Lecture Notes in Artificial Intelligence 5110

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

FoLLI Publications on Logic, Language and Information

Wilfrid Hodges Ruy de Queiroz (Eds.)

Logic, Language, Information and Computation

15th International Workshop, WoLLIC 2008
Edinburgh, UK, July 1-4, 2008
Proceedings

Springer

Volume Editors

Wilfrid Hodges
School of Mathematical Sciences, Queen Mary
University of London
London, UK
E-mail: w.hodges@qmul.ac.uk

Ruy de Queiroz
Centro de Informática
Universidade Federal de Pernambuco
Recife, PE, Brasil
E-mail: ruy@cin.ufpe.br

Library of Congress Control Number: 2008929581

CR Subject Classification (1998): I.2, F.1, F.2.1-2, F.4.1, G.1.0

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-69936-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-69936-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12434992 06/3180 5 4 3 2 1 0

Preface

The Workshop on Logic, Language, Information and Computation (WoLLIC) has met every year since 1994 with the aim of fostering interdisciplinary research in pure and applied logic. The idea is to have a forum which is large enough in the number of possible interactions between logic and the sciences related to information and computation, and yet is small enough to allow for concrete and useful interaction among participants.

This volume contains the texts of the 21 contributed papers selected for presentation at WoLLIC 2008. Between them they give a representative sample of some of the most active areas of research on the frontiers between computation, logic and linguistics. The authors range from well-established leaders in the field to researchers still working for their PhDs or masters degrees.

The volume also includes abstracts of talks by some of the seven invited speakers. Full texts will appear in a peer-reviewed issue of the *Journal of Computer and Systems Sciences.*

April 2008

Wilfrid Hodges
Ruy de Queiroz

Organization

WoLLIC 2008 Programme Committee

Lev Beklemishev (University of Utrecht)
Eli Ben-Sasson (Technion, Haifa)
Xavier Caicedo (University of Los Andes, Colombia)
Mary Dalrymple (University of Oxford)
Martin Escardo (University of Birmingham)
Wilfrid Hodges (Queen Mary, University of London, Chair)
Achim Jung (University of Birmingham)
Louis Kauffman (University of Illinois at Chicago)
Ulrich Kohlenbach (University of Darmstadt)
Leonid Libkin (Edinburgh University)
Giuseppe Longo (Ecole Normal Supérieure, Paris)
Michael Moortgat (University of Utrecht)
Valeria de Paiva (PARC, USA)
Andre Scedrov (University of Pennsylvania)
Valentin Shehtman (Inst. for Info. Transmission Problems, Moscow)
Joe Wells (Heriot-Watt University, Edinburgh)

Referees (In Addition to the Programme Committee)

Natasha Alechina, Jeremy Avigad, Pablo Barcelo, Guillaume Burel, Balder ten Cate, Nils Danielsson, Stefan Dantchev, Hans van Ditmarsch, Andreas Herzig, Robin Hirsch, Ian Hodkinson, Mark Jago, Elham Kashefi, Vladimir Krupski, Wilfried Meyer-Viol, Pavel Naumov, Henrik Nilsson, Graham Priest, Panu Raatikainen, Stephen Read, Mark Reynolds, Mikhail Rybakov, Yo Sato, Ronen Shaltiel, Alexander Shen, Kristian Støvring, Christoph Zengler

WoLLIC 2008 Organizing Committee

Mauricio Ayala-Rincon (University of Brasilia, Brazil)
Fairouz Kamareddine (Heriot-Watt University, Scotland, Co-chair)
Anjolina de Oliveira (Federal University of Pernambuco, Brazil)
Ruy de Queiroz (Federal University of Pernambuco, Brazil, Co-chair)

WoLLIC Steering Committee

S. Abramsky, J. van Benthem, J. Halpern, W. Hodges, D. Leivant, A. Macintyre, G. Mints, R. de Queiroz

Table of Contents

Tutorials and Invited Lectures

Contributed Papers

Inter-deriving Semantic Artifacts for Object-Oriented Programming
(Extended Abstract)

Olivier Danvy and Jacob Johannsen

Department of Computer Science, University of Aarhus
IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark
{danvy,cnn}@daimi.au.dk
http://www.daimi.au.dk/~{danvy,cnn}

Abstract. We present a new abstract machine for Abadi and Cardelli's untyped calculus of objects. What is special about this semantic artifact (i.e., man-made construct) is that is mechanically corresponds to both the reduction semantics (i.e., small-step operational semantics) and the natural semantics (i.e., big-step operational semantics) specified in Abadi and Cardelli's monograph. This abstract machine therefore embodies the soundness of Abadi and Cardelli's reduction semantics and natural semantics relative to each other.

To move closer to actual implementations, which use environments rather than actual substitutions, we then represent object methods as closures and in the same inter-derivational spirit, we present three new semantic artifacts: a reduction semantics for a version of Abadi and Cardelli's untyped calculus of objects with explicit substitutions, an environment-based abstract machine, and a natural semantics (i.e., an interpreter) with environments. These three new semantic artifacts mechanically correspond to each other, and furthermore, they are coherent with the previous ones since as we show, the two abstract machines are bisimilar. Overall, though, the significance of these artifacts lies in them not having been designed from scratch and then proved correct: instead, they were mechanically inter-derived.

1 Introduction

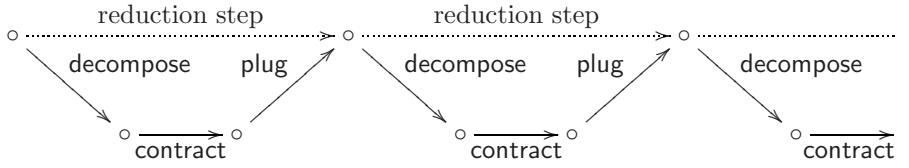
Our goal here is to apply Danvy et al.'s 'syntactic correspondence' and 'functional correspondence' [3, 8, 12, 21, 37, 38, 39], which were developed for the λ-calculus with effects, to Abadi and Cardelli's untyped calculus of objects [1, Chapter 6].

1.1 Background and First Contribution

The syntactic correspondence between reduction semantics and abstract machines: This correspondence mechanically links a reduction semantics (i.e., a small-step operational semantics with an explicit representation of reduction contexts [27, 28]) to an abstract machine. In such a reduction semantics, evaluation

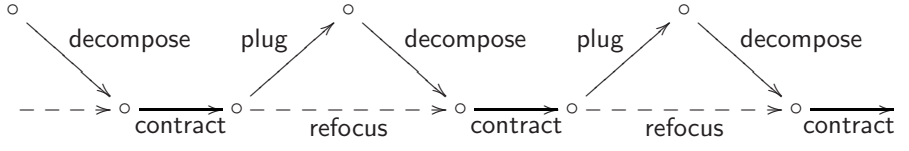
W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 1–16, 2008.

is implemented by iterated reduction, and the corresponding reduction sequence can be depicted as follows:



At each step, a non-value term is decomposed into a reduction context and a potential redex. If the potential redex is an actual one (i.e., if it is not stuck), it is contracted. The contractum is then plugged into the context, yielding the next term in the reduction sequence.

At each step, the function plug therefore constructs an intermediate term. In the course of evaluation, this term is then immediately decomposed by the subsequent call to decompose. The composition of plug and decompose can thus be replaced by a more efficient function, refocus, that directly goes from redex site to redex site in the reduction sequence:



As shown by Danvy and Nielsen [25], refocus can take the form of a state-transition function. Therefore, together with contract, the result is an abstract machine. And what is remarkable here is that the abstract machines obtained by refocusing are not unnatural ones.

In fact, this syntactic correspondence between reduction semantics and abstract machines has made it possible to obtain a variety of abstract machines for the λ-calculus, be it pure or with effects. Some of these machines were independently known and some others are new [10, 11]. Symmetrically, it also has made it possible to exhibit the calculi and the reduction strategies (in the form of reduction semantics) corresponding to pre-existing abstract machines.

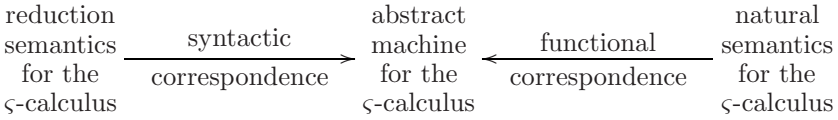
The functional correspondence between natural semantics and abstract machines: This correspondence mechanically links a natural semantics (i.e., a big-step operational semantics, as implemented by an interpreter [32, 40]) to an abstract machine. It is based on the framework initiated by Reynolds in his seminal article "Definitional Interpreters for Higher-Order Programming Languages" [41]. In a nutshell, successively transforming an interpreter using closure conversion, transformation into continuation-passing style (CPS), and defunctionalization yields an abstract machine [4]. And what is remarkable here is that the abstract machines obtained by CPS transformation and defunctionalization are not unnnatural ones.

In fact, this functional correspondence between natural semantics and abstract machines has made it possible to obtain a variety of abstract machines for the λ-calculus, be it pure or with effects. Some of these machines were independently known and some others are new [5, 6, 9]. Symmetrically, it also has made it possible to exhibit the interpreter (in the form of a natural semantics) corresponding to pre-existing abstract machines.

Our starting point here: Together, the syntactic and the functional correspondences make it possible to connect three semantic artifacts (i.e., man-made constructs) soundly: reduction semantics, abstract machines, and natural semantics. Better: the correspondence make it possible to *inter-derive* these semantics (or more precisely, their representation as functional programs), mechanically. This inter-derivation contrasts with defining several semantics, which requires work, and proving their soundness relative to each other, which requires more work. As Rod Burstall soberly put it once, "theory should be call by need." Our goal here is to apply these two correspondences to Abadi and Cardelli's untyped calculus of objects.

Abadi and Cardelli's untyped calculus of objects: Abadi and Cardelli's monograph "A Theory of Objects" is a landmark. Nowadays it provides standard course material about object-oriented languages and programming. Of interest to us here is its Chapter 6 where an untyped calculus of objects, the ς-calculus, is developed in the same spirit as its predecessor, the λ-calculus [7, 14], which was initially developed as an untyped calculus of functions. The ς-calculus is specified with a reduction semantics, for a given reduction order, and with a natural semantics, for a given evaluation order. A soundness theorem (Proposition 6.2-3, page 64) links the two semantics. Operational reduction is also shown to be complete with respect to many-step reduction with a completeness theorem (Theorem 6.2-4, page 65). Soundness matters because it shows that the interpreter implementing the natural semantics is faithful to the reduction semantics and vice versa. Completeness matters because it shows that the reductions may be meaningfully re-ordered, thus enabling practical optimizations such as constant propagation and more generally partial evaluation [15, 31].

First contribution: Using the syntactic correspondence, we exhibit an abstract machine that embodies the reduction semantics of the ς-calculus and its reduction strategy. Using the functional correspondence, we exhibit an abstract machine that embodies the natural semantics of the ς-calculus and its evaluation strategy. The two abstract machines are identical. This abstract machine, which is new, therefore mediates between the reduction semantics and the natural semantics, and practically confirms the soundness theorem:

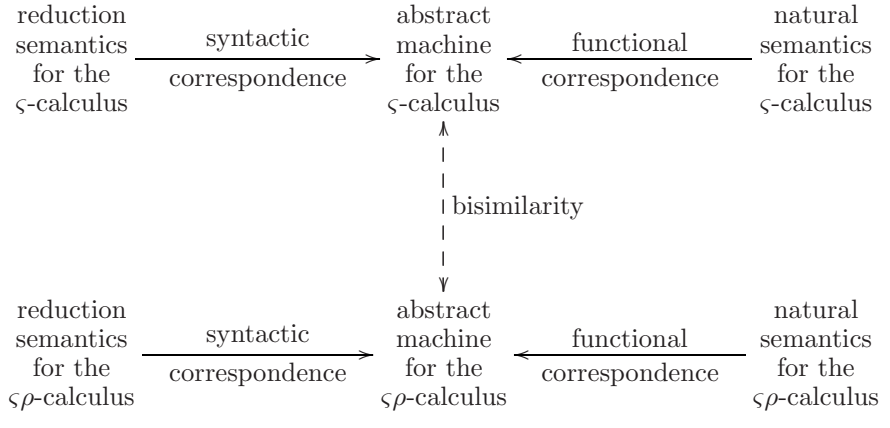


1.2 Further Background and Contributions

Substitutions vs. environments: Practical implementations of the λ-calculus do not use actual substitutions. Instead, they use 'environments,' which are mappings representing delayed substitutions, and represent functions with 'closures,' which are pairs of terms and environments [34]. In such practical implementations, an identifier is not a thing to be substituted by a term, but a thing to be looked up in the current environment. At the turn of the 1990's [17], Curien proposed a 'calculus of closures,' the $\lambda\rho$-calculus, to account for this implementation strategy of the λ-calculus, and explicit substitutions were born [2, 18, 43]. Both the syntactic and the functional correspondences have been applied to calculi of explicit substitutions, environment-based abstract machines, and natural semantics using environments [4, 10].

Abadi and Cardelli's untyped calculus of objects with methods as closures: We present a version of the ς-calculus with explicit substitutions, the $\varsigma\rho$-calculus. Instead of performing substitution when invoking a method, we represent methods as closures. We state three semantic artifacts for the $\varsigma\rho$-calculus: a natural semantics, an abstract machine, and a reduction semantics.

Contributions: Using the syntactic correspondence, we exhibit an environment-based abstract machine that embodies the reduction semantics of the $\varsigma\rho$-calculus and its reduction strategy. Using the functional correspondence, we exhibit an environment-based abstract machine that embodies the natural semantics of the $\varsigma\rho$-calculus and its evaluation strategy. Again, the two abstract machines are identical, which establishes the soundness of the reduction semantics and of the natural semantics for the $\varsigma\rho$-calculus relative to each other. We then show that this environment-based abstract machine and the abstract machine with actual substitutions from Section 1.1 are bisimilar, which establishes the coherence of the $\varsigma\rho$-calculus with respect to the ς-calculus:



As for having a completeness theorem for the $\varsigma\rho$-calculus, Melliès's proof applies mutatis mutandis [1, Theorem 6.2-4, page 65].

