterator `LookForGraphAlternative` is two-fold: *(i)* it verifies whether nodes s_{nl} and s_{nr} satisfy the conditions stated in $\mathbf{R}_1$, $\mathbf{R}_2$ and $\mathbf{R}_3$ and *(ii)* it generates new graphs (G_B in the algorithm of Fig. 4) over which the algorithm `GraphToRegExp` is applied to generate new regular expressions. The following definitions formalize how `LookForGraphAlternative` builds new graphs from the original one.

Definition 2. *Let $G_A = (X_A, U_A)$ be a Glushkov graph and $x, y \in X_A$ be two nodes on which $\mathbf{R}_1$ can be applied. Define each $G_i = (X_i, U_i)$ from G_A as follows:*

Case 1 ($x = s_{nl}$ *and* $y = s_{nr}$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(x, s_{new})\} \cup \{(s_{new}, y)\}$.
Case 2 ($x = s_{nr}$ *and* $y = s_{nl}$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(y, s_{new})\} \cup \{(s_{new}, z) \mid z \in Q^+(y)\}$.
G_2: $X_2 = X_A \cup \{s_{new}\}$; $U_2 = U_A \cup \{(s_{new}, x)\} \cup \{(z, s_{new}) \mid z \in Q^-(x)\}$.

Definition 3. *Let $G_A = (X_A, U_A)$ be a Glushkov graph and $x, y \in X_A$ be two nodes on which $\mathbf{R}_2$ can be applied. Define each $G_i = (X_i, U_i)$ from G_A as follows:*

Case 1 ($x = s_{nl}$ *and* $s_{nr} \in Q^+(x)$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(x, s_{new})\} \cup \{(s_{new}, z) \mid z \in Q^+(x)\}$.
Case 2 ($x = s_{nr}$ *and* $s_{nl} \in Q^-(x)$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(s_{new}, x)\} \cup \{(z, s_{new}) \mid z \in Q^-(x)\}$.
Case 3 ($s_{nl} \in Q^-(x)$ *and* $s_{nr} \in Q^+(x)$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(z, s_{new}) \mid z \in Q^-(x)\} \cup \{(s_{new}, v) \mid v \in Q^+(x)\}$.

Definition 4. *Let $G_A = (X_A, U_A)$ be a Glushkov graph and $x \in X_A$ be a node on which $\mathbf{R}_3$ can be applied. Define each $G_i = (X_i, U_i)$ from G_A as follows:*

Case 1 ($s_{nl} \in Q^-(x)$ *and* $s_{nr} \in Q^+(x)$) *or* ($s_{nl} \in Q^-(x)$ *and* $s_{nr} \notin Q^+(x)$) *or* ($s_{nl} \notin Q^-(x)$ *and* $s_{nr} \in Q^+(x)$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(s_{new}, x)\} \cup \{(z, s_{new}) \mid z \in Q^-(x)\}$.
G_2: $X_2 = X_A \cup \{s_{new}\}$; $U_2 = U_A \cup \{(x, s_{new})\} \cup \{(s_{new}, z) \mid z \in Q^+(x)\}$.
G_3: $X_3 = X_A \cup \{s_{new}\}$; $U_3 = U_A \cup \{(z, s_{new}) \mid z \in Q^-(x)\} \cup \{(s_{new}, v) \mid v \in Q^+(x)\}$.

Rules $\mathbf{R}_1$, $\mathbf{R}_2$ and $\mathbf{R}_3$ are first applied inside each orbit [7]. During the reduction process, each orbit $\mathcal{O}$ of the original graph is reduced to just one node containing a regular expression. This regular expression is then decorated by $^+$. Before applying this decoration we have to consider the insertion of s_{new} in the orbit $\mathcal{O}$. The next definition summarizes the situations in which we perform a modification on an orbit. It gives the conditions and modifications concerning the cases in which a whole orbit is represented by one node of the graph.

Definition 5. *Let $G_A = (X_A, U_A)$ be a Glushkov graph. Let $\mathcal{O}$ be an orbit reduced to one node $x \in X_A$. We define each $G_i = (X_i, U_i)$ from G_A as follows:*

Case 1 ($x = s_{nl}$ *and* $x = s_{nr}$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(s_{new}, x)\} \cup \{(z, s_{new}) \mid z \in Q^-(x)\}$.
G_2: $X_2 = X_A \cup \{s_{new}\}$; $U_2 = U_A \cup \{(x, s_{new})\} \cup \{(s_{new}, z) \mid z \in Q^+(x)\}$.
Case 2 ($s_{nl} \in Q^-(x)$ *and* $s_{nr} \in \mathcal{O}$):

G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(s_{new}, x)\} \cup \{(z, s_{new}) \mid z \in Q^-(x)\}$.
G_2: $X_2 = X_A \cup \{s_{new}\}$; $U_2 = U_A \cup \{(v, s_{new}) \mid v \in Q^-(x)\} \cup \{(s_{new}, z) \mid z \in Q^+(x)\}$.
Case 3 ($s_{nl} \in \mathcal{O}$ *and* $s_{nr} \in Q^+(x)$):
G_1: $X_1 = X_A \cup \{s_{new}\}$; $U_1 = U_A \cup \{(x, s_{new})\} \cup \{(s_{new}, z) \mid z \in Q^+(x)\}$.
G_2: $X_2 = X_A \cup \{s_{new}\}$; $U_2 = U_A \cup \{(v, s_{new}) \mid v \in Q^-(x)\} \cup \{(s_{new}, z) \mid z \in Q^+(x)\}$.
Moreover, in all cases, s_{new} *is added to the orbit* $\mathcal{O}$.

Notice that for each graph built using the definitions above a regular expression is obtained.

Example 3. Consider Example 1 and the Glushkov automaton (Fig. 5(a)) corresponding to $E = q_{Sub}(q_{Year}\ q_{JPaper}{}^+)^*\#$ (and $\overline{E} = 1\ (2\ 3^+)^*4$). In this case, GREC takes as input a hierarchy of orbits containing $\mathcal{O}_1 = \{3\}$ and $\mathcal{O}_2 = \{2, 3\}$. It starts the reduction by $\mathcal{O}_1$. At this step, Definition 5 (third case) applies since $\mathcal{O}_1$ is represented by just one node, $s_{nl} = 3$ and $s_{nr} = 4$. Graphs (without orbits) G_1 and G_2 (Fig. 5(b)-(c)) are built, giving rise to $(q_{Sub}(q_{Year}(q_{JPaper}\ q_{ConfPaper}!)^+)^*)$ and $(q_{Sub}(q_{Year}(q_{JPaper} \mid q_{ConfPaper})^+)^*)$, respectively (options 3 and 4 from Example 1).

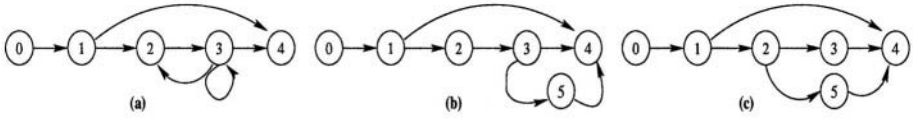


Fig. 5. (a)Automaton. (b)-(c) Graphs updated by `LookForGraphAlternative`

GREC solutions respect the properties stated by the following theorems.

Theorem 2. *Let G be Glushkov graph and* $G_A = SO(G)$. *Let* O_A *be the hierarchy of orbits obtained during the construction of* G_A. *Let* R_i *be one of the reduction rules* $\boldsymbol{R_1}$, $\boldsymbol{R_2}$ *or* $\boldsymbol{R_3}$. *For any nodes* s_{nl}, s_{nr} *and* s_{new}, *each pair* (G_B, O_B) *resulting from the execution of* `LookForGraphAlternative` $(G_A, O_A, R_i, s_{nl}, s_{nr}, s_{new})$ *is a representation of a Glushkov graph* G', *where* G_B *is a graph without orbits, and* O_B *is the hierarchy of orbits obtained when constructing* G_B *from* G'.

Theorem 3. *Let E be a regular expression and L(E) be the language associated to E. Given* $w \in L(E)$ *such that* $w = \alpha\beta$, *let* $w' = \alpha n \beta$ *where* n *is a symbol and* $w' \notin L(E)$. *Let* M_E *be a deterministic Glushkov automaton corresponding to* E, *let* G_A *be the graph without orbits obtained from* M_E *and let* O_A *be the hierarchy of orbits obtained during the construction of* G_A. *Let* s_{nl} *be a state in* M_E *reached after reading* α *and let* s_{nr} *be a state that succeeds* s_{nl} *in* M_E *when reading* w. *Let* s_{new} *be a new node not in* G_A. *The execution of* GREC $(G_A, O_A, s_{nl}, s_{nr}, s_{new})$ *returns a finite, nonempty set of regular expressions* $\{E_1, \ldots, E_m\}$. *For each* E_i, *we have* $L(E) \cup \{w'\} \subset L(E_i)$ *and* $\mathcal{D}(E, E_i) = 1$.

If unambiguous expressions are required as a result, GREC signalizes the ambiguity of a candidate regular expression - we can then transform the chosen candidate into an equivalent unambiguous regular expression along the lines of [1]. Notice that if E is unambiguous and n is not in E then each candidate regular expression E_i given by GREC is unambiguous.

4 Conclusion

In this paper we propose a method to dynamically change the schema for XML databases based on an update of just one document. Our approach is original in the sense that it does not impose changes on documents, but rather, computes a set of new schema that preserve the consistency of the document and that has minimal changes in relation to original regular expression. GREC solutions are ordered according to the hierarchy of orbits (given as input) which can define different context of updates (see [3] for details). We have implemented a prototype of GREC using the ASF+SDF [6] meta-environment under Linux. We are currently considering the following research directions: *(i)* The generalization of the schema evolution process discussed here, to consider a sequence (or a set) of administrator's updates and *(ii)* the implementation of an XML update language such that UpdateX [11] in which incremental schema evolution will be integrated.

Acknowledgements. We would like to thank the anonymous referees for their useful comments and corrections.

References

1. H. Ahonen. Disambiguation of SGML content models. In *PODP,* volume 1293 of *Lecture Notes in Computer Science.* Springer, 1997.
2. D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.,* 75(2):87–106, 1987.
3. B. Bouchou, D. Duarte, M. Halfeld Ferrari Alves, D. Laurent, and M. Musicante. Evolving schemas for XML: An incremental approach. Technical Report (To appear), Université de François Rabelais, LI, 2004.
4. B. Bouchou, D. Duarte, M. Halfeld Ferrari Alves, and D. Laurent. Extending tree automata to model XML validation under element and attribute constraints. In *ICEIS,* 2003.
5. B. Bouchou and M. Halfeld Ferrari Alves. Updates and incremental validation of XML documents. In *The 9th DBPL,* number 2921 in LNCS, 2003.
6. M. G. J. van den Brand, J. Heering, P. Klint, and P. A. Olivier. Compiling rewrite systems: The ASF+SDF compiler. *ACM, Transactions on Programming Languages and Systems,* 24, 2002.
7. P. Caron and D. Ziadi. Characterization of Glushkov automata. *TCS: Theorical Computer Science,* 233:75–90, 2000.
8. E. Kuikka, P. Leinonen, and M. Penttonen. An approach to document structure transformations. In *Proceedings of Conference on Software: Theory and Practice, pp. 906-913.,* 2000.

9. R. Parekh and V. Honavar. Learning DFA from simple examples. *Machine Learning,* 44(1/2):9–35, 2001.
10. J. Roddick, L. Al-Jadir, L. Bertossi, M. Dumas, F. Estrella, H. Gregersen, K. Hornsby, J. Lufter, F. Mandreoli, T. Männistö, E. Mayol, and L. Wedemeijer. Evolution and change in data management - issues and directions. *SIGMOD Record,* 29(1):21–25, 2000.
11. G. M. Sur, J. Hammer, and J. Simeon. An XQuery-based language for processing updates in XML. In *PLAN-X,* 2004.
12. H. Su, D. Kramer, L. Chen, K. T. Claypool, and E. A. Rundensteiner. XEM: Managing the evolution of XML documents. In *RIDE-DM,* pages 103–110, 2001.
13. H. Su, H. Kuno, and E. A. Rundensteiner. Automating the transformation of XML documents. In *3rd WIDM.* ACM, 2001.

Complexity of Decision Problems for Simple Regular Expressions

Wim Martens[1], Frank Neven[1], and Thomas Schwentick[2]

[1] Limburgs Universitair Centrum
Universitaire Campus
B-3590 Diepenbeek, Belgium
{wim.martens,frank.neven}@luc.ac.be
[2] Philipps Universität Marburg
Fachbereich 12, Mathematik und Informatik
tick@informatik.uni-marburg.de

Abstract. We study the complexity of the inclusion, equivalence, and intersection problem for simple regular expressions arising in practical XML schemas. These basically consist of the concatenation of factors where each factor is a disjunction of strings possibly extended with '$*$' or '?'. We obtain lower and upper bounds for various fragments of simple regular expressions. Although we show that inclusion and intersection are already intractable for very weak expressions, we also identify some tractable cases. For equivalence, we only prove an initial tractability result leaving the complexity of more general cases open. The main motivation for this research comes from database theory, or more specifically XML and semi-structured data. We namely show that all lower and upper bounds for inclusion and equivalence, carry over to the corresponding decision problems for extended context-free grammars and single-type tree grammars, which are abstractions of DTDs and XML Schemas, respectively. For intersection, we show that the complexity only carries over for DTDs.

1 Introduction

XML (eXtensible Markup Language) is becoming the standard data exchange format for the Web. Within a community, parties usually agree to only produce XML data conforming to a certain format. The presence of such a schema improves the efficiency of many tasks like, for instance, query processing, query optimization, and automatic data integration. For typechecking or type inference [11,14,18,21], schema information is even crucial. As standard decision problems of schema languages, like inclusion, equivalence, and non-emptiness of intersection, are among the basic building blocks for many of the algorithms for those problems, it is critical to establish their exact complexity.

Among the various proposals for XML schema languages, DTD (Document Type Definition) and XML Schema Definitions (XSDs) [8,9] are the most widely spread. Generally these languages are abstracted by extended context-free

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 889–900, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Table 1. Possible factors in simple regular expressions and how they are denoted ($a \in \Sigma$, $w \in \Sigma^*$).

Factor	Abbr.
a	a
a^*	a^*
$a?$	$a?$
$(a_1 + \cdots + a_n)$	$(+a)$

Factor	Abbr.
$(a_1 + \cdots + a_n)^*$	$(+a)^*$
$(a_1 + \cdots + a_n)?$	$(+a)?$
$w?$	$w?$
w^*	w^*

Factor	Abbr.
$(a_1^* + \cdots + a_n^*)$	$(+a^*)$
$(w_1 + \cdots + w_n)$	$(+w)$
$(w_1 + \cdots + w_n)?$	$(+w)?$
$(w_1 + \cdots + w_n)^*$	$(+w)^*$

grammars (ECFGs) and unranked tree automata, respectively [20,26]. The former are context-free grammars with regular expressions as right-hand sides of rules, while the latter are a natural extension of classical tree automata to trees where nodes can have an unbounded number of children [3]. A formalism equivalent to such tree automata but which is grammar based are specialized DTDs (SDTDs) [21]. The complexity of the three afore mentioned problems is known and is PSPACE-complete for DTDs (as it reduces to the corresponding problems of regular expressions) and EXPTIME-complete for tree automata or specialized DTDs [13,22,23,24].

In the present paper, we revisit the complexity of the inclusion, equivalence, and intersection problem for XML schemas occurring effectively in practice. For instance, the PSPACE-hardness of inclusion of DTDs crucially depends on the presence of involved regular expressions that are quite unlikely to occur in realistic DTDs. Actually, a study by Bex, Neven, and Van den Bussche [2] reveals that more than 90 percent of the regular expressions occurring in practical DTDs and XSDs are of the following simple form: $e_1 \cdots e_n$ where every e_i is a factor of the form $(w_1 + \cdots + w_n)$ possibly extended with Kleene-star or question mark, and each w_i is a string (cf. Section 2 for a detailed definition and Table 1 for a list of allowed factors together with their abbreviated notation). Further, Murata et al. argued that XML Schemas do not correspond to the full class of tree automata or SDTDs, but to a strict subset of those, namely, single-type SDTDs [19].