1.3 Overview

In Section 2, we remind the reader of the ς-calculus (Section 2.1) and we present its reduction semantics; through the syntactic correspondence, we obtain the corresponding abstract machine (Section 2.2). Through the functional correspondence, we then present the natural semantics corresponding to this abstract machine (Section 2.3). This natural semantics coincides with Abadi and Cardelli's. In Section 3, we introduce the ςρ-calculus, which is a version of the ς-calculus with explicit substitutions where methods are represented with closures, and we specify it with a natural semantics that uses environments (Section 3.1); through the functional correspondence, we obtain the corresponding abstract machine (Section 3.2). Through the syntactic correspondence, we then present the reduction semantics corresponding to this abstract machine (Section 3.3). In Section 4.1, we present a mapping from ςρ-closures to ς-terms that performs the actual substitutions that were delayed by the given environments in the given terms. In Section 4.2, using this mapping, we show that the two abstract machines are bisimilar, which establishes a coherence between the three semantic artifacts for the ς-calculus and the three semantic artifacts for the ςρ-calculus. We then review related work in Section 5 and conclude in Section 6.

Prerequisites: We assume the reader to be mildly familiar with Sections 6.1 and 6.2 of Abadi and Cardelli's monograph [1] and with the concepts of reduction semantics (BNF of terms and of reduction contexts, notion of redex, one-step reduction, evaluation as iterated reduction), of abstract machines (initial, intermediate, and final states, and state-transition functions), of natural semantics (interpreters as evaluation functions), and of bisimulation. As for the syntactic and functional correspondences, the unfamiliar reader can just flip through Danvy's invited paper at WRS'04 [20] or through Danvy and Millikin's recent note about small-step and big-step abstract machines [23] for what is not self-explanatory.

2 Abadi and Cardelli's Untyped Calculus of Objects: The ς-Calculus

We consider in turn a reduction semantics for the ς-calculus (Section 2.1), the corresponding abstract machine (Section 2.2), and the corresponding natural semantics (Section 2.3).

2.1 A Reduction Semantics

BNF of terms and of values: An object is a collection of named attributes. Names are labels and all labels are distinct within each object. All attributes are methods with a bound variable representing self (and to be bound to the host object at invocation time) and a body whose execution yields a result.

$$\begin{array}{ll} (\mathsf{Term}) & t ::= x \mid [l = \varsigma(x)t,\ \ldots,\ l = \varsigma(x)t] \mid t.l \mid t.l \Leftarrow \varsigma(x)t \\ (\mathsf{Value}) & v ::= [l = \varsigma(x)t,\ \ldots,\ l = \varsigma(x)t] \end{array}$$

This grammar for terms defines the same language as in Abadi and Cardelli's book but it uses a more uniform naming convention.

NB: Occasionally, we index a value by its number of methods, as in $v^n = [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}]$.

Notion of redex: Methods can be invoked or updated [1, Definition 6.2-1 (1)]. Here is the grammar of potential redexes:

$$pr ::= v.l \mid v.l \Leftarrow \varsigma(x)t$$

The contraction rules read as follows:

$$\begin{array}{rl} v^n.l_j \rightarrowtail & t_j\{v^n/x_j\} \\ & \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}] \\ v^n.l_j \Leftarrow \varsigma(x)t \rightarrowtail & [l_j = \varsigma(x)t,\ l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}\setminus\{j\}}] \\ & \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}] \end{array}$$

A potential redex is an actual one when its side conditions are satisfied, and contraction can take place. Otherwise, the potential redex is stuck.

BNF of reduction contexts: The following grammar for reduction contexts does not occur in Abadi and Cardelli's book but it plausibly reflects the 'evaluation strategy of the sort commonly used in programming languages' [1, Section 6.2.4, page 63]:

$$(\mathsf{Context}) \qquad C ::= [\,] \mid C[[\,].l] \mid C[[\,].l \Leftarrow \varsigma(x)t]$$

Lemma 1 (Unique decomposition). *Any term which is not a value can be uniquely decomposed into a reduction context and a potential redex.*

One is then in position to define a decomposition function mapping a term to either a value or to a reduction context and a potential redex, a contraction function mapping an actual redex to its contractum, and a plug function mapping a reduction context and a term to a term. Thus equipped, one can define a one-step reduction function (noted $\rightarrow$ below) and then an evaluation function as the iteration of the one-step reduction function (noted $\rightarrow^*$ below). We have implemented and copiously tested this reduction semantics (as well as all the other semantic artifacts of this article) in Standard ML.

2.2 The Corresponding Abstract Machine

Applying the syntactic correspondence (i.e., calculating the refocus function) yields the following eval/apply abstract machine [36]:

$$
\begin{array}{rcl}
\langle v,\, C\rangle & \Rightarrow_S & \langle C,\, v\rangle \\
\langle t.l,\, C\rangle & \Rightarrow_S & \langle t,\, C[[\,].l]\rangle \\
\langle t.l \Leftarrow \varsigma(x)t',\, C\rangle & \Rightarrow_S & \langle t,\, C[[\,].l \Leftarrow \varsigma(x)t']\rangle
\end{array}
$$

$$
\begin{array}{rcl}
\langle [\,],\, v\ \rangle & \Rightarrow_S & v \\
\langle C[[\,].l_j],\, v^n\rangle & \Rightarrow_S & \langle t_j\{v^n/x_j\},\, C\rangle \\
 & & \quad \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}] \\
\langle C[[\,].l_j \Leftarrow \varsigma(x)t],\, v^n\rangle & \Rightarrow_S & \langle C,\, [l_j = \varsigma(x)t, l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}\setminus\{j\}}]\rangle \\
 & & \quad \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}]
\end{array}
$$

This machine evaluates a closed term t by starting in the configuration $\langle t, [\,]\rangle$ and by iterating $\Rightarrow_S$ (noted $\Rightarrow_S^*$ below). It halts with a value v if it reaches a configuration $\langle [\,], v\rangle$ It becomes stuck if it reaches either of the configurations $\langle C[[\,].l], v\rangle$ or $\langle C[[\,].l \Leftarrow \varsigma(x)t], v\rangle$ and v does not contain a method with the label l.

The following proposition is a corollary of the soundness of refocusing:

Proposition 1 (Full correctness). *For any closed term t, $t \to^* v$ if and only if $\langle t, [\,]\rangle \Rightarrow_S^* v$.*

2.3 The Corresponding Natural Semantics

In Section 2.2, the function implementing the abstract machine is in defunctionalized form [24]. Refunctionalizing it [22] yields an evaluation function in continuation-passing style (CPS). Writing this evaluation function in direct style [19] yields an evaluation function that implements the following natural semantics:

$$
(\mathrm{INV}_\varsigma) \qquad \frac{\vdash t \rightsquigarrow v^n \qquad \vdash t_j\{v^n/x_j\} \rightsquigarrow v}{\vdash t.l_j \rightsquigarrow v} \qquad \begin{array}{l}\text{if } 1 \le j \le n, \text{ where} \\ v^n = [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}]\end{array}
$$

$$
(\mathrm{UPD}_\varsigma) \quad \frac{\vdash t \rightsquigarrow v^n}{\vdash t.l_j \Leftarrow \varsigma(x)t' \rightsquigarrow [l_j = \varsigma(x)t',\ l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}\setminus\{j\}}]} \quad \begin{array}{l}\text{if } 1 \le j \le n, \text{ where} \\ v^n = [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}]\end{array}
$$

This natural semantics coincides with Abadi and Cardelli's [1, Section 6.2.4, page 64].

The following proposition is a corollary of the soundness of the CPS transformation and of defunctionalization:

Proposition 2 (Full correctness). *For any closed term t, $\langle t, [\,]\rangle \Rightarrow_S^* v$ if and only if $\vdash t \rightsquigarrow v$.*

2.4 Summary and Conclusion

Using the syntactic correspondence and the functional correspondence, we have mechanically derived an abstract machine that mediates between Abadi and Cardelli's reduction semantics and natural semantics for the ς-calculus and the 'evaluation strategy of the sort commonly used in programming languages.' The two derivations confirm (1) the soundness of the two semantics relative to each other and (2) the BNF of the reduction contexts we put forward in Section 2.1. They also pave the way to using closures, which we do next.

3 Object Methods as Closures: the $\varsigma\rho$-Calculus

We consider in turn a natural semantics for the ς-calculus with environments (Section 3.1), the corresponding environment-based abstract machine (Section 3.2), and the corresponding reduction semantics (Section 3.3). The resulting calculus is one of explicit substitutions, the $\varsigma\rho$-calculus.

3.1 A Natural Semantics

Let us adapt the natural semantics of Section 2.3 to operate with environments. Three changes take place:

1. The category of values changes to objects where each method holds its own environment (noted 'e'):

$$(\mathsf{Value}) \qquad v ::= [l = (\varsigma(x)t)[e],\ \ldots,\ l = (\varsigma(x)t)[e]]$$

2. The environment is defined as an association list:

$$(\mathsf{Environment}) \qquad e ::= \bullet \mid (x,\, v) \cdot e$$

   and an auxiliary function *lookup* is used to look up an identifier in the current environment.
3. The evaluation judgment now reads as follows:

$$e \vdash t \rightsquigarrow v$$

Again, we occasionally index a value with the number of its methods.

The two rules from Section 2.3 are then straightforwardly adapted:

$$(\mathrm{INV}_{\varsigma\rho}) \quad \frac{e \vdash t \rightsquigarrow v^n \quad (x_j,\, v^n) \cdot e_j \vdash t_j \rightsquigarrow v}{e \vdash t.l_j \rightsquigarrow v} \quad \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = (\varsigma(x_i)t_i)[e_i]^{i \in \{1..n\}}]$$

$$(\mathrm{UPD}_{\varsigma\rho}) \quad \frac{e \vdash t \rightsquigarrow v^n}{e \vdash t.l_j \Leftarrow \varsigma(x)t' \rightsquigarrow v} \quad \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = (\varsigma(x_i)t_i)[e_i]^{i \in \{1..n\}}] \text{ and } v = [l_j = (\varsigma(x)t')[e],\ l_i = (\varsigma(x_i)t_i)[e_i]^{i \in \{1..n\} \setminus \{j\}}]$$

We also need the following rule to convert the methods of an object literal to method closures:

$$(\mathrm{CLO}_{\varsigma\rho})\quad \frac{}{e \vdash [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}] \rightsquigarrow [l_i = (\varsigma(x_i)t_i)[e]^{i\in\{1..n\}}]}$$

In addition, we need the following new rule to look up variables in the current environment:

$$(\mathrm{VAR\text{-}L}_{\varsigma\rho})\quad \frac{}{e \vdash x \rightsquigarrow v} \qquad \text{if } \mathit{lookup}\,(x,\, e) = v$$

Alternatively, and as done, e.g., in the Categorical Abstract Machine [16], one could use two rules to incrementally peel off the environment. For closed terms, x always occurs in e. For open terms, evaluation would become stuck here.

3.2 The Corresponding Abstract Machine

To apply the functional correspondence, we successively CPS-transform and defunctionalize the evaluation function implementing the natural semantics of Section 3.1. The grammar of evaluation contexts now reads as follows:

$$(\mathsf{Context}) \qquad C ::= [\,] \mid C[[\,].l] \mid C[[\,].l \Leftarrow (\varsigma(x)t)[e]]$$

All in all, the functional correspondence yields the following eval/apply abstract machine:

$$\begin{aligned}
\langle x,\, e,\, C\rangle &\Rightarrow_E \langle C,\, v\rangle \\
&\quad \text{if } \mathit{lookup}\,(x,\, e) = v \\
\langle [l_i = \varsigma(x_i)t_i{}^{i\in\{1..n\}}],\, e,\, C\rangle &\Rightarrow_E \langle C,\, [l_i = (\varsigma(x_i)t_i)[e]^{i\in\{1..n\}}]\rangle \\
\langle t.l,\, e,\, C\rangle &\Rightarrow_E \langle t,\, e,\, C[[\,].l]\rangle \\
\langle t.l \Leftarrow \varsigma(x)t',\, e,\, C\rangle &\Rightarrow_E \langle t,\, e,\, C[[\,].l \Leftarrow (\varsigma(x)t')[e]]\rangle
\end{aligned}$$

$$\begin{aligned}
\langle [\,],\, v\ \rangle &\Rightarrow_E v \\
\langle C[[\,].l_j],\, v^n\rangle &\Rightarrow_E \langle t_j,\, (x_j,\, v^n)\cdot e_j,\, C\rangle \\
&\quad \text{if } 1 \le j \le n, \text{ where} \\
&\quad v^n = [l_i = (\varsigma(x_i)t_i)[e_i]^{i\in\{1..n\}}] \\
\langle C[[\,].l_j \Leftarrow (\varsigma(x)t)[e]],\, v^n\rangle &\Rightarrow_E \langle C,\, [l_j = (\varsigma(x)t)[e], l_i = (\varsigma(x_i)t_i)[e_i]^{i\in\{1..n\}\setminus\{j\}}]\rangle \\
&\quad \text{if } 1 \le j \le n, \text{ where} \\
&\quad v^n = [l_i = (\varsigma(x_i)t_i)[e_i]^{i\in\{1..n\}}]
\end{aligned}$$

This machine evaluates a closed term t by starting in the configuration $\langle t,\, \bullet,\, [\,]\rangle$ and by iterating $\Rightarrow_E$ (noted $\Rightarrow_E^*$ below). It halts with a value v if it reaches a configuration $\langle [\,],\, v\rangle$. It becomes stuck if it reaches either of the configurations $\langle C[[\,].l],\, v\rangle$ or $\langle C[[\,].l \Leftarrow (\varsigma(x)t)[e]],\, v\rangle$ and v does not contain a method with the label l.