Clearly, complexity lower bounds for the inclusion, equivalence, or the intersection problem for a class of regular expressions $\mathcal{R}$ imply lower bounds for the corresponding decision problems for DTDs and (single-type) SDTDs with right-hand sides in $\mathcal{R}$. Interestingly, we show that for inclusion and equivalence, also upper bounds for the string case carry over to DTDs and single-type SDTDs. For intersection, the latter still holds for DTDs but not for single-type SDTDs. So, it suffices to restrict attention to the complexity of simple regular expressions to derive complexity bounds for XML schema languages.

Our results on the complexity of simple regular expressions are summarized in Table 2. We denote by $\mathrm{RE}(\mathcal{S})$, the set of all simple regular expressions. Recall that the three decision problems are PSPACE-complete for the class of all regular expressions [13,24]. We briefly discuss our results.

- We show that inclusion is already CONP-complete for very innocent expressions: where every factor is of the form a or a^*, or of the form a or $a?$

Table 2. Summary of Results. Theorem numbers are given in brackets.

RE-fragment	Inclusion	Equivalence	Intersection
a, a^+	in PTIME (DFA!)	in PTIME	in PTIME (9)
a, a^*	CONP-complete (4)	in PTIME (6)	NP-complete (7)
$a, a?$	CONP-complete (4)	in PTIME (6)	NP-complete (7)
$\mathcal{S} - \{(+a)^*, (+w)^*\}$	CONP-complete (4)	in CONP	NP-complete (7)
$a, (+a)^*$	PSPACE-complete (4)	in PSPACE	NP-complete (7)
$\mathcal{S} - \{(+w)^*\}$	PSPACE-complete (4)	in PSPACE	NP-complete (7)
$\mathcal{S}$	PSPACE-complete (4)	in PSPACE	in PSPACE
$\mathrm{RE}^{\leq k}$ $(k \geq 3)$	PTIME (5)	PTIME	PSPACE-complete (8)
one-unambiguous	in PTIME	in PTIME	PSPACE-complete (8)

with a an arbitrary alphabet symbol. Even worse, when factors of the form $(a_1 + \cdots + a_n)^*$ are added we already obtain the maximum complexity: PSPACE. When such factors are disallowed the complexity remains CONP. If the number of occurrences of the same symbol in expressions is bounded by some k ($\mathrm{RE}^{\leq k}$), inclusion is in PTIME. As the running time is n^k, k should of course be small to be feasible.

- The precise complexity of the equivalence problem largely remains open. Of course, it is never harder than inclusion but we conjecture that it is tractable for a large fragment of $\mathrm{RE}(\mathcal{S})$. We only prove a PTIME upper bound for expressions where each factor is a or a^*, or a or $a?$. Even for these restricted fragments the proof is non-trivial. Basically, we show that two expressions are equivalent iff they have the same sequence normal form modulo one rewrite rule. Interestingly, the sequence normal form specifies factors much in the same way as XML Schema does. For every symbol an explicit upper and lower bound is specified. For instance, $aa^*bbc?c?$ becomes $a[1,*]b[2,2]c[0,2]$.
- Intersection is also CONP-hard when each factor is either of the form a or a^*, or of the form a or $a?$. However, the complexity is not always the same as for inclusion. In fact, there are cases where inclusion is harder and others where intersection is harder. Indeed, intersection remains in CONP even if we allow all kinds of factors except $(w_1 + \cdots + w_n)^*$. On the other hand, the intersection problem is PSPACE-hard for $\mathrm{RE}^{\leq 3}$. The only tractable fragment we obtain is when each factor is restricted to a or a^+.

The complexities of equivalence, inclusion and intersection for general regular expressions and several fragments were studied in [12,13,24]. From these, the most related result is the CONP-completeness of equivalence and inclusion of bounded languages [12]. A language L is bounded if there are strings $v_1, \ldots, v_n$ such that $L \subseteq v_1^* \cdots v_n^*$. It should be noted that the latter is much more general than, e.g., our $\mathrm{RE}(w^*)$. More recently, two fragments of simple regular expressions have been shown to be tractable: inclusion for $\mathrm{RE}(a?, (+a)^*)$ [1], and $\mathrm{RE}(a, \Sigma, \Sigma^*)$ [17]. This last result should be contrasted with the PSPACE-completeness of inclusion for $\mathrm{RE}(a, (+a), (+a)^*)$.

We conclude by a remark on one-unambiguous or deterministic regular expressions. Basically, these are regular expressions which have a deterministic Glushkov automaton [4]. The XML specification requires DTD content models to be deterministic because of compatibility with SGML (Section 3.2.1 of [8]). Of course, for such expressions, inclusion and equivalence are in PTIME. Nevertheless, intersection remains PSPACE-hard (cf. Theorem 8). Unfortunately, the notion of deterministic content models is not a transparent one for the average user, as is witnessed by practical studies [7,2] which found a number of non-deterministic content models in actual DTDs. Actually, for this very reason Clarke and Murata abandoned the notion in their Relax NG specification [25]. Hence, from a scientific viewpoint, we believe it makes sense to study the broader class of possibly non-deterministic but simple and practical regular expressions. Another consequence of our results, independent of the one-unambiguous issue, is that optimization problems for navigational queries as expressed by caterpillar expressions [5], $\mathcal{X}_{\mathrm{reg}}^{\mathrm{CPath}}$ and $\mathcal{X}_{\mathrm{reg}}$ [16], or regular path queries [6] quickly turn intractable. Due to space limitations many proofs are omitted. We refer the interested reader to [15].

2 Definitions

Regular expressions. For the rest of the paper let Σ denote a finite alphabet. A Σ-string (or simply string) is a finite sequence $w = a_1 \cdots a_n$ of Σ-symbols. We denote the length of w by $|w|$. The empty string is denoted by ε. The set of all strings is denoted by Σ^*. The syntax and semantics of regular expressions is defined in the usual way. By $L(r)$ we denote the language defined by regular expression r. By $r?$ and r^+, we abbreviate the expressions $r + \varepsilon$ and rr^*. Sometimes, we denote $w \in L(r)$ simply by $w \in r$.

We consider simple regular expressions occurring in practice in DTDs and XML Schemas [7]. The vast majority of these can be defined as follows.

Definition 1. A *base symbol* is a regular expression s, $s?$, or s^* where s is a non-empty string; a *factor* is of the form e, e^*, or $e?$ where e is a disjunction of base symbols. A *simple regular expression* is ε, $\emptyset$, or a sequence of factors.

An example of a simple regular expression is $((abc)^* + b^*)(a + b)?(ab)^*(ac + b)^*$. We introduce a uniform syntax to denote subclasses of simple regular expressions by specifying the allowed factors. We distinguish whether the string s of a base symbol consists of a single symbol (a) or a string (w) and whether it is extended by ? or $*$. Further, we distinguish between factors with one disjunct or with arbitrarily many disjuncts, the latter is denoted by $(+\cdots)$. Finally, factors can again be extended by $*$ or ?. A list of possible factors is displayed in Table 1. This table only shows types that are different. E.g., we write $\mathrm{RE}((+a)^*, w?)$ for the set of regular expressions $e_1 \cdots e_n$ where every e_i is $(a_1 + \cdots + a_n)^*$ for some $a_1, \ldots, a_n \in \Sigma$ and $n \geq 1$, or $w?$ for some $w \in \Sigma^*$.

If $A = \{a_1, \ldots, a_n\}$ is a set of symbols, we often denote $(a_1 + \ldots + a_n)$ simply by A. We denote the class of all simple regular expressions by $\mathrm{RE}(\mathcal{S})$.

Decision problems. Two regular expressions r, r' are *included,* denoted $r \subseteq r'$ iff $L(r) \subseteq L(r')$. They are *equivalent,* denoted $r \equiv r'$, if $L(r) = L(r')$. The following three problems are fundamental to this paper.

Definition 2. Let $\mathcal{R}$ be a class of regular expressions.

- The *inclusion problem* for $\mathcal{R}$, is to test for two given expressions $r, r' \in \mathcal{R}$, whether $r \subseteq r'$.
- The *equivalence problem* for $\mathcal{R}$, is to test for two expressions $r, r' \in \mathcal{R}$, whether $r \equiv r'$.
- The *intersection problem* for $\mathcal{R}$, is to determine for an arbitrary number of expressions $r_1, \ldots, r_n \in \mathcal{R}$, whether $\bigcap_{i=1}^{n} L(r_i) \neq \emptyset$.