The following proposition is a corollary of the soundness of the CPS transformation and of defunctionalization:

Proposition 3 (Full correctness). *For any closed term t, $\bullet \vdash t \rightsquigarrow v$ if and only if $\langle t, \bullet, [\,] \rangle \Rightarrow_E^* v$.*

3.3 The Corresponding Reduction Semantics

BNF of terms, of values, and of closures: The BNF of terms does not change. The BNF of values is as in Section 3.1. In addition, as in Curien's $\lambda\rho$-calculus compared to the λ-calculus, a new syntactic category appears, that of closures:

$$\text{(Closure)} \qquad c ::= t[e] \mid [l = (\varsigma(x)t)[e], \ldots, l = (\varsigma(x)t)[e]] \mid c.l \mid c.l \Leftarrow (\varsigma(x)t)[e]$$

Notion of redex: The two original contraction rules are adapted to closures as follows:

$$\begin{aligned} v^n.l_j &\rightarrowtail t_j[(x_j, v^n) \cdot e_j] \\ &\qquad \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = (\varsigma(x_i)t_i)[e_i]^{i \in \{1..n\}}] \\ v^n.l_j \Leftarrow (\varsigma(x)t)[e] &\rightarrowtail [l_j = (\varsigma(x)t)[e], l_i = (\varsigma(x_i)t_i)[e_i]^{i \in \{1..n\} \setminus \{j\}}] \\ &\qquad \text{if } 1 \le j \le n, \text{ where } v^n = [l_i = (\varsigma(x_i)t_i)[e_i]^{i \in \{1..n\}}] \end{aligned}$$

As could be expected, there is also a contraction rule for looking variables up in the environment:

$$\begin{aligned} x[e] &\rightarrowtail v \\ &\qquad \text{if } \mathit{lookup}\,(x, e) = v \end{aligned}$$

In addition, we need three contraction rules to propagate the environment inside the terms:

$$\begin{aligned} [l_i = \varsigma(x_i)t_i{}^{i \in \{1..n\}}][e] &\rightarrowtail [l_i = (\varsigma(x_i)t_i)[e]^{i \in \{1..n\}}] \\ (t.l)[e] &\rightarrowtail t[e].l \\ (t.l \Leftarrow \varsigma(x)t')[e] &\rightarrowtail t[e].l \Leftarrow (\varsigma(x)t')[e] \end{aligned}$$

The grammar of potential redexes therefore reads as follows:

$$\begin{aligned} pr ::= {} & v.l \mid v.l \Leftarrow (\varsigma(x)t)[e] \mid \\ & x[e] \mid [l = \varsigma(x)t, \ldots, l = \varsigma(x)t][e] \mid (t.l)[e] \mid (t.l \Leftarrow \varsigma(x)t')[e] \end{aligned}$$

BNF of reduction contexts: The grammar for reduction contexts is the same as in Section 3.2.

Lemma 2 (Unique decomposition). *Any closure which is not a value can be uniquely decomposed into a reduction context and a potential redex.*

One is then in position to define a decomposition function mapping a closure to either a value or to a reduction context and a potential redex, a contraction

function mapping an actual redex to its contractum, and a plug function mapping a reduction context and a closure to a closure. Thus equipped, one can define a one-step reduction function (noted $\rightarrow$ below) and then an evaluation function as the iteration of the one-step reduction function (noted $\rightarrow^*$ below).

Applying the syntactic correspondence yields the abstract machine from Section 3.2.

The following proposition is a corollary of the soundness of refocusing:

Proposition 4 (Full correctness). *For any closed term t, $\langle t, \bullet, [\,] \rangle \Rightarrow_E^* v$ if and only if $t[\bullet] \rightarrow^* v$.*

3.4 Summary and Conclusion

On the ground that practical implementations do not use actual substitutions, we have presented an analogue of the ς-calculus, the $\varsigma\rho$-calculus, that uses explicit substitutions. We have inter-derived three semantics artifacts for the $\varsigma\rho$-calculus: a natural semantics, an abstract machine, and a reduction semantics. These specifications are more suitable to support the formalization of a compiler since programs do not change (through substitution) in the course of execution. One is then free to change their representation, e.g., by compiling them.

On the other hand, environments open the issue of space leaks since some of their bindings may become obsolete but can only be recycled when the environment itself it recycled. In functional programming, "flat" closures [13] (or again "display" closures [26]) are used instead: closures whose environment is restricted to the free variables of the term in the closure, which can be computed at compile time. The ς-calculus, however, is too dynamic in general for free variables to be computable at compile time: they need to be computed at run time. One could thus consider another possibility: to represent environments as a lightweight dictionary where each variable only occurs once.

4 Coherence between the ς-Calculus and the $\varsigma\rho$-Calculus

We establish the coherence between the ς-calculus and the $\varsigma\rho$-calculus by showing that their abstract machines are bisimilar (Section 4.2). To this end, we first introduce substitution functions mapping constructs from the $\varsigma\rho$-calculus to the ς-calculus (Section 4.1).

4.1 From Closures to Terms

We define by simultaneous induction three substitution functions that respectively map $\varsigma\rho$-values to ς-values, $\varsigma\rho$-terms to ς-terms, and environments of $\varsigma\rho$-values to temporary environments of ς-values and variables:

$$sub_{\mathbf{V}}([l_i = (\varsigma(x_i)t_i)[e_i]^{i\in\{1..n\}}]) = [l_i = \varsigma(x_i)sub_{\mathbf{T}}(t_i, (x_i,\, x_i) \cdot sub_{\mathbf{E}}(e_i))^{i\in\{1..n\}}]$$

$$\begin{aligned} sub_{\mathbf{T}}(x, e) &= lookup\,(x,\, e) \\ sub_{\mathbf{T}}(t.l, e) &= (sub_{\mathbf{T}}(t, e)).l \\ sub_{\mathbf{T}}(t.l \Leftarrow \varsigma(x)t', e) &= (sub_{\mathbf{T}}(t, e)).l \Leftarrow \varsigma(x)sub_{\mathbf{T}}(t', (x,\, x) \cdot e) \end{aligned}$$

$$\begin{aligned} sub_{\mathbf{E}}(\bullet) &= \bullet \\ sub_{\mathbf{E}}((x,\, v) \cdot e) &= (x,\, sub_{\mathbf{V}}(v)) \cdot sub_{\mathbf{E}}(e) \end{aligned}$$

Lemma 3. *For any closed term t and any environment e, $sub_{\mathbf{T}}(t, e) = t$.*

Proof. By simultaneous induction on the definition of $sub_{\mathbf{V}}$, $sub_{\mathbf{T}}$, and $sub_{\mathbf{E}}$ [30].

Let us also define a substitution function $sub_{\mathbf{C}}$ that maps $\varsigma\rho$-contexts to ς-contexts:

$$\begin{aligned} sub_{\mathbf{C}}([\,]) &= [\,] \\ sub_{\mathbf{C}}(C[[\,].l]) &= (sub_{\mathbf{C}}(C))[[\,].l] \\ sub_{\mathbf{C}}(C[[\,].l \Leftarrow (\varsigma(x)t)[e]]) &= (sub_{\mathbf{C}}(C))[[\,].l \Leftarrow \varsigma(x)sub_{\mathbf{T}}(t, (x,\, x) \cdot sub_{\mathbf{E}}(e))] \end{aligned}$$

4.2 A Bisimulation between the Two Abstract Machines

Definition 1. *Let $ST_{\varsigma\rho}$ denote the set of states of the abstract machine for the $\varsigma\rho$-calculus, and ST_{ς} denote the set of states of the abstract machine for the ς-calculus. The substitution relation $\simeq_S$: $ST_{\varsigma\rho} \times ST_{\varsigma}$ is defined as follows:*

$$\begin{aligned} \langle t,\, e,\, C\rangle &\simeq_S \langle sub_{\mathbf{T}}(t, e),\, sub_{\mathbf{C}}(C)\rangle \\ \langle C,\, v\rangle &\simeq_S \langle sub_{\mathbf{C}}(C),\, sub_{\mathbf{V}}(v)\rangle \\ v &\simeq_S sub_{\mathbf{V}}(v) \end{aligned}$$

Theorem 1. *The abstract machines from Sections 2.2 and 3.2 are weakly bisimilar with respect to $\simeq_S$.*

Proof. By co-induction on the execution of the abstract machine for the $\varsigma\rho$-calculus [30].

5 Related Work

The ς-calculus has already proved a fruitful playground. For example, Kesner and López [33] have defined a set of contraction rules for the ς-calculus based on explicit substitutions and flat closures. Due to the dynamic nature of the ς-calculus, and as already pointed out in Section 3.4, managing flat closures requires the evaluator to recompute sets of free variables dynamically during evaluation. In contrast, we opted for deep closures here. For another example, Gordon, Hankin and Lassen [29] have considered an imperative version of the

ς-calculus extended with λ-terms. They have defined a natural semantics based on explicit substitutions for their extended calculus, and proved it equivalent to substitution-based big-step and small-step semantics. In addition, they also provided a compiler to and a decompiler from a ZINC-like virtual machine [35]. Our approach is more inter-derivational and mechanical.

6 Conclusion and Issues

We have presented an abstract machine that mediates between Abadi and Cardelli's reduction semantics and natural semantics for the ς-calculus. We have then presented a version of the ς-calculus with explicit substitutions, the $\varsigma\rho$-calculus, and inter-derived a natural semantics, an abstract machine, and a reduction semantics for it. By construction, each of these three semantic artifacts is sound with respect to the two others. We have also shown that the abstract machines for the ς-calculus and for the $\varsigma\rho$-calculus are bisimilar, thereby establishing a coherence between the ς-calculus and the $\varsigma\rho$-calculus.

In the conclusion of "A Syntactic Correspondence between Context-Sensitive Calculi and Abstract Machines" [11], Biernacka and Danvy listed 16 distinct, independently published specifications of the control operator call/cc, and candidly asked whether all these artifacts define the same call/cc. It is the authors' belief that inter-deriving these artifacts using correct transformations puts one in position to answer this question.

As a side benefit, the nature of each inter-derivation makes it possible to pinpoint the specific goodness of each of the semantic artifacts. For example, a calculus in the form of a reduction semantics makes it possible to state equations to reason about programs; an abstract machine gives one some idea about the implementation requirements of a run-time system; and an interpreter in the form of a natural semantics is well suited for prototyping. We have illustrated these issues here with Abadi and Cardelli's untyped calculus of objects.

Acknowledgments: This article was written while the first author was visiting the PPS lab at the Université Paris 7 – Denis Diderot on a 'poste rouge' from the CNRS, in the winter of 2007–2008. The first author is grateful to his office mate at PPS, Paul-André Melliès, for insightful discussions and for the description of his completeness proof of the ς-calculus.

This work is partly supported by the Danish Natural Science Research Council, Grant no. 21-03-0545.

References

1. Abadi, M., Cardelli, L.: A Theory of Objects. In: Monographs in Computer Science. Springer, Heidelberg (1996)
2. Abadi, M., Cardelli, L., Curien, P.-L., Lévy, J.-J.: Explicit substitutions. Journal of Functional Programming 1(4), 375–416 (1991); A preliminary version was presented at the Seventeenth Annual ACM Symposium on Principles of Programming Languages (POPL 1990) (1990)

3. Ager, M.S.: Partial Evaluation of String Matchers & Constructions of Abstract Machines. PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark (January 2006)
4. Ager, M.S., Biernacki, D., Danvy, O., Midtgaard, J.: A functional correspondence between evaluators and abstract machines. In: Miller, D. (ed.) Proceedings of the Fifth ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP 2003), Uppsala, Sweden, August 2003, pp. 8–19. ACM Press, New York (2003)
5. Ager, M.S., Danvy, O., Midtgaard, J.: A functional correspondence between call-by-need evaluators and lazy abstract machines. Information Processing Letters 90(5), 223–232 (2004); Extended version available as the research report BRICS RS-04-3
6. Ager, M.S., Danvy, O., Midtgaard, J.: A functional correspondence between monadic evaluators and abstract machines for languages with computational effects. Theoretical Computer Science 342(1), 149–172 (2005); Extended version available as the research report BRICS RS-04-28
7. Barendregt, H.: The Lambda Calculus: Its Syntax and Semantics. Studies in Logic and the Foundation of Mathematics, vol. 103, revised edn. North-Holland, Amsterdam (1984)
8. Biernacka, M.: A Derivational Approach to the Operational Semantics of Functional Languages. PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark (January 2006)
9. Biernacka, M., Biernacki, D., Danvy, O.: An operational foundation for delimited continuations in the CPS hierarchy. Logical Methods in Computer Science 1(2:5), 1–39 (2005); A preliminary version was presented at the Fourth ACM SIGPLAN Workshop on Continuations (CW 2004) (2004)
10. Biernacka, M., Danvy, O.: A concrete framework for environment machines. ACM Transactions on Computational Logic 9(1), 1–30, Article #6 (2007); Extended version available as the research report BRICS RS-06-3
11. Biernacka, M., Danvy, O.: A syntactic correspondence between context-sensitive calculi and abstract machines. Theoretical Computer Science 375(1-3), 76–108 (2007); Extended version available as the research report BRICS RS-06-18
12. Biernacki, D.: The Theory and Practice of Programming Languages with Delimited Continuations. PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark (December 2005)
13. Cardelli, L.: Compiling a functional language. In: Steele Jr., G.L. (ed.) Conference Record of the 1984 ACM Symposium on Lisp and Functional Programming, Austin, Texas, August 1984, pp. 208–217. ACM Press, New York (1984)
14. Church, A.: The Calculi of Lambda-Conversion. Princeton University Press, Princeton (1941)
15. Consel, C., Danvy, O.: Tutorial notes on partial evaluation. In: Graham, S.L. (ed.) Proceedings of the Twentieth Annual ACM Symposium on Principles of Programming Languages, Charleston, South Carolina, January 1993, pp. 493–501. ACM Press, New York (1993)
16. Cousineau, G., Curien, P.-L., Mauny, M.: The Categorical Abstract Machine. Science of Computer Programming 8(2), 173–202 (1987)
17. Curien, P.-L.: An abstract framework for environment machines. Theoretical Computer Science 82, 389–402 (1991)
18. Curien, P.-L., Hardin, T., Lévy, J.-J.: Confluence properties of weak and strong calculi of explicit substitutions. Journal of the ACM 43(2), 362–397 (1996)