In the sequel, we abuse notation and denote $\bigcap_{i=1}^{n} L(r_i) \neq \emptyset$ simply by $\bigcap_{i=1}^{n} r_i \neq \emptyset$. For arbitrary regular expressions these problems are PSPACE-complete [13,24].

3 Decision Problems for DTDs and XML Schemas

As explained in the introduction, our main motivation for this study comes from reasoning about XML schemas. In this section, we describe how the basic decision problems for such schemas, namely whether two schemas describe the same set of documents or whether one describes a subset of the other, basically reduce to the equivalence and inclusion problem for regular expressions. We also address the problem whether a set of schemas define a common XML document. W.r.t. DTDs the latter problem again reduces to the corresponding problem for regular expressions; for XML Schemas it does not. The reader not interested in XML can safely skip this section.

3.1 XML Schema Languages

It is common to view XML documents as finite trees with labels from a finite alphabet Σ. There is no limit on the number of children of a node. Of course, elements in XML documents can also contain references to nodes. But as XML schema languages usually do not constrain these nor the data values at leaves, it is safe to view schemas as simply defining tree languages over a finite alphabet. In the rest of this section, we introduce the necessary background concerning XML schema languages. For a formal definition of Σ-trees see, e.g., [20].

Definition 3. A *DTD* is a pair (d, s_d) where d is a function that maps Σ-symbols to regular expressions and $s_d \in \Sigma$ is the start symbol. We usually simply denote (d, s_d) by d. A tree t satisfies d if its root is labeled by s_d and, for every node u with label a, the sequence $a_1 \cdots a_n$ of labels of its children is in $L(d(a))$. By $L(d)$ we denote the set of trees defined by d.

A simple example of a DTD defining the inventory of a store is the following: store $\rightarrow$ dvd dvd*, dvd $\rightarrow$ title price.

For clarity, in examples we write $w \rightarrow r$ rather than $d(w) = r$. We recall the definition of a specialized DTD from [21].

Definition 4. A *specialized DTD* (SDTD) is a 4-tuple $\mathbf{d} = (\Sigma, \Sigma', d, \mu)$, where Σ' is an alphabet of *types,* d is a DTD over Σ' and μ is a mapping from Σ' to Σ. Note that μ can be extended to define a homomorphism on trees. A tree t then *satisfies* a specialized DTD if $t = \mu(t')$ for some $t' \in L(d)$. Again, we denote by $L(\mathbf{d})$ the set of trees defined by **d.**

The class of tree languages defined by SDTDs corresponds precisely to the regular (unranked) tree languages [3]. For ease of exposition, we always take $\Sigma' = \{a^i \mid 1 \leq i \leq k_a, a \in \Sigma, i \in \mathbb{N}\}$ for some natural numbers k_a and set $\mu(a^i) = a$. If a node is labeled by some a^i we call a^i the *type* of the node. A simple example of an SDTD is the following:

$$\begin{aligned} &\text{store}^1 \rightarrow (\text{dvd}^1 + \text{dvd}^2)^*\text{dvd}^2(\text{dvd}^1 + \text{dvd}^2)^*\text{dvd}^2(\text{dvd}^1 + \text{dvd}^2)^* \\ &\text{dvd}^1 \rightarrow \text{title}^1\ \text{price}^1 \qquad\qquad \text{dvd}^2 \rightarrow \text{title}^1\ \text{price}^1\ \text{discount}^1 \end{aligned}$$

Here, dvd^1 defines ordinary DVDs while dvd^2 defines DVDs on sale. The rule for store specifies that there should be at least two DVDs on discount. The following restriction on SDTDs corresponds to the expressiveness of XML schema languages [19].

Definition 5. A *single-type SDTD* (SDTD^{st}) is an SDTD $(\Sigma, \Sigma', d, \mu)$ with the property that in each regular expression $d(a)$ no two types b^i and b^j with $i \neq j$ occur.

The above defined SDTD is not single type as both dvd^1 and dvd^2 occur in the rule for store. The classes of tree languages defined by the grammars introduced above are included as follows: $\text{DTD} \subsetneq \text{SDTD}^{\text{st}} \subsetneq \text{SDTD}$ [3,19].

Remark 1. There is a very simple deterministic algorithm to check validity of a tree t with respect to a SDTD^{st} **d.** It proceeds top-down and assigns to every node with some symbol a a type a^i. To the root the start symbol of d is assigned; then, for every interior node u with type a^i, it is checked whether the children of u match $\mu(d(a^i))$; if not the tree is rejected, otherwise, as **d** is single-type, to each child a unique type can be assigned. The tree is accepted, if this process terminates at the leaves without any rejection. □

The *inclusion problem, equivalence problem* and *intersection problem* for DTDs, SDTDs and SDTD^{st}s is defined analogously to the corresponding problems for regular expressions.

3.2 Inclusion and Equivalence of XML Schema Languages

As already mentioned, testing equivalence and inclusion of XML schema languages is related to testing equivalence and inclusion of regular expressions. It is immediate that complexity lower bounds for regular expressions imply lower bounds for XML schema languages. A consequence is that testing equivalence and inclusion of XML schemas is PSPACE-hard, which suggests looking for simpler regular expressions.

Interestingly, in the case of the practically important DTDs and single-type SDTDs, it turns out that the complexities of the equivalence and inclusion problem on strings also imply *upper bounds* for the corresponding problems on XML trees.

For $\mathcal{R}$ a class of regular expressions, we denote by $\mathrm{DTD}(\mathcal{R})$, $\mathrm{SDTD}(\mathcal{R})$, and $\mathrm{SDTD}^{\mathrm{st}}(\mathcal{R})$, the class of DTDs, SDTDs and $\mathrm{SDTD}^{\mathrm{st}}$s with regular expressions in $\mathcal{R}$.

We call a complexity class $\mathcal{C}$ *closed under positive reductions* if the following holds for every $L \in \mathcal{C}$: Let L' be accepted by a deterministic polynomial-time Turing machine M with oracle O (denoted $L' = L(M^O)$). Let M further have the property that $L(M^A) \subseteq L(M^B)$ whenever $A \subseteq B$. Then L' is also in $\mathcal{C}$.

For a more precise definition of this notion we refer the reader to [10]. For our purposes it is sufficient that all important complexity classes like PTIME, NP, CONP, and PSPACE have this property and that every such class contains PTIME.

Theorem 1. *Let $\mathcal{R}$ be a class of regular expressions and $\mathcal{C}$ be a complexity class which is closed under positive reductions. Then the following are equivalent.*

(a) The inclusion problem for $\mathcal{R}$ expressions is in $\mathcal{C}$.
(b) The inclusion problem for $DTD(\mathcal{R})$ is in $\mathcal{C}$.
(c) The inclusion problem for $SDTD^{st}(\mathcal{R})$ is in $\mathcal{C}$.

The corresponding statement holds for the equivalence problem.

3.3 Intersection of DTDs and XML Schemas

We show in this section that the complexity of the intersection problem for regular expressions is an upper bound for the corresponding problem on XML trees. The latter is not the case for single-type DTDs.

Theorem 2. *Let $\mathcal{R}$ be a class of regular expressions and let $\mathcal{C}$ be a complexity class which is closed under positive reductions. Then the following are equivalent.*

(a) The intersection problem for $\mathcal{R}$ expressions is in $\mathcal{C}$.
(b) The intersection problem for $DTD(\mathcal{R})$ is in $\mathcal{C}$.

The following theorem shows that single-type SDTDs probably cannot be included in Theorem 2 as by Theorem 7, intersection of $\mathrm{RE}((+a), w?, (+a)?, (+a)^*)$ is in NP. The reduction is similar to the proof that intersection of deterministic top-down tree automata is EXPTIME-complete [23]. However, single-type DTDs and the latter automata are incomparable. Indeed, the tree language consisting of the trees $\{a(bc), a(cb)\}$ is not definable by a top-down deterministic tree automaton, while it is by the DTD consisting of the rules $a \to bc + cb$, $b \to \varepsilon$, $c \to \varepsilon$. Conversely, the tree language $\{a(b(c)b(d))\}$ is not definable by a single-type SDTD, but is definable by a top-down deterministic tree automaton.