19. Danvy, O.: Back to direct style. Science of Computer Programming 22(3), 183–195 (1994); A preliminary version was presented at the Fourth European Symposium on Programming (ESOP 1992) (1992)
20. Danvy, O.: From reduction-based to reduction-free normalization. In: Antoy, S., Toyama, Y. (eds.) Proceedings of the Fourth International Workshop on Reduction Strategies in Rewriting and Programming (WRS 2004), Invited talk, Aachen, Germany, May 2004. Electronic Notes in Theoretical Computer Science, vol. 124(2), pp. 79–100. Elsevier Science, Amsterdam (2004)
21. Danvy, O.: An Analytical Approach to Program as Data Objects. DSc thesis, Department of Computer Science, University of Aarhus, Aarhus, Denmark (October 2006)
22. Danvy, O., Millikin, K.: Refunctionalization at work. Science of Computer Programming (in press); A preliminary version is available as the research report BRICS RS-07-7
23. Danvy, O., Millikin, K.: On the equivalence between small-step and big-step abstract machines: a simple application of lightweight fusion. Information Processing Letters 106(3), 100–109 (2008)
24. Danvy, O., Nielsen, L.R.: Defunctionalization at work. In: Søndergaard, H. (ed.) Proceedings of the Third International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP 2001), Firenze, Italy, September 2001, pp. 162–174. ACM Press, New York (2001); Extended version available as the research report BRICS RS-01-23
25. Danvy, O., Nielsen, L.R.: Refocusing in reduction semantics. Research Report BRICS RS-04-26, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark (November 2004); A preliminary version appeared in the informal proceedings of the Second International Workshop on Rule-Based Programming (RULE 2001), Electronic Notes in Theoretical Computer Science 59(4)
26. Dybvig, R.K.: The development of Chez Scheme. In: Lawall, J.L. (ed.) Proceedings of the 2006 ACM SIGPLAN International Conference on Functional Programming (ICFP 2006), Keynote talk, Portland, Oregon, September 2006. SIGPLAN Notices, vol. 41(9), pp. 1–12. ACM Press, New York (2006)
27. Felleisen, M.: The Calculi of λ-v-CS Conversion: A Syntactic Theory of Control and State in Imperative Higher-Order Programming Languages. PhD thesis, Computer Science Department, Indiana University, Bloomington, Indiana (August 1987)
28. Felleisen, M., Flatt, M.: Programming languages and lambda calculi (1989-2001) (last accessed, April 2008), unpublished lecture notes available at `http://www.ccs.neu.edu/home/matthias/3810-w02/readings.html`
29. Gordon, A.D., Hankin, P.D., Lassen, S.B.: Compilation and equivalence of imperative objects. Journal of Functional Programming 9(4), 373–426 (1999); Extended version available as the technical report BRICS RS-97-19
30. Johannsen, J.: Master's thesis, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark (forthcoming, 2008)
31. Jones, N.D., Gomard, C.K., Sestoft, P.: Partial Evaluation and Automatic Program Generation. Prentice-Hall International, London (1993), `http://www.dina.kvl.dk/~sestoft/pebook/`
32. Kahn, G.: Natural semantics. In: Brandenburg, F.J., Wirsing, M., Vidal-Naquet, G. (eds.) STACS 1987. LNCS, vol. 247, pp. 22–39. Springer, Heidelberg (1987)
33. Kesner, D., López, P.E.M.: Explicit substitutions for objects and functions. Journal of Functional and Logic Programming Special issue 2 (1999); A preliminary version was presented at PLILP 1998/ALP 1998 (1998)

34. Landin, P.J.: The mechanical evaluation of expressions. The Computer Journal 6(4), 308–320 (1964)
35. Leroy, X.: The Zinc experiment: an economical implementation of the ML language. Rapport Technique 117, INRIA Rocquencourt, Le Chesnay, France (February 1990)
36. Marlow, S., Peyton Jones, S.L.: Making a fast curry: push/enter vs. eval/apply for higher-order languages. In: Fisher, K. (ed.) Proceedings of the 2004 ACM SIGPLAN International Conference on Functional Programming (ICFP 2004), Snowbird, Utah, September 2004. SIGPLAN Notices, vol. 39(9), pp. 4–15. ACM Press, New York (2004)
37. Midtgaard, J.: Transformation, Analysis, and Interpretation of Higher-Order Procedural Programs. PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark (June 2007)
38. Millikin, K.: A Structured Approach to the Transformation, Normalization and Execution of Computer Programs. PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark (May 2007)
39. Nielsen, L.R.: A study of defunctionalization and continuation-passing style. PhD thesis, BRICS PhD School, University of Aarhus, Aarhus, Denmark, BRICS DS-01-7 (July 2001)
40. Nielson, H.R., Nielson, F.: Semantics with Applications, a formal introduction. Wiley Professional Computing. John Wiley and Sons, Chichester (1992)
41. Reynolds, J.C.: Definitional interpreters for higher-order programming languages. In: Proceedings of 25th ACM National Conference, Boston, Massachusetts, pp. 717–740 (1972); reprinted in Higher-Order and Symbolic Computation 11(4), 363–397 (1998) with a foreword [42]
42. Reynolds, J.C.: Definitional interpreters revisited. Higher-Order and Symbolic Computation 11(4), 355–361 (1998)
43. Rose, K.H.: Explicit substitution – tutorial & survey. BRICS Lecture Series LS-96-3, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark (September 1996)

On the Descriptive Complexity of Linear Algebra

Anuj Dawar

University of Cambridge Computer Laboratory, William Gates Building,
J.J. Thomson Avenue, Cambridge, CB3 0FD, UK
Anuj.Dawar@cl.cam.ac.uk

Abstract. The central open question in the field of descriptive complexity theory is whether or not there is a logic that expresses exactly the polynomial-time computable properties of finite structures. It is known, from the work of Cai, Fürer and Immerman that fixed-point logic with counting ($\mathsf{FP} + \mathsf{C}$) does not suffice for this purpose. Recent work has shown that natural problems involving systems of linear equations are not definable in this logic. This focuses attention on problems of linear algebra as a possible source of new extensions of the logic. Here, I explore the boundary of definability in $\mathsf{FP} + \mathsf{C}$ with respect to problems from linear algebra and look at suggestions on how the logic might be extended.

1 Introduction

The central open question in the field of descriptive complexity theory is whether or not there is a logic that expresses exactly the polynomial-time computable properties of finite structures. This question was first posed by Chandra and Harel [7] in the context of database theory and reformulated in logical form by Gurevich [19]. To be precise, the question is whether there is a logic $\mathcal{L}$ such that for each sentence φ the class of finite structures satisfying φ is decidable in polynomial time by an algorithm that can be effectively obtained from φ and, for each polynomial time decidable class Q of finite structures there is a sentence of $\mathcal{L}$ whose finite models are exactly the structures in Q. The original motivation for the question given by Chandra and Harel was the desire to have a database query language that balanced rich expressive power with guarantees of feasible computability. However, the question also has a more fundamental interest, related to our basic understanding of polynomial-time computation. If there is a logic capturing P, that shows that we can construct polynomial-time algorithms for all problems in P by composing a few basic operations—corresponding to the constructs of the logic. On the other hand, to show that there is no such logic (and this can be made mathematically precise, see [19]), is to show that no finite basis of operations will do. Furthermore, since it is known from the results of Fagin [15] that existential second-order logic captures NP, a proof that there is no logic capturing P would show a fundamental difference between these two complexity classes and indeed separate them (see [22] for a more detailed explanation).

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 17–25, 2008.

2 Fixed-Point and Counting Logics

It has been noted (see [25]) that first-order logic has two fundamental weaknesses when it comes to expressing feasibly computable properties: (1) the lack of a mechanism for recursive definitions, which means, for instance, that the transitive closure of a definable relation is not itself first-order definable; and (2) the inability to count which means, for instance, that the computationally simple property of having an even number of elements is not first-order definable. Extensions of first-order logic have been defined to address these weaknesses.

The addition to first-order logic of a mechanism for recursive definitions leads us to the logic LFP of least fixed points. This adds to the rules of first-order logic the following formula-formation rule: if φ is a formula, *positive* in the relational variable R, $\mathbf{x}$ is a tuple of first-order variables and $\mathbf{t}$ a tuple of terms, both of the same length as the arity of R, then $[\mathbf{lfp}_{R,\mathbf{x}}\varphi](\mathbf{t})$ is also a formula (see [14] or [23] for a more detailed exposition of this logic). The intended reading of the formula is that the tuple $\mathbf{t}$ is in the least fixed point of the operator that maps a relation R to the relation defined by $\varphi(R, \mathbf{x})$. A logic on these lines was first-proposed as a query language for databases by Aho and Ullman [1]. It is easy to show that every formula of LFP still defines a polynomial-time decidable property. If we consider only ordered structures, then it turns out that LFP is powerful enough to express all polynomial-time properties, as was shown independently by Immerman and Vardi [21,26]. The presence of the order is crucial to this result, as it enables the inductive definition of a polynomial-time Turing machine computation. In the absence of a linear order, the second weakness of first-order logic still manifests itself. That is, LFP on unordered structures still cannot express simple counting properties.

Fixed-Point Logic with Counting. Immerman [21] suggested adding a mechanism for counting to the logic LFP in the hope of obtaining a logic that characterises P. To be precise, the logic $\mathsf{FP}+\mathsf{C}$ has two sorts of variables: $v_1, v_2, \ldots$ ranging over the domain elements of the structure, and $\nu_1, \nu_2, \ldots$ ranging over the numbers. When a formula φ is interpreted in a structure $\mathbb{A}$, the number variables occurring in φ are interpreted as ranging over the set $\{0, \ldots, n\}$ where n is the number of elements in $\mathbb{A}$. In addition, we also have second order variables $X_1, X_2, \ldots$, each of which has a type, which is a finite string in $\{\text{element}, \text{number}\}^*$. Thus, for instance, if X is a variable of type (element,number), it is to be interpreted by a binary relation relating elements to numbers. The logic allows us to build up *counting terms* according to the following rule: if φ is a formula and x is a variable of the first sort, then $\#x\varphi$ is a term. The intended semantics is that $\#x\varphi$ denotes the number (i.e. the member of the number sort) of elements that satisfy the formula φ.

The formulas of $\mathsf{FP}+\mathsf{C}$ are now described by the following set of rules:

- all atomic formulas of first-order logic are formulas of $\mathsf{FP}+\mathsf{C}$;
- if τ_1 and τ_2 are terms of numeric sort (that is each one is either a number variable or a term of the form $\#x\varphi$) then each of $\tau_1 < \tau_2$ and $\tau_1 = \tau_2$ is a formula;

- if φ and ψ are formulas then so are $\varphi \wedge \psi$, $\varphi \vee \psi$ and $\neg\psi$;
- if φ is a formula, x is an element variable and ν is a number variable then $\exists x\, \varphi$ and $\exists \nu\, \varphi$ are formulas; and
- if X is a relation symbol of type σ; $\mathbf{x}$ is a tuple of variables whose sorts match the type σ; $\mathbf{t}$ is a tuple of terms of type σ; and φ is a formula in which X only appears positively, then $[\mathbf{lfp}_{X,\mathbf{x},\nu}\varphi](\mathbf{t})$ is a formula.

Infinitary Logic with Counting. It turns out that $\mathsf{FP} + \mathsf{C}$ is still too weak to express all polynomial-time properties of finite structures, as was proved by Cai et al. [6]. To put this result in context, it is useful to introduce $C^\omega_{\infty\omega}$, an infinitary logic with counting. $C^\omega_{\infty\omega}$ is obtained from first-order logic by allowing:

- *infinitary* conjunctions and disjunctions: $\bigvee\{\varphi \mid \varphi \in S\} \quad \bigwedge\{\varphi \mid \varphi \in S\}$ for arbitrary sets S;
- *counting quantifiers*: $\exists^i x\, \varphi$; and
- only finitely many distinct variables in any formula.

$C^k_{\infty\omega}$ is the fragment of $C^\omega_{\infty\omega}$ where each formula has at most k variables. It can be shown that every sentence of $\mathsf{FP} + \mathsf{C}$ is equivalent to one of $C^\omega_{\infty\omega}$. Indeed, it has been shown [24] that the expressive power of $\mathsf{FP} + \mathsf{C}$ corresponds to the polynomial-time uniform fragment of $C^\omega_{\infty\omega}$.

The expressive power of $C^\omega_{\infty\omega}$ is not confined to P, or indeed to decidable properties. However, the construction of Cai et al. shows that there are polynomial-time properties of graphs that are not definable even in $C^\omega_{\infty\omega}$. More precisely, they show how to construct a sequence of pairs of graphs $G_k, H_k (k \in \omega)$ such that:

- $G_k \equiv^{C^k_{\infty\omega}} H_k$ for all k, i.e. G_k and H_k cannot be distinguished by any sentence of $C^k_{\infty\omega}$
- There is a polynomial time decidable class of graphs that includes G_k, for all k, and excludes all H_k.

The graphs G_k and H_k are obtained from a single graph G by a transformation that replaces every edge by a pair of edges and every vertex by a gadget (see [22, Chap. 13] for details). They are non-isomorphic graphs but it is shown that if G has no balanced separator of fewer than k vertices, than $G_k \equiv^{C^k_{\infty\omega}} H_k$. Indeed, we can show [13] that a weaker condition, namely that G is connected and has *treewidth* greater than k is sufficient. It seems likely that the latter condition is also necessary.

Following the proof of Cai et al. further inexpressibility results have been obtained by similar methods. Gurevich and Shelah [20] constructed a class of structures called *multipedes* that is first-order definable, contains only rigid structures and on which no linear order is uniformly definable in $C^\omega_{\infty\omega}$. I was able to show [9] that 3-colourability of graphs is not definable in $C^\omega_{\infty\omega}$. The constructions involved in the proofs of both these results bear a strong family resemblance to the construction of Cai, Fürer and Immerman.

In spite of the fact that $\mathsf{FP} + \mathsf{C}$ is too weak to express all polynomial-time computable properties, it forms a natural level of expressiveness it its own right

and has been shown to capture P on many interesting restricted classes of structures, such as planar graphs [17] and graphs of bounded treewidth [18]. Moreover, since the examples of polynomial-time properties that had been proved to be inexpressible in $\mathsf{FP} + \mathsf{C}$, such as the Cai-Fürer-Immerman graphs and the ordering on multipedes, were not the kind of natural queries one might ask, it was sometimes said that all *natural* polynomial-time queries are expressible in $\mathsf{FP} + \mathsf{C}$ (3-colourability is certainly a natural query that was proved inexpressible in $C^{\omega}_{\infty\omega}$, but it is not known or believed to be in P). Nevertheless, it has recently been shown that underlying all these various examples, there is a natural polynomial-time decidable problem that is not expressible in $\mathsf{FP} + \mathsf{C}$, namely the solvability of linear equations over a two-element field.

3 Problems from Linear Algebra

We begin by examining how we can represent systems of linear equations over the field $\mathbb{Z}/2\mathbb{Z}$ as unordered relational structures. First, consider a system of equations with at most three variables per equation, so each equation is of the form: $x_1 + x_2 + x_3 = a$ with ($a = 0$ or 1). We consider this system as a relational structure over the domain $\{x_1, \ldots, x_n\}$ of variables with two ternary relations:

$$R_0 = \{(x_i, x_j, x_k) \mid x_i + x_j + x_k = 0 \text{ is an equation}\}$$
$$R_1 = \{(x_i, x_j, x_k) \mid x_i + x_j + x_k = 1 \text{ is an equation}\}$$

Let $\mathsf{Solv}_3(\mathbb{Z}/2\mathbb{Z})$ be the class of such structures that represent solvable systems. It is shown in [2] that $\mathsf{Solv}_3(\mathbb{Z}/2\mathbb{Z})$ is not definable in $C^{\omega}_{\infty\omega}$.

For the more general case, where we do not limit ourselves to three variables per equation, we can consider structures over the domain $\{x_1, \ldots, x_n, e_1, \ldots, e_m\}$ where the x_i are variables as before and the e_j are the equations. We have the following relations on this domain:

- a unary relation E_0 that includes just those equations e whose right hand side is 0;
- a unary relation E_1 that includes just those equations e whose right hand side is 1; and
- a binary relation M with $M(x, e)$ if x occurs on the left hand side of e.

Writing $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$ for the class of such structures representing solvable equations, it is not difficult to show that $\mathsf{Solv}_3(\mathbb{Z}/2\mathbb{Z})$ is reducible by a first-order reduction to $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$. Since $C^{\omega}_{\infty\omega}$ is closed under first-order reductions, it follows from the above-mentioned result of [2] that $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$ is not definable in $C^{\omega}_{\infty\omega}$ either.

It should be noted that $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$ is indeed a polynomial-time decidable class as systems of equations can be efficiently solved, for instance by Gaussian elimination. The construction in [2] to show that $\mathsf{Solv}_3(\mathbb{Z}/2\mathbb{Z})$ is not definable in $C^{\omega}_{\infty\omega}$ is similar in form to that of Cai, Fürer and Immerman [6] and can be seen to show that underlying the latter construction is indeed the undefinability in

$\mathsf{FP}+\mathsf{C}$ of the solvability of linear equations. Indeed, it can be shown that both the Cai-Fürer-Immerman graphs and the problem of ordering multipedes are reducible to $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$ [10] by first-order definable reductions. Of course, we do not expect to show that 3-colourability is reducible to $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$ as this would imply $\mathsf{P}=\mathsf{NP}$. However, the construction in [9] that shows 3-colourability is not definable in $C^{\omega}_{\infty\omega}$ uses a special class of graphs and it is not difficult to show that on these the problem is efficiently decidable, and reducible to $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$. In other words, $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$ emerges as the canonical polynomial-time problem that is not definable in $\mathsf{FP}+\mathsf{C}$. This leads us naturally to ask how much linear algebra is expressible in $\mathsf{FP}+\mathsf{C}$. It turns out that a surprising amount is.

We can identify a binary relation $A \subseteq I \times I$ over a set I with a matrix M over $\mathbb{Z}/2\mathbb{Z}$ by letting $M_{ij} = 1 \Leftrightarrow (i,j) \in A$. Note, the set I (which we call the index set) is *unordered*, so we are going to be interested in properties of the matrix M that are invariant under simultaneous permutations of its rows and columns. Equivalently, seeing M as a linear operator on the vector space $(\mathbb{Z}/2\mathbb{Z})^I$, we are interested in those properties of the operator that are invariant under a permutation of the basis. This includes most natural properties of M.

We begin by noting that it is easy to express matrix multiplication in $\mathsf{FP}+\mathsf{C}$. Define the formula prod as follows.

$$\mathsf{prod}(x,y,A,B) = \exists\nu_1\exists\nu_2(\nu_1 = \#z(A(x,z)\wedge B(z,y)) \wedge (\nu_1 = 2\cdot\nu_2 + 1)).$$

It is easily seen that the binary relation defined by $\mathsf{prod}(x,y)$ is the product of the two matrices A and B. A simple application of **lfp** then allows us to define $\mathsf{upower}(x,y,\nu,A)$ which gives the matrix A^{ν}. To be precise, we define the formula by:

$$\begin{aligned}\mathsf{upower}(x,y,\nu,A) = [\mathbf{lfp}_{R,uv}\,(\forall\mu(\nu\le\mu)\wedge u = v \vee \\ \exists\mu(\nu = \mu+1 \wedge \mathsf{prod}(u,v,B/R(\mu),A))](x,y),\end{aligned}$$

where $\nu = \mu + 1$ is defined in the natural way on the number domain and $\mathsf{prod}(u,v,B/R(\mu),A)$ is the formula obtained from $\mathsf{prod}(u,v,A,B)$ by replacing the occurrence of $B(z,v)$ by $R(z,v,\mu)$.

Instead of representing the power to which we wish to raise the matrix A by a single numerical variable ν, we can represent it in binary, as a set of numeric values. In other words, we take a unary relation Γ of numeric sort to code the number $\sum_{\gamma\in\Gamma} 2^{\gamma}$. This allows us to use repeated squaring to define a formula $\mathsf{power}(x,y,\Gamma,A)$ giving A^N where Γ codes a number N whose value may be exponentially larger than the size of I (the index set). This now enables us to show that non-singularity of matrices is definable in $\mathsf{FP}+\mathsf{C}$, by an argument due to Blass et al. [4]. Consider $\mathsf{GL}(n,\mathbb{Z}/2\mathbb{Z})$, the *general linear group* of degree n over $\mathbb{Z}/2\mathbb{Z}$, i.e. the group of non-singular $n\times n$ matrices over $\mathbb{Z}/2\mathbb{Z}$. Blass et al. note that the order of this group is $\prod_{i=0}^{n-1}(2^n - 2^i)$, and a unary relation representing this number can be expressed by a formula of $\mathsf{FP}+\mathsf{C}$. Thus, we can write a formula representing A^N, and A is non-singular if, and only if, this is the identity matrix. By the same token, we can define the inverse of A since $A^{-1} = A^{N-1}$.

We could ask the same definability questions with regard to matrices over an arbitrary finite field $\mathbb{F}_q$ other than $\mathbb{Z}/2\mathbb{Z}$. We can represent matrices M over $\mathbb{F}_q$ by taking, for each $a \in \mathbb{F}_q$, a binary relation $A_a \subseteq I \times I$ with $M_{ij} = a \Leftrightarrow (i,j) \in A_a$. Alternatively, we could have the elements of $\mathbb{F}_q$ (along with the field operations) as a separate sort and include a ternary relation R with $M_{ij} = a \Leftrightarrow (i,j,a) \in R$. It is easy to see that these two representations are interdefinable, in that, for each fixed field $\mathbb{F}_q$, we can define a first-order interpretation of either one in the other.

The definability results for matrices over $\mathbb{Z}/2\mathbb{Z}$ extend easily to other finite fields. In particular, the argument for the definability of non-singularity and of the inverse of a matrix are easily seen to apply. The proof that the solvability of systems of equations is undefinable extends similarly. However, computing the determinant of a matrix is a more interesting problem. In the case of matrices over $\mathbb{Z}/2\mathbb{Z}$ determining singularity is the same thing as computing the determinant as there is only one possible non-zero value that the determinant can take. The problem of computing the determinant over other fields is somewhat trickier. However, Rossman observed that Csanky's algorithm [8] for computing the determinants of integer matrices can be expressed in the formalism of choiceless polynomial time with counting. As noted by Blass and Gurevich [3] this means that computing the determinant over any finite field can also be expressed in this formalism. My student, Bjarki Holm [10], has strengthened this by showing that these algorithms do yield definability of the determinant in $\mathsf{FP} + \mathsf{C}$, both for integer matrices and for matrices over an arbitrary finite field, or indeed over the integers.

While the determinant of a matrix can be defined in $\mathsf{FP} + \mathsf{C}$, it is clear that there can be no formula of the logic that defines the *rank* of a matrix, either over $\mathbb{Z}/2\mathbb{Z}$ or any other finite field. If such a formula existed, we would also be able to define $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$, as from an instance of $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$ representing a system of equations $A\mathbf{x} = \mathbf{b}$, we can easily define the two matrices A and $A\mathbf{b}$, and the system is solvable if, and only if, the two matrices have the same rank.

It appears that linear algebra provides an interesting frontier in which to explore the limits of definability of $\mathsf{FP} + \mathsf{C}$. Some problems, such as computing the determinant of a matrix, turn out to be definable though through non-trivial means. On the other hand, closely related problems, such as computing the rank of a matrix are not expressible in the logic. It is worth remarking that in terms of computational complexity, these problems are very similar. To be precise, the problems of computing the determinant, the rank, or the inverse of a matrix over $\mathbb{Z}/2\mathbb{Z}$ are all complete for the complexity class $\oplus\mathsf{L}$ under logspace-reductions [5]. $\oplus\mathsf{L}$ is the complexity class containing languages L for which there is a *nondeterministic, logspace* machine M such that $x \in L$ if, and only if, the number of accepting paths of M on input x is *odd*. $\oplus\mathsf{L}$ contains the class L and is also conjectured to contain NL. A natural complete problem for $\oplus\mathsf{L}$ is $\oplus\mathsf{GAP}$ which, given an *acyclic, directed* graph G with distinguished vertices s and t asks whether the number of distinct paths from s to t is odd. It is easily verified that this problem can be expressed in $\mathsf{FP} + \mathsf{C}$ as it amounts to checking the entry in $(A^n_G)_{st}$ where A is the adjacency matrix of the graph G.

4 Further Directions

The previous section has established that problems in linear algebra provide a fertile testing ground for the expressivity of logics such as $\mathsf{FP}+\mathsf{C}$. There is a fine line between parameters of a matrix that are definable, such as the determinant, and those that are not, such as the rank. One could also say that we have discovered a third weakness in first-order logic, to go along with the lack of a mechanism for inductive definitions and the inability to count—namely, the inability to determine the row-rank of a binary relation. As it turns out, this is just the second weakness in a different guise and a more accurate summary might be that the counting mechanism that was added to LFP to obtain $\mathsf{FP}+\mathsf{C}$ turns out to be inadequate to capture the full power of counting and a more general mechanism is required.

Suppose that, instead of the counting terms $\#x\varphi$ to denote the number of elements satisfying φ, we could form a term $\mathbf{rk}_{xy}\varphi$ which binds the variables x and y in φ and denotes the number that is the row-rank of the binary relation $\varphi(x,y)$. We can define $\#x\varphi$ in terms of this new operator, as it is equivalent to the term $\mathbf{rk}_{xy}(x = y \wedge \varphi(x))$. Thus, the rank operator $\mathbf{rk}$ can be seen as a more general form of the counting operator. Moreover, in LFP extended with this rank operator, we can define $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$. More generally, writing $\mathsf{FP}+\mathsf{R}$ for the logic that extends LFP with a rank operator of each arity (i.e. it allows us to form terms $\mathbf{rk}^{r}_{\mathbf{xy}}\varphi$, where $\mathbf{x}$ and $\mathbf{y}$ are r-tuples of variables) we find that we can define in $\mathsf{FP}+\mathsf{R}$ the Cai-Fürer-Immerman graphs, the ordering on multipedes and indeed every polynomial time problem which we have previously proved undefinable in $\mathsf{FP}+\mathsf{C}$. What is the expressive power of $\mathsf{FP}+\mathsf{R}$? Could it be all of P? This seems unlikely, but it remains a challenge to find a polynomial-time problem that is not definable in this logic. The particular rank operator may not be to everyone's taste as a logical operator. Might one find other, perhaps more general, ways of incorporating linear algebraic operators in a logic for P?

The proposal above for a logic $\mathsf{FP}+\mathsf{R}$ is not the first proposed way of extending the expressive power of $\mathsf{FP}+\mathsf{C}$ to encompass the examples of Cai, Fürer and Immerman. Other notable efforts include Choiceless Polynomial Time with Counting [4] and the logic of Specified Symmetric Choice [16,12,11]. However, it remains an open question whether either of these logics can express $\mathsf{Solv}(\mathbb{Z}/2\mathbb{Z})$.

A related open question, posed in [4] is whether the problem of *general graph matching* is definable in $\mathsf{FP}+\mathsf{C}$. It is known that matching on bipartite graphs is definable [4]. The close relationship between graph matching and problems of linear algebra leads us to include this among related directions of research.

5 Conclusion

In conclusion, in seeking to find a logic to capture P, we are seeking to build the complexity class P from below. Extending first-order logic with mechanisms for recursion and counting takes us so far, and the limits of this can be described in terms of linear algebra. Matrix rank seems to be a more general way to include

counting in the logic than a simple operator for cardinality, and takes us further. It remains an active area of research to explore the boundary of expressibility and see how much of the complexity class P we can build from below.

References

1. Aho, A.V., Ullman, J.D.: Universality of data retrieval languages. In: 6th ACM Symp. on Principles of Programming Languages, pp. 110–117 (1979)
2. Atserias, A., Bulatov, A., Dawar, A.: Affine systems of equations and counting infinitary logic. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 558–570. Springer, Heidelberg (2007)
3. Blass, A., Gurevich, Y.: A quick update on the open problems in Blass-Gurevich-Shelah's article "on polynomial time compututations over unordered structures", `http://research.microsoft.com/~gurevich/annotated.html`
4. Blass, A., Gurevich, Y., Shelah, S.: On polynomial time computation over unordered structures. Journal of Symbolic Logic 67(3), 1093–1125 (2002)
5. Buntrock, G., Damm, C., Hertrampf, U., Meinel, C.: Structure and importance of logspace-MOD class. Mathematical Systems Theory 25, 223–237 (1992)
6. Cai, J.-Y., Fürer, M., Immerman, N.: An optimal lower bound on the number of variables for graph identification. Combinatorica 12(4), 389–410 (1992)
7. Chandra, A., Harel, D.: Structure and complexity of relational queries. Journal of Computer and System Sciences 25, 99–128 (1982)
8. Csanky, L.: Fast parallel matrix inversion algorithms. SIAM J. on Computing 5, 618–623 (1976)
9. Dawar, A.: A restricted second order logic for finite structures. Information and Computation 143, 154–174 (1998)
10. Dawar, A., Holm, B. (forthcoming)
11. Dawar, A., Richerby, D.: A fixed-point logic with symmetric choice. In: Baaz, M., Makowsky, J.A. (eds.) CSL 2003. LNCS, vol. 2803, pp. 169–182. Springer, Heidelberg (2003)
12. Dawar, A., Richerby, D.: Fixed-point logics with nondeterministic choice. Journal of Logic and Computation 13, 503–530 (2003)
13. Dawar, A., Richerby, D.: The power of counting logics on restricted classes of finite structures. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, Springer, Heidelberg (2007)
14. Ebbinghaus, H.-D., Flum, J.: Finite Model Theory, 2nd edn. Springer, Heidelberg (1999)
15. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. In: Karp, R.M. (ed.) Complexity of Computation, SIAM-AMS Proceedings, vol. 7, pp. 43–73 (1974)
16. Gire, F., Hoang, H.: An extension of fixpoint logic with a symmetry-based choice construct. Information and Computation 144, 40–65 (1998)
17. Grohe, M.: Fixed-point logics on planar graphs. In: Proc. 13th IEEE Annual Symp. Logic in Computer Science, pp. 6–15 (1998)
18. Grohe, M., Mariño, J.: Definability and descriptive complexity on databases of bounded tree-width. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 70–82. Springer, Heidelberg (1998)
19. Gurevich, Y.: Logic and the challenge of computer science. In: Börger, E. (ed.) Current Trends in Theoretical Computer Science, pp. 1–57. Computer Science Press (1988)

20. Gurevich, Y., Shelah, S.: On rigid structures. Journal of Symbolic Logic 61, 549–562 (1996)
21. Immerman, N.: Relational queries computable in polynomial time. Information and Control 68, 86–104 (1986)
22. Immerman, N.: Descriptive Complexity. Springer, Heidelberg (1999)
23. Libkin, L.: Elements of Finite Model Theory. Springer, Heidelberg (2004)
24. Otto, M.: The expressive power of fixed-point logic with counting. J. Symbolic Logic 61, 147–176 (1996)
25. Otto, M.: Bounded Variable Logics and Counting — A Study in Finite Models. Lecture Notes in Logic, vol. 9. Springer, Heidelberg (1997)
26. Vardi, M.Y.: The complexity of relational query languages. In: Proc. of the 14th ACM Symp. on the Theory of Computing, pp. 137–146 (1982)

Talks on Quantum Computing

Sam Lomonaco

Department of Computer Science & Electrical Engineering,
University of Maryland Baltimore County
Lomonaco@UMBC.EDU
http://www.csee.umbc.edu/~lomonaco

Tutorial 1: A Rosetta Stone for Quantum Computing

Abstract. This talk will give an overview of quantum computing in an intuitive and conceptual fashion. No prior knowledge of quantum mechanics will be assumed.