Theorem 3. *The intersection problem for $SDTD^{st}((+a), w?, (+a)?, (+a)^*)$ is* EXPTIME*-hard.*

4 Inclusion of Regular Expressions

We start our investigation with the inclusion problem for simple regular expressions. As mentioned before, it is PSPACE-complete for general regular expressions. Some tractable cases have been identified in previous work: *(i)* in [1] it is shown that inclusion for $RE(a?, (+a)^*)$ can be solved in linear time. Whether $p \subseteq p_1 + \cdots + p_k$ for $p, p_1, \ldots, p_k$ from $RE(a?, (+a)^*)$ can be checked in quadratic time; and *(ii)* in [17] it is stated that inclusion for $\mathrm{RE}(a, \Sigma, \Sigma^*)$ is in PTIME.

Some of the fragments we defined look so innocent that one might expect their inclusion problem to be tractable. It therefore came as a surprise to find out that even for $RE(a, a?)$ and $RE(a, a^*)$ it is already CONP-complete. Even worse, for $RE(a, (+a)^*)$ we already obtain the maximum complexity, PSPACE-completeness. The latter should be contrasted with the PTIME inclusion for $\mathrm{RE}(a, \Sigma, \Sigma^*)$ obtained in [17], where disjunctions can only be over the complete alphabet. Our results, together with corresponding upper bounds are summarized in the following theorem. Let $\mathrm{RE}(\mathcal{S} - \{(+a)^*, (+w)^*\})$ denote the fragment of $\mathrm{RE}(\mathcal{S})$ where no factor of the form $(+a)^*$ or $(+w)^*$ is allowed.

Theorem 4. *The inclusion problem*

(a) is CONP*-hard for* $RE(a, a^*)$ *and* $RE(a, a?)$*;*
(b) is PSPACE*-hard for* $RE(a, (+a)^*)$*;*
(c) is in CONP *for* $RE(\mathcal{S} - \{(+a)^*, (+w)^*\})$*; and,*
(d) is in PSPACE *for* $RE(\mathcal{S})$*.*

This result does not leave much room for tractable cases. Of course, inclusion is in PTIME for any class of regular expressions for which expressions can be transformed into DFAs in polynomial time. An easy example of such a class is $\mathrm{RE}(a, a^+)$. The following example has probably more importance in practice. Often, the same symbol occurs only a few times in a regular expression of a DTD. As a matter of fact, if we impose a fixed bound k on the number of such occurences, then the inclusion problem becomes tractable. For every k, let $\mathrm{RE}^{\leq k}$ denote the class of all regular expressions where every symbol can occur at most k times.

Theorem 5. *Inclusion for* $RE^{\leq k}$ *is in* PTIME.

It should be noted though that the upper bound for the running time is of the form n^k, therefore k should be very small to be really useful.

In the rest of this section, we prove Theorem 4(a). The proofs of parts (b) and (c) can be found in the full version of this paper [15].

We show that for both cases there is a PTIME reduction from VALIDITY of prepositional 3DNF formulas. Let $\Phi = C_1 \vee \cdots \vee C_k$ be a propositional formula in 3DNF using variables $\{x_1, \ldots, x_n\}$. In both cases, we are going to construct regular expressions R_1, R_2 such that $L(R_1) \subseteq L(R_2)$ if and only if Φ is valid, i.e., if Φ is true under all truth assignments. More specifically, we encode truth assignments for Φ by strings. The basic idea is to construct R_1 and R_2 such

that $L(R_1)$ contains all such strings and R_2 captures a string w if and only if it represents an assignment which makes Φ true.

Let $U = \#a\$a\$\cdots\$a\#$ (n occurrences of a), be a regular expression describing exactly one string u. We will construct a regular expression W such that the strings of $L(W)$ can be interpreted as truth assignments. More precisely, for each truth assignment A there is a string $w_A \in L(W)$ and for each string $w \in L(W)$ there is a truth assignment A_w. Then we set $R_1 = U^{k-1}WU^{k-1}$ and $R_2 = N^{k-1}F_1 \cdots F_k N^{k-1}$, where for N and the F_i the following conditions should hold:

(i) $\{\varepsilon, u\} \subseteq L(N)$.
(ii) If N^ℓ captures a string $u^\ell z$ or zu^ℓ, for some ℓ, then $z \in L(\#^*)$.
(iii) If A_w makes C_i true then $w \in L(F_i)$. If $w_A \in L(F_i)$ then A makes C_i true.
(iv) $u \in L(F_i)$.
(v) Each string in $L(F_i)$ begins and ends with # and has no other occurences of #.
(vi) Each string in $L(W)$ starts and ends with a # and has no other occurences of #. Further, $\#\# \notin L(W)$.

Assume the existence of such expressions W, N and F_i. We claim that $L(R_1) \subseteq L(R_2)$ if and only if Φ is valid. To prove this claim let first $L(R_1) \subseteq L(R_2)$. Let A be an arbitrary truth assignment for Φ and let $v = u^{k-1}w_A u^{k-1}$. By assumption, as $v \in L(R_1)$ we get $v \in L(R_2)$. By (ii) and (vi), and the form of R_2, w_A can not be captured by a part of the leading or trailing N^{k-1}. By (v) and (vi), it follows that w_A must be covered by some F_i. Hence, C_i is made true by A.

For the other direction, suppose now that Φ is valid. Take an arbitrary $v = u^{k-1}wu^{k-1} \in L(R_1)$ and let A be the truth assigment A_w corresponding to w. Let C_i be a clause of Φ that becomes true under A_w. Due to (iii), we have that $w \in L(F_i)$. Further, as u is captured by each F_i and by N and as $L(N)$ also contains the empty string $u^{k-1}wu^{k-1} \in L(R_2)$. This completes the proof of the claim.

It remains to construct regular expressions W, N and F_i with the required properties. First of all, for both fragments it is easy to define N. For the first fragment, we can take $N = \#^*a^*\$^*a^*\$^*\cdots\$^*a^*\#^*$ (n occurrences of a), for the second, we take '?' in place of '*'.

The construction of the other expressions differs slightly for both cases. Let the expressions r_0, r_1, r_{01} be defined as follows:

(1) $r_0 = b^*a^*$, $r_1 = aa^*b^*a^*$ and $r_{01} = a^*b^*a^*$, in the case of $RE(a, a^*)$, and as
(2) $r_0 = a?$, $r_1 = aa?$ and $r_{01} = a?a?$ for $RE(a, a?)$.

In both cases, these expressions have the properties (INC1)-(INC4) below.

$$a \in L(r_0) \cap L(r_1) \qquad \text{(INC1)}$$
$$L(r_{01}) \subseteq L(r_0) \cup L(r_1) \qquad \text{(INC2)}$$
$$L(r_{01}) - L(r_0) \neq \emptyset \qquad \text{(INC3)}$$
$$L(r_{01}) - L(r_1) \neq \emptyset \qquad \text{(INC4)}$$

Now we define $W = \#r_{01}\$\cdots\$r_{01}\#$ (n occurrences of r_{01}). With $w = \#w_1\$\cdots\$w_n\# \in L(W)$ we associate a truth assignment A_w as follows: $A_w(x_j) :=$ true, if $w_j \in L(r_1)$; and false otherwise.

Let $z_0 \in L(r_{01}) - L(r_1)$ and $z_1 \in L(r_{01}) - L(r_0)$. They exist by (INC3) and (INC4). For a truth assignment A, let w_A be the string $\#y_1\$\cdots\$y_n\#$, where $y_j = z_1$ if $A(x_j) =$ true and $y_j = z_0$, otherwise.

For each i, we set $F_i = \#e_1\$\cdots\$e_n\#$, where for each $j = 1, \ldots, n$, $e_j = r_0$, $e_j = r_1$ or $e_j = r_{01}$ if x_j occurs negated, unnegated or not at all in C_i. It is easy to show that condition (iii) holds. That the other conditions hold is obvious from the construction.

Remark 2. The CONP lower bounds even hold, if in $\mathrm{RE}(a, a^*)$, subexpressions of the form aa^* are forbidden. Indeed, with $r_1 = ac^*a^*b^*a^*$ in case (1), the reduction still holds. Further, in the case of $\mathrm{RE}(a, a?)$ two letters suffice: take b and bb in place of \$ and #, respectively.

5 Equivalence of Regular Expressions

In the present section, we merely initiate the research on the equivalence of simple regular expressions. Of course, upper bounds for inclusion induce upper bounds for equivalence, but testing equivalence can be simpler. We show that the problem is in PTIME for $RE(a, a?)$ and $RE(a, a^*)$ by showing that such expressions are equivalent if and only if they have the same normal form (defined below). We conjecture that equivalence remains tractable for larger fragments or even the full fragment of simple regular expressions.

Theorem 6. *The equivalence problem is in* PTIME *for* $RE(a, a?)$, *and* $RE(a, a^*)$.

Proof (Sketch). We start by introducing a normal form for $\mathrm{RE}(a, a?, a^*)$ expressions. It is often useful to cumulate successive factors of a regular expression that have the same base symbol (or disjunction of base symbols). For such a sequence we only need to know the minimal and maximal number of occurences of the basic expression. By $e[i, j]$ we denote a sequence with at least i and at most j occurences of e. Here, $j = *$ stands for an unlimited number of repetitions. To this end, we write $e[1, 1]$ for e, $e[0, 1]$ for $e?$, and $e[0, *]$ for e^*. By combining successive factors $e[i_1, j_1]$ and $e[i_2, j_2]$ into $e[i_1 + i_2, j_2 + j_2]$ whenever possible, we arrive at the *sequence normal form* of a simple regular expression. E.g., the sequence normal form of $aa?aa?b^*bb?b^*$ is $a[2, 4]b[1, *]$.

For the sake of the proof, this normal form is not sufficient, as e.g., the regular expressions $r_1(a, b) = a[i, *]b[0, *]a[0, *]b[1, *]a[l, *]$ and $r_2(a, b) = a[i, *]b[1, *]a[0, *]b[0, *]a[l, *]$ are equivalent but have different normal forms. Whenever an expression of the form $r_1(a, b)$ occurs it can be replaced by $r_2(a, b)$. The *strong sequence normal form* of an expression r is the expression r' obtained by applying this rule as often as possible. It is easy to see, that r' is unique. For both fragments, it can be shown that two expressions are equivalent if and only if they have the same strong sequence normal form. □

6 Intersection of Regular Expressions

For arbitrary regular expressions, the intersection problem is PSPACE-complete. We show that the problem is NP-hard for the same two innocent fragments $\mathrm{RE}(a, a^*)$ and $\mathrm{RE}(a, a?)$ already studied in Section 4. By $\mathrm{RE}(\mathcal{S} - \{(+w)^*\})$, we denote the fragment of $\mathrm{RE}(\mathcal{S})$ where no factor can be of the form $(+w)^*$; so, factors of the form $(+a)^*$ are allowed. For the latter fragment, we obtain a matching NP-upper bound. The complexity of the full fragment $\mathrm{RE}(\mathcal{S})$ remains open. Our results are summarized in the following theorem:

Theorem 7. *The intersection problem is (a)* NP*-hard for* $RE(a, a^*)$ *and* $RE(a, a?)$*; and (b) in* NP *for* $RE(\mathcal{S} - (+w)^*)$.

Note that Theorem 7(a) makes the difference between the inclusion and intersection problem apparent. Indeed, inclusion of $\mathrm{RE}(\mathcal{S} - \{(+w)^*\})$ is PSPACE-complete, while it is NP-complete for intersection. The latter, however, does not imply that inclusion is always harder than intersection. Indeed, we obtained that for any fixed k inclusion of $\mathrm{RE}^{\leq k}$ is in PTIME. The next theorem shows that for $k = 3$, intersection is PSPACE-hard.

Content models of DTDs are restricted to one-unambiguous regular expressions. For a concrete definition, we refer to [4]. Although inclusion for such expressions is in PTIME, we obtain that the intersection problem is as hard as for arbitrary regular expressions.

Theorem 8. *The intersection problem is* PSPACE*-hard for (a) one-unambiguous regular expressions; and for (b)* $RE^{\leq 3}$.

A tractable fragment is the following:

Theorem 9. *The intersection problem is in* PTIME *for* $RE(a, a^+)$.

7 Conclusion

We revisited the inclusion, equivalence and intersection problem for regular expressions that are common in practical DTDs. Moreover, we showed that for all these problems, the complexities carry over to the corresponding problems for DTDs. For inclusion and equivalence, the complexity bounds for regular expressions also carry over to single-type SDTDs. We left the following complexities open: *(i)* equivalence for $\mathrm{RE}(\mathcal{S})$ or any fragment extending $\mathrm{RE}(a, a^*)$ or $\mathrm{RE}(a, a?)$; and, *(ii)* intersection for $\mathrm{RE}(\mathcal{S})$.

References

1. P. A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy FIFO channels. In *Proc. of CAV 1998,* pages 305–318, 1998.
2. G. J. Bex, F. Neven, and J. Van den Bussche. DTDs versus XML Schema: A practical study. To be presented at WebDB 2004.

3. A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets. Technical Report HKUST-TCSC-2001-0, The Hongkong University of Science and Technology, 2001.
4. A. Brüggemann-Klein and D. Wood. One-unambiguous regular languages. *Information and Computation,* 142(2): 182–206, 1998.
5. A. Brüggemann-Klein and D. Wood. Caterpillars: A context specification technique. *Markup Languages,* 2(1):81–106, 2000.
6. D. Calvanese, De G. Giacomo, M. Lenzerini, and M. Y. Vardi. Reasoning on regular path queries. *SIGMOD Record,* 32(4):83–92, 2003.
7. B. Choi. What are real DTDs like? In *WebDB 2002,* pages 43–48, 2002.
8. World Wide Web Consortium. Extensible Markup Language (XML). http://www.w3.org/XML.
9. World Wide Web Consortium. XML Schema. http://www.w3.org/XML/Schema.
10. L. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion,* Springer, 2002.
11. H. Hosoya and B. C. Pierce. XDuce: A statically typed XML processing language. *ACM Transactions on Internet Technology (TOIT),* 3(2):117–148, 2003.
12. Harry B. Hunt III, Daniel J. Rosenkrantz, and Thomas G. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *Journal of Computer and System Sciences,* 12(2):222–268, 1976.
13. D. Kozen. Lower bounds for natural proof systems. In *Proc. FOCS 1977,* pages 254–266. IEEE, 1977.
14. W. Martens and F. Neven. Typechecking top-down uniform unranked tree transducers. In *Proc. ICDT 2003,* pages 64–78, 2003.
15. W. Martens, F. Neven, and T. Schwentick. Complexity of decision problems for simple regular expressions: Full version. `http://alpha.luc.ac.be/lucp1436/pubs.html`.
16. M. Marx. XPath with conditional axis relations. In *Proc. EDBT 2004,* pages 477–494, 2004.
17. T. Milo and D. Suciu. Index structures for path expressions. In *Proc. ICDT 1999,* pages 277–295. 1999.
18. T. Milo, D. Suciu, and V. Vianu. Typechecking for XML transformers. *Journal of Computer and System Sciences,* 66(1):66–97, 2003.
19. M. Murata, D. Lee, and M. Mani. Taxonomy of XML schema languages using formal language theory. In *Extreme Markup Languages,* Montreal, Canada, 2001.
20. F. Neven. Automata, logic, and XML. In *Proc. CSL 2002,* pages 2–26. 2002.
21. Y. Papakonstantinou and V. Vianu. DTD inference for views of XML data. In *Proc. PODS 2000,* pages 35–46. ACM Press, 2000.
22. H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal on Computing,* 19(3):424–437, 1990.
23. H. Seidl. Haskell overloading is DEXPTIME-complete. *Information Processing Letters,* 52(2):57–60, 1994.
24. L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proc. STOC 1973,* pages 1–9, 1973.
25. E. van der Vlist. *Relax NG.* O'Reilly, 2003.
26. V. Vianu. A web odyssey: From Codd to XML. In *Proc. PODS 2001,* pages 1–15, 2001.