The talk will begin with an introduction to the strange world of the quantum. Such concepts as quantum superposition, Heisenberg's uncertainty principle, the "collapse" of the wave function, and quantum entanglement (i.e., EPR pairs) are introduced. This part of the talk will also be interlaced with an introduction to Dirac notation, Hilbert spaces, unitary transformations, quantum measurement, and the density operator.

Simple examples will be given to explain and to illustrate such concepts as quantum measurement, quantum teleportation, quantum dense coding, and the first quantum algorithm, i.e., the Deutsch-Jozsa algorithm.

The PowerPoint slides for this talk will be posted at the URL: http://www.csee.umbc.edu/~lomonaco/Lectures.html

Tutorial 2: Quantum Algorithms: Past, Present, and Future

Abstract. The talk begins with the present, i.e., with an overview of the current status of quantum algorithms. The talk then moves to the past by explaining and illustrating two quantum algorithms, i.e., Grover's Search Algorithm and Shor's Factoring Algorithm. Shor's algorithm is then generalized to the larger class of quantum hidden subgroup algorithms. In this context, Grover's algorithm is then shown to be very similar to Shor's. Finally, the talk ends with future predictions, i.e., with a discussion as to how one might find and create new quantum algorithms.

The PowerPoint slides for this talk will be posted at the URL: http://www.csee.umbc.edu/~lomonaco/Lectures.html

Invited Talk: Quantum Knots and Mosaics

Abstract. In this talk, we give a precise and workable definition of a quantum knot system, the states of which are called quantum knots. This definition can

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 26–27, 2008.

be viewed as a blueprint for the construction of an actual physical quantum system.

Moreover, this definition of a quantum knot system is intended to represent the "quantum embodiment" of a closed knotted physical piece of rope. A quantum knot, as a state of this system, represents the state of such a knotted closed piece of rope, i.e., the particular spatial configuration of the knot tied in the rope. Associated with a quantum knot system is a group of unitary transformations, called the ambient group, which represents all possible ways of moving the rope around (without cutting the rope, and without letting the rope pass through itself.)

Of course, unlike a classical closed piece of rope, a quantum knot can exhibit non-classical behavior, such as quantum superposition and quantum entanglement. This raises some interesting and puzzling questions about the relation between topological and quantum entanglement. The knot type of a quantum knot is simply the orbit of the quantum knot under the action of the ambient group. We investigate quantum observables which are invariants of quantum knot type. We also study the Hamiltonians associated with the generators of the ambient group, and briefly look at the quantum tunneling of overcrossings into undercrossings.

A basic building block in this paper is a mosaic system which is a formal (rewriting) system for symbol strings. We conjecture that this formal system fully captures in an axiomatic way all of the properties of tame knot theory.

The PowerPoint slides for this talk will be posted at the URL: `http://www.csee.umbc.edu/ lomonaco/Lectures.html`

On Game Semantics of the Affine and Intuitionistic Logics
(Extended Abstract)

Ilya Mezhirov[1] and Nikolay Vereshchagin[2,⋆]

[1] The German Research Center for Artificial Intelligence,
TU Kaiserslautern
ilya.mezhirov@dfki.uni-kl.de
[2] Lomonosov Moscow State University,
Leninskie Gory, Moscow 119991
ver@mccme.ru

Abstract. We show that the uniform validity is equivalent to the non-uniform validity for both Blass' semantics of [1] and Japaridze's semantics of [5] (thus proving a conjecture from [5]). We present a shorter proof (than that of [10]) of the completeness of the positive fragment of intuitionistic logic for both these semantics. Finally, we show that validity for the "parallel recurrence" version of Japaridze's semantics of [5] is equivalent to accomplishability in the sense of [4].

1 Logic of Tasks and Intuitionistic Logic

The linear and affine logics of Girard [3] are often understood as logics of resources. Propositional variables mean certain types of abstract resources (rather than assertions, as in classical logic). Each occurrence of a variable p identifies one unit of a resource p. The connectives $\wedge$ (the multiplicative AND), $\vee$ (the multiplicative OR), $\sqcap$ (the additive AND), $\sqcup$ (the additive OR), $\neg$ (the negation), $!, ?$ (exponential AND and OR) and $\mathbf{0}, \mathbf{1}$ (constants) have the following meaning. The expression $x \wedge y$ means one unit of x and one unit of y (for example, $x \wedge x$ is two units of the resource x). The expression $x \sqcap y$ means an obligation to provide one unit of the resource x or one unit of the resource y where the consumer of resources (the user) makes the choice between x and y (for example, $x \sqcap x$ is the same as x). The formula $x \sqcup y$ means also an obligation to provide one unit of the resource x or one unit of the resource y. However, this time the choice between x and y is made by the provider (again $x \sqcup x$ is the same as x).

What is the interpretation of $\vee$ (multiplicative OR)? Let us use the following metaphor. Assume that resources are coins of different types. Each occurrence of a variable x is regarded as a coin of type x. The coins can be genuine or fake, and the consumer cannot distinguish between genuine and fake ones. The expression $x \vee y$ is understood as a pair of coins, a coin of type x and a coin of

⋆ The work was supported in part by the RFBR grants 06-01-00122.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 28–42, 2008.

type y, such that at least one of them is genuine (however, the user does not know which one).

The expression $\neg x$ means the obligation of the user to pass to the provider one coin of type x (we can understand x also in this way, as the obligation of the provider to pass to the user one coin of type x). The formula $!\,x$ is understood as an infinite stock of genuine coins of type x. The expression $?\,x$ means an infinite stock of coins of type x such that at least one coin in the stock is genuine (and the user does not know which one). In other words $!\,x$ is a countable version of the multiplicative AND $x \wedge x \wedge x \ldots$, and $?\,x$ is a countable version of the multiplicative OR $x \vee x \vee x \ldots$.

The constants $\mathbf{0}, \mathbf{1}$ are understood as non-accomplishable obligations (tasks): $\mathbf{0}$ is the obligation of the provider and $\mathbf{1}$ is the obligation of the user.

In the next section we give a formal definition of the semantics which clarifies this intuition. From both the technical and philosophical viewpoints, this semantics coincides with Japaridze's "The logic of tasks" semantics defined in [4] and later extended to what has been termed *abstract resource semantics* in [8]. What we call "a coin" is called "a task" in [4]. A genuine coin is a task accomplished by the provider. A false coin is a task that the provider failed to accomplish. We are using a different language, as we think our language is more convenient to present the definition.

1.1 The Semantics

Fix a countable set of *variables* $x_1, x_2, \ldots$. A *literal* is a variable x_i or a negated variable $\neg x_i$, called *dual* to x_i. The literals of the form x_i are called *positive* and literals of the form $\neg x_i$ are called *negative*. Formulas are obtained from literals, constants $\mathbf{1}, \mathbf{0}$ and connectives $\wedge, \vee, \sqcap, \sqcup, ?, !$ in the usual way. We consider formulas where negations appear only before variables. When we write $\neg A$, we always mean the formula dual to A, that is, the formula which is obtained from A by changing each connective, each variable and its constant to its dual: $\vee \leftrightarrow \wedge$, $\sqcup \leftrightarrow \sqcap$, $! \leftrightarrow ?$, $\neg x_i \leftrightarrow x_i$ and $\mathbf{1} \leftrightarrow \mathbf{0}$.

Each formula is assigned a two player game $[A]$ of perfect information between *Provider* (or, *Producer*) and *User* (or *Consumer*). If A is derivable in the affine logic then the Consumer will have a winning strategy in the game $[A]$.

First we replace in A each occurrence of a formula of the type $!\,B$ by the formula $B \wedge B \wedge B \ldots$, and each occurrence of a formula of the type $?\,B$ by the infinite formula $B \vee B \vee B \ldots$. It is easy to see that the order of replacements does not affect the result, which is an infinite formula of a finite depth.

In the game $[A]$, the players make moves in turn. It does not matter who moves first. In his turn, Producer may perform any finite sequence $p_1, \ldots, p_k$ of *actions* (the sequence may be empty). Each action p_i has the form "choose the left formula in $B \sqcap C$" or "choose the right formula in $B \sqcap C$", where $B \sqcap C$ is an occurrence of a formula in A.

For each occurrence of a formula $B \sqcap C$ in A Producer may only once make an action of the type "choose something in $B \sqcap C$" and he is unable to change the choices he has made.

In her turn, Consumer may also perform any finite (possibly empty) sequence of *actions*. Each her action has one of the following two forms. (1) "Choose the left (right) formula in $B \sqcup C$" where $B \sqcup C$ is an occurrence of a formula in A. (2) "Allocate U to V", where U is an occurrence of a negative literal in A and V is an occurrence of a positive literal similar to U (the literals are called similar if they have the same variable: x_i and $\neg x_i$ are similar). The informal meaning of this action is that Consumer wants to accomplish her obligation V using the coin U. It is instructive to visualise this action as follows. Imagine that each occurrence of a negative literal $\neg x$ is a box belonging to Producer and containing a coin of type x. Regard all occurrences of positive literals x as empty boxes of type x belonging to Consumer. The action $U \mapsto V$ moves the coin from the box U to the box V. According to this interpretation, we will sometimes call occurrences of the negative literals *Producer's boxes* and occurrences of the positive literals *Consumer's boxes*.

For each occurrence of a positive literal V Consumer may perform at most one action of the form $U \mapsto V$, and for each occurrence of a negative literal U she may perform at most one action of the form $U \mapsto V$ (every box can contain at most one coin). Consumer is not allowed to move coins in the other direction (from her boxes to Producer's ones). Neither is she allowed to move coins from boxes to "nowhere". In other words, actions of type (2) establish a matching between Producer's and Consumer's boxes that respects variable's names. Note that Producer is not allowed move coins at all.

For each occurrence of a formula $B \sqcup C$ in A Consumer may only once make an action of the type "choose something in $B \sqcup C$".

Each play consists of infinite (countably many) number of moves. When the play is finished, we define who has won as follows. Call a *coin evaluation* a mapping e from Producer's boxes to the set $\{\mathtt{f}, \mathtt{g}\}$. If $e(U) = \mathtt{f}$ we say that the coin in the box U is fake, otherwise (if $e(U) = \mathtt{g}$) we say that it is genuine.

Given a coin evaluation, we recursively define for each occurrence of a formula B in A who has won B, Consumer or Producer.

(1) If V is an occurrence of a positive literal, then Consumer has won V iff, in the course of the play, Consumer has performed an action "allocate U to V" and $e(U) = \mathtt{g}$. In other words, if in the end of the game the box V contains a genuine coin. If the box V is empty or contains a fake coin then Consumer has lost V.

(2) If U is an occurrence of a negative literal, then Consumer has won U iff $e(U) = \mathtt{f}$.

Rules (1) and (2) imply the following. If Consumer has performed an action "allocate U to V" then exactly one of U, V is won by Consumer.

(3) Consumer has won an occurrence of a formula $B \wedge C$ iff she has won both B and C. She has won an occurrence of a formula $B \vee C$ iff she has won B or C.

(4) In a similar way we define who has won occurrences of $!\,B$ and $?\,B$: an occurrence $B_1 \wedge B_2 \wedge B_3 \ldots$ (we use subscripts to distinguish between different occurrences of B) is won by Consumer iff she has won all the occurrences

$B_1, B_2, B_3 \ldots$. An occurrence $B_1 \vee B_2 \vee B_3 \ldots$ is won by Consumer iff she has won at least one of the occurrences $B_1, B_2, B_3 \ldots$.

(5) Consumer has won an occurrence of $B \sqcup C$ iff, in the course of the play, she has decided between B and C in $B \sqcup C$ and has won the chosen formula. That is, she has performed the action "choose the left formula $B \sqcup C$" and she has won B, or she has performed the action "choose the right formula in $B \sqcup C$" and she has won C. If she has not decided between B and C in $B \sqcup C$ then she has lost $B \sqcup C$.

(6) For a subformula of the form $B \sqcap C$ the definition is similar (we swap Consumer and Producer). Producer has won an occurrence of $A \sqcap B$ iff in the course of the play, he has decided between B and C in $B \sqcup C$ and has won the chosen formula.

(7) Every occurrence of $\mathbf{0}$ is won by Producer and every occurrence of $\mathbf{1}$ is won by Consumer.

Finally, we say that the Consumer has won the play iff for every coin evaluation e she has won the entire formula A.

There is an equivalent way to define who was won the play. In the end of the play replace by $\mathbf{0}$ all the occurrences of the formulas of type $B \sqcup C$ in A such that Consumer has not chosen B or C. Replace by $\mathbf{1}$ all the occurrences of $B \sqcap C$ where Producer has not decided between B and C. Replace each remaining occurrence of a formula of the form $B \sqcup C$ or $B \sqcap C$ by the chosen subformula. (The order of replacements does not affect the result.) Then replace each occurrence of a variable by a new variable in such a way that matching occurrences are replaced by the same variables and non-matching occurrences by different ones. (We call *matching* the occurrences of x in U and V such that Consumer has allocated U to V.) The resulting formula is an infinite formula of finite depth with connectives $\vee, \wedge$ and constants $\mathbf{1}, \mathbf{0}$. Consumer has won the play iff this formula is a classical tautology (where $\wedge$ is understood as AND, $\vee$ as OR, and $\neg x$ as the negation of x).