Author Index

Ambos-Spies, Klaus 711
André, Yves 863

Bab, Sebastian 404
Baumann, Nadine 599
Béal, Marie-Pierre 611
Beaudry, Martin 851
Bernardi, Vincent 416
Beygelzimer, Alina 346
Biedl, Therese 735, 748
Björklund, Henrik 673
Bodlaender, Hans L. 180
Böckenhauer, Hans-Joachim 227
Böhler, Elmar 392
Bonelli, Eduardo 119
Bongartz, Dirk 227
Boros, Endre 298
Bouchou, Béatrice 876
Brand, Matthew 106
Broersma, Hajo 204
Busse, Edgar 711

Caragiannis, Ioannis 215
Caron, Anne-Cécile 863
Chen, Hubie 239
Chen, Jianer 500
Chlebík, Miroslav 263
Chlebíková, Janka 263
Compagnoni, Adriana 119
Crasmaru, Marcel 660
Crescenzo, Giovanni Di 356

Debarbieux, Denis 863
Demaine, Erik D. 191
Dezani-Ciancaglini, Mariangiola 119
Doerr, Benjamin 760
Domaratzki, Michael 723
Downey, Rod 42
Duarte, Denio 876
Durand, Bruno 416

Elbassioni, Khaled 298
Elsässer, Robert 586
Epstein, Leah 562
Ésik, Zoltán 648

Fernandez, José M. 851
Fernau, Henning 488
Fiorenzi, Francesca 611
Fishkin, Aleksei 215
Fomin, Fedor V. 180, 204
Formenti, Enrico 416
Fraigniaud, Pierre 451
Frisken, Sarah 106

Gairing, Martin 574
Garralda, Pablo 119
Gimbert, Hugo 686
Glabbeek, Rob van 550
Glaßer, Christian 392, 660
Gronemeier, Andre 251
Gurvich, Vladimir 298

Hajiaghayi, MohammadTaghi 191
Halfeld Ferrari Alves, Mírian 876
Halldórsson, Magnús M. 25
Hansen, Kristoffer Arnsfelt 334
Hasan, Masud 735
Hebbinghaus, Nils 760
Hemaspaandra, Edith 287, 368
Hemaspaandra, Lane A. 287, 368
Hempel, Harald 368
Hitchcock, John M. 476
Hliněný, Petr 772
Holzer, Markus 851
Huang, Xiuzhen 500

Ilcinkas, David 451
Ilie, Lucian 818

Juedes, David 488

Kaklamanis, Christos 215
Kanj, Iyad A. 500
Kari, Jarkko 416
Kawachi, Akinori 839
Khachiyan, Leonid 298
Kobayashi, Hirotada 839
Köhler, Ekkehard 599
Kortsarz, Guy 25
Koshiba, Takeshi 839
Koutsoupias, Elias 177

Král', Daniel 274
Kuich, Werner 648

Lanotte, Ruggero 525
Laurent, Dominique 876
Lee, Troy 463
Lesh, Neal 106
Lisitsa, Alexei 623
López-Ortiz, Alejandro 735
López-Valdés, María 476
Lorenz, Ulf 586
Loyer, Yann 513
Lubiw, Anna 748
Lücking, Thomas 574
Ly, Olivier 537

Maggiolo-Schettini, Andrea 525
Makino, Kazuhisa 298
Marks, Joe 106
Martens, Wim 889
Mavronicolas, Marios 574
Mayordomo, Elvira 476
Miltersen, Peter Bro 334
Monien, Burkhard 574
Musicante, Martin A. 876

Nejedlý, Pavel 274
Neven, Frank 889
Nickelsen, Arfst 404
Niedermeier, Rolf 84
Nikovski, Daniel 106
Nishimura, Harumichi 827
Nordh, Gustav 380
Nutov, Zeev 310

Ochem, Pascal 818
Ogihara, Mitsunori 346
Okhotin, Alexander 698

Pál, Martin 239
Papaioannou, Evi 215
Peer, Guy 451
Pelc, Andrzej 451
Peleg, David 451
Peron, Adriano 525
Perrin, Dominique 611
Perry, Ron 106
Persiano, Giuseppe 356
Petre, Elena 807
Plotkin, Gordon 550
Polák, Libor 635
Potapov, Igor 623
Poupet, Victor 439
Putra, Raymond H. 839

Radziszowski, Stanisław 287
Regan, Kenneth W. 660
Romashchenko, Andrei 463
Roos, Yves 863
Rudnicki, Ryszard 1

Salomaa, Kai 723
Sandberg, Sven 673
Santis, Alfredo De 356
Sauerwald, Thomas 586
Schwarz, Bernhard 392
Schwentick, Thomas 889
Selivanov, Victor L. 783
Sengupta, Samik 660
Shallit, Jeffrey 818
Spriggs, Michael J. 748
Straccia, Umberto 513

Tassa, Tamir 562
Theyssier, Guillaume 427
Thilikos, Dimitrios M. 191
Tison, Sophie 863
Tiuryn, Jerzy 1
Travers, Stephen D. 322
Tripathi, Rahul 287

Vishkin, Uzi 104
Vorobyov, Sergei 673

Wagner, Klaus 392, 783
Weil, Pascal 149
Welzl, Emo 176
Werth, Sören 760
Wich, Klaus 794
Woeginger, Gerhard J. 204
Wójtowicz, Damian 1

Xia, Ge 500

Yamakami, Tomoyuki 827
Yedidia, Jonathan 106
Yuster, Raphael 310

Zielonka, Wiesław 686

This page intentionally left blank

This page intentionally left blank

Lecture Notes in Computer Science

For information about Vols. 1–3053

please contact your bookseller or Springer-Verlag

Vol. 3172: M. Dorigo, M. Birattari, C. Blum, L. M.Gambardella, F. Mondada, T. Stützle (Eds.), Ant Colony, Optimization and Swarm Intelligence. XII, 434 pages. 2004.

Vol. 3158: I. Nikolaidis, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. IX, 344 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004. (Subseries LNAI).

Vol. 3156: M. Joye, J.-J. Quisquater (Eds.), Cryptographic Hardware and Embedded Systems -CHES 2004. XIII, 455 pages. 2004.

Vol. 3153: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), Mathematical Foundations of Computer Science 2004. XIV, 902 pages. 2004.

Vol. 3152: M. Franklin (Ed.), Advances in Cryptology – CRYPTO 2004. XI, 579 pages. 2004.

Vol. 3148: R. Giacobazzi (Ed.), Static Analysis. X, 393 pages. 2004.

Vol. 3146: P. Érdi, A. Esposito, M. Marinaro, S. Scarpetta (Eds.), Computational Neuroscience: Cortical Dynamics. XI, 161 pages. 2004.

Vol. 3144: M. Papatriantafilou, P. Hunel (Eds.), Principles of Distributed Systems. XI, 246 pages. 2004.

Vol. 3143: W. Liu, Y. Shi, Q. Li (Eds.), Advances in Web-Based Learning – ICWL 2004. XIV, 459 pages. 2004.

Vol. 3142: J. Diaz, J. Karhumäki, A. Lepistö, D. Sannella (Eds.), Automata, Languages and Programming. XIX, 1253 pages. 2004.

Vol. 3140: N. Koch, P. Fraternali, M. Wirsing (Eds.), Web Engineering. XXI, 623 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004. (Subseries LNAI).

Vol. 3138: A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, D.d. Ridder (Eds.), Structural, Syntactic, and Statistical Pattern Recognition. XXII, 1168 pages. 2004.

Vol. 3134: C. Zannier, H. Erdogmus, L. Lindstrom (Eds.), Extreme Programming and Agile Methods - XP/Agile Universe 2004. XIV, 233 pages. 2004.

Vol. 3133: A.D. Pimentel, S. Vassiliadis (Eds.), Computer Systems: Architectures, Modeling, and Simulation. XIII, 562 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004. (Subseries LNAI).

Vol. 3130: A. Syropoulos, K. Berry, Y. Haralambous, B. Hughes, S. Peter, J. Plaice (Eds.), TEX, XML, and Digital Typography. VIII, 265 pages. 2004.

Vol. 3129: Q. Li, G. Wang, L. Feng (Eds.), Advances in Web-Age Information Management. XVII, 753 pages. 2004.

Vol. 3128: D. Asonov (Ed.), Querying Databases Privately. IX, 115 pages. 2004.

Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004. (Subseries LNAI).

Vol. 3126: P. Dini, P. Lorenz, J.N.d. Souza (Eds.), Service Assurance with Partial and Intermittent Resources. XI, 312 pages. 2004.

Vol. 3125: D. Kozen (Ed.), Mathematics of Program Construction. X, 401 pages. 2004.