The above described transformation is what is called *elemntarization* in [4,5,6].

The definition of the game $[A]$ is completed. We call a formula A *accomplishable*[1] if Consumer has a winning strategy in the game $[A]$. We call a formula A *computably accomplishable* if Consumer has a computable winning strategy in the game $[A]$.

If A has no exponential connectives then the game $[A]$ is essentially finite (every player can perform only finitely many actions) and thus the game $[A]$ is accomplishable iff it is computably accomplishable. In this case at least one of the players has a (computable) winning strategy.

In the general case it is unknown whether accomplishability is equivalent to computable accomplishability.

Note that the rules of the game favour Producer. For example, it might happen that Producer has a winning strategy in both games A and $\neg A$. This happens, say for $A = x$.

The simplest accomplishable formula is $x \vee \neg x$: in order to win Consumer just moves Producer's coin to her box.

[1] We use here the terminology of [4].

Important remark. It is important that Consumer cannot distinguish fake and genuine coins. (Formally, that means that we choose coin evaluations after the play is finished.) That is why the formula $A = (\neg x \wedge \neg x) \vee x$, which is not derivable in the affine logic, is not accomplishable. In the game $[A]$ Consumer has essentially two strategies: (1) she moves the first of Producer's coins (2) she moves the second one. Both strategies do not win. Indeed, if only one coin is genuine, namely, the coin that has not been moved, Consumer looses the formula A.

The definition of an accomplishable formula is very robust: we can change the definition of the game in many ways so that the class of (computably) accomplishable formulas does not change. Below we present several such modifications.

1. We can forbid Consumer (or Producer, or both) to perform several actions in one move. Indeed, postponing actions never hurts the player. By the same reason it does not matter who starts the play.[2]

2. We can ban all actions inside occurrences B and C belonging to an occurrence of $B \sqcup C$ [or $B \sqcap C$] such the choice action has not yet been applied to $B \sqcup C$ [$B \sqcap C$, respectively].

3. We can assume that all the Producers boxes are empty at the start of the play and he is allowed to perform an action "deposit a coin in a box". Consumer is allowed to apply action "allocate U to V" only when U is not empty. At the end of a play a Producer's box is won by him iff he has deposited a genuine coin in it.

4. We can define the game $[A]$ recursively. To this end we need a notion of a "game with coins" and operations on such games that correspond to connectives.

Games of the form $[A]$ can be regarded as vending machines: Consumer's boxes can be identified with slots for coins and Producer's boxes with compartments for releasing the products. For instance, a vending machine that accepts 1 Euro coin and 50 cents coins and sells coffee and tee, for 1 Euro each, can be represented by the formula

$$!(\neg(1\text{ Euro} \sqcup (50\text{ cents} \wedge 50\text{ cents})) \vee (\text{tee} \sqcap \text{coffee})).$$

Informally, the game $[A]$ is accomplishable iff the vending machine can work without having any resources in advance.

Historical remarks. As we have said, the notion of an accomplishable formula was defined in [4] (Logic of Tasks semantics). That paper also provides a sound and complete calculus for additive fragment of accomplishable formulas. For the multiplicative fragment an equivalent semantics was defined in [8] (Abstract Resource Semantics) together with a sound a complete calculus (CL5). Finally, the calculus CL4 from [8] provides a sound and complete axiomatisation for the set of all accomplishable formulas that have no exponential connectives. It is unknown whether the entire set of accomplishable formulas is computably enumerable. The similar question is open for computably accomplishable formulas as well.

[2] The games having that property are called static, see e.g. [5]. We will consider other static games in the next section.

1.2 Accomplishable Formulas and Affine Logic

We will use the following variant of affine logic. A *sequent* is a finite list of formulas. A sequent consisting of formulas $A_1, \ldots, A_n$ is denoted by $\vdash A_1, \ldots, A_n$. The order of formulas in a sequent does not matter, but the multiplicity of each formula matters. That is, $\vdash p, p$ and $\vdash p$ are different sequents.

Axioms: $\vdash A, \neg A, \quad \vdash \mathbf{1}$.

Derivation rules:

$\frac{\vdash \Gamma, A \quad \vdash \Delta, \neg A}{\vdash \Gamma, \Delta}$ (Cut). $\frac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, (A \wedge B)}$ (Introducing $\wedge$).

$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, (A \vee B)}$ (Introducing $\vee$). $\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, (A \sqcap B)}$ (Introducing $\sqcap$).

$\frac{\vdash \Gamma, A}{\vdash \Gamma, (A \sqcup B)}$, $\frac{\vdash \Gamma, B}{\vdash \Gamma, (A \sqcup B)}$ (Introducing $\sqcup$). $\frac{\vdash \Gamma}{\vdash \Gamma, A}$ (Weakening).

$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?\, A}$ (Dereliction). $\frac{\vdash \Gamma, ?\, A, ?\, A}{\vdash \Gamma, ?\, A}$ (Contraction). $\frac{\vdash ?\, \Gamma, A}{\vdash ?\, \Gamma, !\, A}$ (R).

Here $?\,\Gamma$ denotes the list obtained from Γ by prefixing by ? all formulas in the list.

Call the formula $A_1 \vee \ldots \vee A_n$ the *formula image* of the sequent $\vdash A_1, \ldots, A_n$. If the list is empty (that is, $n = 0$), its formula image is equal to $\mathbf{0}$. A sequent is called *(computably) accomplishable* if so is its formula image.

Theorem 1. *The set of accomplishable sequents contains all axioms of the affine logic and is closed under all its derivation rules and under the substitution. The same applies to computable accomplishable formulas. (Hence all derivable formulas are computably accomplishable.)*

The affine logic is not complete with respect to accomplishability semantics. An example of an accomplishable non-provable formula is

$$[(\neg a \vee \neg b) \wedge (\neg c \vee \neg d)] \vee [(a \vee c) \wedge (b \vee d)].$$

(This formula was used by Blass in [1] to show that the affine logic is incomplete with respect to his semantics.)

1.3 Accomplishable Formulas and the Intuitionistic Propositional Calculus IPC

Consider the Girard's translation from the language of propositional formulas with connectives $\wedge, \vee, \rightarrow, \bot$ into the language of affine logic. Each formula A is assigned a formula A^* defined recursively: $(x_i)^* = x_i$ and

$$(A \vee B)^* = !\, A^* \sqcup !\, B^*, \quad (A \wedge B)^* = A^* \sqcap B^*,$$
$$(A \rightarrow B)^* = \neg(!\, A^*) \vee B^* = ?(\neg A^*) \vee B^*, \quad \bot^* = \mathbf{0}.$$

This translation preserves provability. The next lemma shows that the combination of this translation with accomplishability semantics yields a sound semantics for the intuitionistic calculus.

Lemma 1. *The set L of all formulas A whose translation A^* is accomplishable is a super-intuitionistic logic (that is, L contains all axioms of IPC and is closed under Modus Ponens and substitution). The same holds for the set of all formulas whose translation is computable accomplishable.*

It turns out that this semantics of IPC is complete for the positive fragment of IPC.

Theorem 2. *If A is a positive formula (that is, it does not contain $\bot$) and A^* is accomplishable then A is derivable in IPC.*

One of the technical tools in the proof of this theorem is the following theorem, which was stated by Medvedev in [11] (without a complete proof) and proved in [2]. A *critical implication* is a (positive) formula of intuitionistic language of the form $A_1 \wedge \cdots \wedge A_m \to R$, where R is an OR of variables and each A_i has the form $(P_i \to Q_i) \to Q_i$. Here every P_i is an AND of variables and Q_i is an OR of variables and for all i the formulas P_i and Q_i have disjoint sets of variables. The number of variables in R, P_i and Q_i is positive and may be equal to 1.

Theorem 3 ([11,2]). *If a super-intuitionistic logic contains no critical implication then its positive fragment coincides with the positive fragment of IPC. Moreover, if a positive formula A is not derivable in IPC then there is a critical implication J such that $IPC \vdash (A' \to J)$ where A' is a formula obtained from A by a substituting certain formulas of type OR-of-AND-of-variables for A's variables.*

Theorem 2 does not generalise to negative formulas. We know two examples of a formula A such that A^* is accomplishable but A is not provable in IPC: Rose formula from [12] and Japaridze's formula from [7]. The latter one is simple enough to present it here:

$$(\neg p \to x \vee y) \wedge (\neg\neg p \to x \vee y) \to (\neg p \to x) \vee (\neg p \to y) \vee (\neg\neg p \to x) \vee (\neg\neg p \to y). \tag{1}$$

Here $\neg B$ is an abbreviation for $B \to \bot$.

Note that if we defined the translation of $A \vee B$ as just $A^* \sqcup B^*$ then the translation of the axiom

$$(x \to r) \to ((y \to r) \to (x \vee y \to r))$$

would not be accomplishable.

2 Japaridze's Game Semantics and Accomplishability

2.1 Static Games

We will use a rather general notion of games from [5] between two players, called Environment and Machine.

A *move* is a string over the keyboard alphabet. A *labelled move (labmove)* is a move prefixed by E or M (the prefix indicates who has done the move, Environment or Machine). A *run* is a finite or infinite sequence of labmoves. A *position* is a finite run.

A *game*[3] is specified by a set of runs L and a function W mapping runs to the set $\{\mathtt{E},\mathtt{M}\}$. All runs in L are called *legal*, all other runs are called *illegal*. If $W(\Gamma) = P$, we say that the run Γ is *won* by P. Otherwise it is *lost* by P.

The set L must have the following properties: (1) the empty sequence (called the *initial position*) belongs to L and (2) if a (finite or infinite) run Γ is in L then all its finite prefixes are in L too.

Let $\Gamma = \alpha_1, \alpha_2, \ldots$ be a run and α_i a labmove in that run. We say that α_i is the first *first illegal* labmove in Γ if $\alpha_1, \ldots, \alpha_{i-1}$ is legal run but $\alpha_1, \ldots, \alpha_i$ is not. Each illegal run Γ has exactly one first illegal labmove. The function W must have the following property: every illegal run is lost by that Player who has made the first illegal move in it.

We have not yet defined how to play a game. It is not obvious since in some positions both players can make a legal move and the rules do not specify who has the turn to play in such a position.

There are eight ways to play a game. We have to make three choices: who starts the game, how many moves (at most one or several) is Environment allowed to make in its turn, and how many moves (at most one or several) is Machine allowed to make in its turn. For example, the game can be played as follows: Environment starts the play; in its turn, each player either makes a move or passes. Another way: Machine starts the play; in its turn, Environment can make any finite sequence of moves (including the empty sequence); in its turn, Machine can make a move or pass. For all ways, we assume that the game lasts infinitely long and the turn to play alternates.

For certain games it is crucial which of the eight modes to play is chosen (for example, if $W(\Pi) = \mathtt{E}$ if Π starts with a move of E and $W(\Pi) = \mathtt{M}$ otherwise). There are however two important classes of games, *strict* games and more general *static* games, for which it does not matter.

A game is called *strict* if for every legal position Δ at most one player can play a move α so that the resulting position Δ, α is legal.

Most games considered in the literature are strict ones. However, the operation on games we are going to define do not look natural when applied to strict games. They look natural when applied to another type of games (called static games) defined in the next two paragraphs. Informally, static games are those games in which it never hurts the player to postpone moves.

Let Γ, Δ be (finite or infinite) runs. We say that Δ is a *Machine-delay* of Γ if Δ is obtained from Γ by postponing certain Machine's moves (may be infinitely many). Formally, the following conditions should hold. (1) Erasing all moves of Environment in Γ and Δ results in the same run; the same holds for erasing Machine's moves. (2) For all k and l if kth move of Machine is made later than lth move of Environment in Γ then so is in Δ.

[3] called a *constant game* in [5].

We define a notion of *Environment-delay* in a similar way. We say that the game is *static* if the following holds for every player P, every run Γ and every P-delay Δ of Γ. (1) If P has not made the first illegal move in Γ then P has not made the first illegal move in Δ either. (2) If Γ is won by P then so is Δ.

We call a static game *winnable* if Machine has a winning strategy in the game. It does not matter which of the above eight ways to play the game to choose: the class of winnable games is robust under switching between the eight playing modes. However, it is important that we do not force any player to make a move in its turn. We call a static game *computably winnable* if Machine has a computable winning strategy in the game (that is, there is a Turing machine that wins the game).

Now we will define operations $\neg, \wedge, \vee, \sqcup, \sqcap, ?, !$ on games. Those operation will preserve static property. We will then call a formula of affine language (computably) winnable if every substitution of static games for variables results in a (computably) winnable game. One of our main result states that a formula is (computably) winnable iff it is (computably) accomplishable.

2.2 Operations on Games

The operation of *negation* $\neg$ just swaps the roles of players: Machine plays in Environment's role and vice versa. The set of legal runs of $\neg A$ is obtained from that of A by replacing each run by its dual (a run Γ' is dual to Γ if it is obtained from Γ by exchanging labels $\mathtt{E}$ and $\mathtt{M}$ in all labmoves). Machine wins a run Γ in $\neg A$ iff the dual run Γ' is won by Environment in A.

The *choice conjunction* applied to games A, B produces the following game $A \sqcap B$. Environment decides between A and B. Then the chosen game is played. If Environment has not decided, it looses. Formally a non-empty run is legal iff it starts with Environment's move "choose left" of "choose right" and the rest of the run is the legal run of A if the first move is "choose left" and is the legal run of B if the first move is "choose right". A legal run is won by Machine in the following three cases: (1) it is empty, (2) the first move is "choose left" and the rest of the run is won by Machine in the game A, and (3) the first move is "choose right" and the rest of the run is won by Machine in the game B.

The *choice disjunction* $A \sqcup B$ of A, B is dual to $A \sqcap B$. This time Machine has to decide between A and B (and it looses if it has not decided). In other words, $A \sqcup B = \neg(\neg A \sqcap \neg B)$.

Parallel disjunction $\vee$. In the game $A \vee B$ the players play two games A and B simultaneously. In order to win, Machine has to win at least one game. Formally, a run Γ is legal if the following holds. Let Γ^i denote the result of removing from Γ all the labmoves that do not have the form $Pi.\alpha$ (where $P = \mathtt{E}, \mathtt{M}$) and replacing the prefix $Pi.\alpha$ by $P\alpha$ in all the remaining labmoves. A run Γ is legal if every its labmove has the form $Pi.\alpha$ (where $i = 1, 2$) and the runs Γ^1, Γ^2 are legal runs of A, B, respectively. Such a run is is won by Machine if either Γ^1 is won by Machine in A or Γ^2 is won by Machine in B (or both).