Vol. 3124: J.N. de Souza, P. Dini, P. Lorenz (Eds.), Telecommunications and Networking - ICT 2004. XXVI, 1390 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004. (Subseries LNAI).

Vol. 3121: S. Nikoletseas, J.D.P. Rolim (Eds.), Algorithmic Aspects of Wireless Sensor Networks. X, 201 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004. (Subseries LNAI).

Vol. 3118: K. Miesenberger, J. Klaus, W.Zagler, D. Burger (Eds.), Computer Helping People with Special Needs. XXIII, 1191 pages. 2004.

Vol. 3116: C. Rattray, S. Maharaj, C. Shankland (Eds.), Algebraic Methodology and Software Technology. XI, 569 pages. 2004.

Vol. 3115: P. Enser, Y. Kompatsiaris, N.E. O'Connor, A.F. Smeaton, A.W.M. Smeulders (Eds.), Image and Video Retrieval. XVII, 679 pages. 2004.

Vol. 3114: R. Alur, D.A. Peled (Eds.), Computer Aided Verification. XII, 536 pages. 2004.

Vol. 3113: J. Karhumäki, H. Maurer, G. Paun, G. Rozenberg (Eds.), Theory Is Forever. X, 283 pages. 2004.

Vol. 3112: H. Williams, L. MacKinnon (Eds.), Key Technologies for Data Management. XII, 265 pages. 2004.

Vol. 3111: T. Hagerup, J. Katajainen (Eds.), Algorithm Theory - SWAT 2004. XI, 506 pages. 2004.

Vol. 3110: A. Juels (Ed.), Financial Cryptography. XI, 281 pages. 2004.

Vol. 3109: S.C. Sahinalp, S. Muthukrishnan, U. Dogrusoz (Eds.), Combinatorial Pattern Matching. XII, 486 pages. 2004.

Vol. 3108: H. Wang, J. Pieprzyk, V. Varadharajan (Eds.), Information Security and Privacy. XII, 494 pages. 2004.

Vol. 3107: J. Bosch, C. Krueger (Eds.), Software Reuse: Methods, Techniques and Tools. XI, 339 pages. 2004.

Vol. 3105: S. Göbel, U. Spierling, A. Hoffmann, I.Iurgel, O. Schneider, J. Dechau, A. Feix (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. XVI, 304 pages. 2004.

Vol. 3104: R. Kralovic, O. Sykora (Eds.), Structural Information and Communication Complexity. X, 303 pages. 2004.

Vol. 3103: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. XLIX, 1439 pages. 2004.

Vol. 3102: K. Deb, e. al. (Eds.), Genetic and Evolutionary Computation – GECCO 2004. L, 1445 pages. 2004.

Vol. 3101: M. Masoodian, S. Jones, B. Rogers (Eds.), Computer Human Interaction. XIV, 694 pages. 2004.

Vol. 3100: J.F. Peters, A. Skowron, **J.W. Grzymała-Busse,** B. Kostek, **R.W. Świniarski,** M.S. Szczuka (Eds.), Transactions on Rough Sets I. X, 405 pages. 2004.

Vol. 3099: J. Cortadella, W. Reisig (Eds.), Applications and Theory of Petri Nets 2004. XI, 505 pages. 2004.

Vol. 3098: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets. VIII, 849 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004. (Subseries LNAI).

Vol. 3096: G. Melnik, H. Holz (Eds.), Advances in Learning Software Organizations. X, 173 pages. 2004.

Vol. 3095: C. Bussler, D. Fensel, M.E. Orlowska, J. Yang (Eds.), Web Services, E-Business, and the Semantic Web. X, 147 pages. 2004.

Vol. 3094: A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval. VIII, 229 pages. 2004.

Vol. 3093: S.K. Katsikas, S. Gritzalis, J. Lopez (Eds.), Public Key Infrastructure. XIII, 380 pages. 2004.

Vol. 3092: J. Eckstein, H. Baumeister (Eds.), Extreme Programming and Agile Processes in Software Engineering. XVI, 358 pages. 2004.

Vol. 3091: V. van Oostrom (Ed.), Rewriting Techniques and Applications. X, 313 pages. 2004.

Vol. 3089: M. Jakobsson, M. Yung, J. Zhou (Eds.), Applied Cryptography and Network Security. XIV, 510 pages. 2004.

Vol. 3087: D. Maltoni, A.K. Jain (Eds.), Biometric Authentication. XIII, 343 pages. 2004.

Vol. 3086: M. Odersky (Ed.), ECOOP 2004 – Object-Oriented Programming. XIII, 611 pages. 2004.

Vol. 3085: S. Berardi, M. Coppo, F. Damiani (Eds.),Types for Proofs and Programs. X, 409 pages. 2004.

Vol. 3084: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering. XIV, 596 pages. 2004.

Vol. 3083: W. Emmerich, A.L. Wolf (Eds.), Component Deployment. X, 249 pages. 2004.

Vol. 3080: J. Desel, B. Pernici, M. Weske (Eds.), Business Process Management. X, 307 pages. 2004.

Vol. 3079: Z. Mammeri, P. Lorenz (Eds.), High Speed Networks and Multimedia Communications. XVIII, 1103 pages. 2004.

Vol. 3078: S. Cotin, D.N. Metaxas (Eds.), Medical Simulation. XVI, 296 pages. 2004.

Vol. 3077: F. Roli, J. Kittler, T. Windeatt (Eds.), Multiple Classifier Systems. XII, 386 pages. 2004.

Vol. 3076: D. Buell (Ed.), Algorithmic Number Theory. XI, 451 pages. 2004.

Vol. 3074: B. Kuijpers, P. Revesz (Eds.), Constraint Databases and Applications. XII, 181 pages. 2004.

Vol. 3073: H. Chen, R. Moore, D.D. Zeng, J. Leavitt (Eds.), Intelligence and Security Informatics. XV, 536 pages. 2004.

Vol. 3072: D. Zhang, A.K. Jain (Eds.), Biometric Authentication. XVII, 800 pages. 2004.

Vol. 3071: A. Omicini, P. Petta, J. Pitt (Eds.), Engineering Societies in the Agents World. XIII, 409 pages. 2004. (Subseries LNAI).

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004. (Subseries LNAI).

Vol. 3068: E. André, L. Dybkjær, W. Minker, P. Heisterkamp (Eds.), Affective Dialogue Systems. XII, 324 pages. 2004. (Subseries LNAI).

Vol. 3067: M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Programming Multi-Agent Systems. X, 221 pages. 2004. (Subseries LNAI).

Vol. 3066: S.Tsumoto, **R. Słowiński,** J. Komorowski, **J.W. Grzymała-Busse** (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004. (Subseries LNAI).

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004. (Subseries LNAI).

Vol. 3064: D. Bienstock, G. Nemhauser (Eds.), Integer Programming and Combinatorial Optimization. XI, 445 pages. 2004.

Vol. 3063: A. Llamosí, A. Strohmeier (Eds.), Reliable Software Technologies - Ada-Europe 2004. XIII, 333 pages. 2004.

Vol. 3062: J.L. Pfaltz, M. Nagl, B. Böhlen (Eds.), Applications of Graph Transformations with Industrial Relevance. XV, 500 pages. 2004.

Vol. 3061: F.F. Ramos, H. Unger, V. Larios (Eds.), Advanced Distributed Systems. VIII, 285 pages. 2004.

Vol. 3060: A.Y.Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004. (Subseries LNAI).

Vol. 3059: C.C. Ribeiro, S.L. Martins (Eds.), Experimental and Efficient Algorithms. X, 586 pages. 2004.

Vol. 3058: N. Sebe, M.S. Lew, T.S. Huang (Eds.), Computer Vision in Human-Computer Interaction. X, 233 pages. 2004.

Vol. 3057: B. Jayaraman (Ed.), Practical Aspects of Declarative Languages. VIII, 255 pages. 2004.

Vol. 3056: H. Dai, R. Srikant, C. Zhang (Eds.), Advances in Knowledge Discovery and Data Mining. XIX, 713 pages. 2004. (Subseries LNAI).

Vol. 3055: H. Christiansen, M.-S. Hacid, T. Andreasen, H.L. Larsen (Eds.), Flexible Query Answering Systems. X, 500 pages. 2004. (Subseries LNAI).

Vol. 3054: I. Crnkovic, J.A. Stafford, H.W. Schmidt, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 311 pages. 2004.