Parallel conjunction $\wedge$ is dual to parallel disjunction. The difference is that this time Machine has to win both games. In other words, $A \wedge B = \neg(\neg A \vee \neg B)$.

Parallel recurrence $\curlywedge$. The game $\curlywedge A$ is essentially an infinite parallel conjunction $A \wedge A \wedge A \wedge \ldots$. Players play simultaneously infinite plays and in order to win Machine has to win all plays. Formally, a run is legal if all its moves have the form $Pi.\alpha$ where $i = 1, 2, 3, \ldots$ (we spell natural numbers in decimal notation, say) and all $\Gamma^1, \Gamma^2, \Gamma^3, \ldots$ are legal runs of A. Such a run is won by Machine if all Γ_i are won by Machine in the game A.

Parallel corecurrence $\curlyvee$ is dual to parallel recurrence $\curlywedge$. Again players play infinitely many plays in the game A. However, this time in order to win, Machine has to win at least one play. That is, $\curlyvee A = \neg(\curlywedge \neg A)$.

All these operations preserve the static property. However not all of them preserve strictness property. Consider, for example, the multiplicative conjunction. Assume that the games A and B are strict, the first move in A is made by Environment and the first move in B is made by Machine. Then the first move in $A \wedge B$ can be made by either player.

Nevertheless all eight ways to play $A \wedge B$ are equivalent (for all strict games A, B). So we could fix any of the eight ways to play and thus convert $A \wedge B$ into an equivalent strict game. However such stipulating would be quite unnatural (bureaucratic, using Japaridze's word). As a result, we would not have the property $A \wedge B = \neg(\neg A \vee \neg B)$. The games $A \wedge B$ and $\neg(\neg A \vee \neg B)$ would be only equivalent in a sense.

Another reason to prefer static games is that all the computational problems can be quite naturally expressed as static games and not as strict games (see [6] for many examples). That is why Japaridze has chosen static games as a basis for his Game semantics.

2.3 Winnable = Accomplishable

Let $A(x_1, \ldots, x_n)$ be a formula of affine language and $G_1, \ldots, G_n$ static games. Substituting G_i for x_i in A and performing all operations spelled in A we obtain a static game $A(G_1, \ldots, G_n)$. Exponential AND and OR are interpreted as $\curlywedge, \curlyvee$ respectively. We say that a formula A is *winnable*, if for all static games $G_1, \ldots, G_n$ the resulting game $A(G_1, \ldots, G_n)$ is winnable. We say that A is *computably winnable*, if for all static games $G_1, \ldots, G_n$ the game $A(G_1, \ldots, G_n)$ is computably winnable.

Consider also the uniform version of winnability. Call a formula A is *uniformly* (computably) winnable, if there is a (computable) strategy winning the game $A(G_1, \ldots, G_n)$ for all static games $G_1, \ldots, G_n$. We will see that winnability coincides with accomplishability and computable winnability coincides with computable accomplishability.

Note that in the definition of winnability we do not require the games $G_1, \ldots, G_n$ be determined.[4] Why? The class of determined games is closed under all operations considered. And who wins the game is determined just classically: Machine wins (i.e., it has a winning strategy in) $A \wedge B$ iff it wins A and wins B,

[4] The game is called *determined* in either Machine, or Environment has a winning strategy in the game.

Machine wins $A \vee B$ if it wins A or wins B etc. Thus if we restrict the class of static games by determined ones, a formula would be winnable iff it is a classical tautology.

For uniform winnability and computable winnability this is not the case: for example, it might be that Machine has a computable winning strategy in the game $A \vee B$ but does not have computable winning strategy in either A or B. Therefore for these versions of winnability it seems quite natural to restrict the class of games to determined ones. However, it turns out that this restriction does not affect the classes of uniformly winnable, computably winnable and uniformly computably winnable formulas.

Theorem 4. *The following three properties of a formula A are equivalent:*
(1) A is computably accomplishable,
(2) A is uniformly computably winnable, and
(3) For every 2-moves[5] *(hence determined) static games $G_1, \ldots, G_n$ the game $A(G_1, \ldots, G_n)$ is computably winnable.*

The equivalence of (2) and (3) implies that every winnable formula is uniformly computably winnable. This proves Conjecture 26.1 from [5] (a similar statement for branching recurrences follows from Theorem 6 below).

Theorem 5. *The following four properties of a formula A are equivalent:*
(1) A is accomplishable.
(2) A is uniformly winnable.
(3) A is winnable.
(4) There is a Machine's strategy winning every game of the form $A(G_1, \ldots, G_n)$, where $G_1, \ldots, G_n$ are 2-moves static games.

These theorems together with Theorem 2 and Lemma 1 show that winnability is a sound complete game theoretic semantics for the intuitionistic propositional calculus.

Corollary 1. *A positive formula is provable in IPC if and only if the formula A^* is computably winnable. The same holds for winnability, uniform winnability and uniform computable winnability.*

2.4 Countable Branching Recurrence

There is another way to interpret exponential connectives as operations on games, called *branching recurrence* and *corecurrence* in [5]. There are two versions of them, countable and uncountable. Countable branching recurrence and corecurrence were essentially introduced by Blass in [1].

The countable branching recurrence $\mathbin{⫝}^{\aleph_0}$ is the operation on games defined as follows. In the game $\mathbin{⫝}^{\aleph_0} A$ players play countably many plays in game A and Machine has to win all plays (like in the game $\mathbin{⅄} A$). However this time its job

[5] A game is a k-move game if every legal run has at most k labmoves.

is even more difficult, as Environment may "copy" positions. Informally, we can imagine that a position in the game $\mathbin{⫰}^{\aleph_0} A$ is a tuple $\langle p_1, \dots, p_n \rangle$ of positions of A. The initial position is the tuple $\langle p_1 \rangle$ where p_1 is the initial position in A. If the current position is $\langle p_1, \dots, p_n \rangle$, then each player is allowed to make a legal move in any of the positions $p_1, \dots, p_n$. Environment is also allowed to copy any p_i, in which case the new position is equal to $\langle p_1, \dots, p_n, p_i \rangle$.

This definition is very informal because we have defined moves in terms of positions and not the other way around, as our framework prescribes.

Moreover, the decsribed game may be not static.[6] To obtain an equivalent static game we need to understand a copying operation as splitting one position (the parent) into two new positions (the children). If a move made in a position P is postponed so that at the time when it is made the position P has been splitted, then that move is automatically played in all descendants of P. More specifically, we assign to each position an address, which is a binary string rather than a natural number. When a position with address w is splitted, the children positions receive addresses $w0$ and $w1$. When a play is finished we obtain a finite or infinite tree consisting of all addresses used. If that tree is finite then all its leaves are addresses of the played games. If the tree is infinite then some infinite paths (say $010101\dots$) are not addresses of "real" plays. Only those infinite paths are addresses of real plays which have finite number of 1s.

Formally, a run Γ is legal if it is a sequence of labmoves of the form $Pw.\alpha$ and $\mathtt{E}(\text{split } w)$ having the following two properties. (1) For every w and every proper prefix u of w every occurrence of a labmove of the form $Pw.\alpha$ or $\mathtt{E}(\text{split } w)$ is preceeded by exaclty one occurence of the labmove $\mathtt{E}(\text{split } u)$. (2) For every infinite string w that has only finitely many 1s consider the sequence $\Gamma(w)$ of all labmoves in Γ of the form $Pu.\alpha$—with "u." removed—where u is a prefix of w. For each w that has only finitely many 1s the run $\Gamma(w)$ must be a legal run of A. A legal run Γ is won by Machine if $\Gamma(w)$ is won by Machine for all w that has only finitely many 1s.

In the definition of *uncountable branching recurrence* $⫰$, we stipulate that Γ is won be Machine if $\Gamma(w)$ is won for all infinite w (and not only for w having finite number of 1s). Thus the difference between countable and uncountable versions is due to different understandings which plays are "real" and which are not.

Countable branching corecurrence $⫯^{\aleph_0}$ is defined in the dual way so that $⫯^{\aleph_0} A = \neg(⫰^{\aleph_0} \neg A)$.

Let us change the interpretation of ! and ? to $⫰^{\aleph_0}$ and $⫯^{\aleph_0}$, respectively. We obtain new notions of winnability and computable winnability, which does not coincide with the old ones (and hence differ from accomplishability). Indeed, it is not hard to see that the formula

$$?(\neg x \sqcap \neg y) \vee (!\, x \sqcup !\, y),$$

[6] Indeed, assume that the initial position is lost by Machine and every run of length 1 is won by Machine. Then the position $\mathtt{M1}.\alpha, \mathtt{E}(\text{copy the first position})$ is won by Machine but the position $\mathtt{E}(\text{copy the first position}), \mathtt{M1}.\alpha$ is lost.

is not accomplishable but is uniformly computably winnable provided exponentials are interpreted as countable branching recurrences.

However, Theorems 4 and 5 remain partially valid for the new notions of winnability.

Theorem 6. *The following two properties of a formula A are equivalent (if exponentials are interpreted as countable branching recurrences):*
(1) A is uniformly computably winnable, and
(2) For every 2-moves (hence determined) static games $G_1, \dots, G_n$ the game $A(G_1, \dots, G_n)$ is computably winnable.

Theorem 7. *The following three properties of a formula A are equivalent:*
(1) A is uniformly winnable.
(2) A is winnable.
(3) There is a Machine's strategy winning every game of the form $A(G_1, \dots, G_n)$, where $G_1, \dots, G_n$ are 2-moves static games.

Corollary 1 remains true for countable branching recurrences and, moreover, in the translation of $A \vee B$ we may omit !. More specifically, let $(x_i)^\dagger = x_i$ and

$$(A \vee B)^\dagger = A^\dagger \sqcup B^\dagger, \quad (A \wedge B)^\dagger = A^\dagger \sqcap B^\dagger,$$
$$(A \to B)^\dagger = \neg(!\, A^\dagger) \vee B^\dagger = ?(\neg A^\dagger) \vee B^\dagger, \quad \bot^\dagger = \mathbf{0}.$$

Theorem 8. *The set of all formulas A such that $A^\dagger$ is winnable is a superintuitionistic logic (if exponentials are interpreted as countable branching recurrences). The same holds for computable winnability. On the other hand, if a positive formula is not provable in IPC then the formula $A^\dagger$ is not winnable and there are 3-moves games such that the game $A^\dagger(G_1, \dots, G_n)$ is not computably winnable.*

Theorem 8 is true for the Girard's translation as well.

2.5 Historical and Terminological Remarks

The notions of accomplishability and computable accomplishability come back to [4]. Later they were extended to what has been termed *abstract resource semantics* in [8]. The notions of a computably winnable and uniformly computably winnable formula were defined in [5] under the name a *(uniformly) valid* formula. The uncomputable versions of these were also considered in [5], as a property of games rather than a property of formulas. The notion of a winnable formula (only for countable branching recurrence) was first considered by Blass in [1] (although Blass has considered only strict games and his definition of a countable branching recurrence differs in some technical details from the above definition, the class of winnable formulas is the same).

2.6 Uncountable Branching Recurrence

Theorem 8 is true for uncountable branching recurrence as well, which was shown in [10]). Our proof of Theorem 8 (based on Theorem 3) also works for uncountable branching recurrences and provides a shorter proof than that of [10].

3 Summary of Results

We have conidered the following four classes of formulas in the language of affine logic:

(1) accomplishable formulas,
(2) computably accomplishable formulas,
(3) winnable formulas for parallel recurrences,
(4) computably winnable formulas for parallel recurrences,
(5) winnable formulas for countable branching recurrences,
(6) computable winnable formulas for countable branching recurrences.

We have shown that (1)=(3) and (2)=(4). It is unknown whether (1)=(2) and (3)=(4). Both classes (1) and (2) are different from (3) and (4). All the classes (3)–(6) coincide with their uniform versions.

It is unknown whether the classes (1)–(4) are decidable or computably enumerable.

Both winnability and computable winnability (both parallel and branching versions) provide a sound semantics for the intuitionistic propositional calculus, which is complete for its positive fragment.

Acknowledgements

We are sincerely grateful to Giorgi Japaridze for explaining many subtle things about game operations, pointing to the relevant papers and many useful comments on previous versions of the paper. We are grateful to Alexander Shen and other participants of Kolmogorov seminar at Moscow State University for bringing our interest to game semantics of the intuitionistic logic.

References

1. Blass, A.: A game semantics for linear logic. Annals of Pure and Applied Logic 56, 183–220 (1992)
2. Chernov, A.V., Skvortsov, D.P., Skvortsova, E.Z., Vereshchagin, N.K.: Variants of Realisability for Propositional Formulas and the Logic of the Weak Law of Excluded Middle. In: Bradfield, J.C. (ed.) CSL 2002 and EACSL 2002. LNCS, vol. 2471, pp. 74–88. Springer, Heidelberg (2002)
3. Girard, G.Y.: Linear logic. Theoretical Computer Science 50(1), 1–102 (1987)
4. Japaridze, G.: The logic of tasks. Annals of Pure and Applied Logic 117, 263–295 (2002)
5. Japaridze, G.: Introduction to computability logic. Annals of Pure and Applied Logic 123, 1–99 (2003)
6. Japaridze, G.: In the beginning was game semantics, arXiv:cs.LO/0507045, 111 pages (2005)
7. Japaridze, G.: A letter to N.V. (04.11.2005)
8. Japaridze, G.: Introduction to Cirquent Calculus and Abstract Resource Semantics. Journal of Logic and Computation 16(4), 489–532 (2006)

9. Japaridze, G.: Intuitionistic computability logic. Acta Cybernetica 18(1), 77–113 (2007)
10. Japaridze, G.: The intuitionistic fragment of computability logic at the propositional level. Annals of Pure and Applied Logic 147(3), 187–227 (2007)
11. Medvedev, Y.T.: Finite problems. Doklady Akad. Nauk SSSR 3, 227–230 (1962)
12. Rose, G.F.: Propositional calculus and realizability. Transactions of the American Mathematical Society 75(1), 1–19 (1953)

The Grammar of Scope

Mark Steedman

School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW
Scotland, United Kingdom
steedman@inf.ed.ac.uk
http://www.inf.ed.ac.uk/~steedman

The program of research that seeks a "Natural Logic" to which the forms of natural language are transparent has been frustrated by the existence of ambiguities of scope in interpretations for multiply quantified sentences, which appear to require grammatical operations that compromise the strong assumptions of syntactic/semantic transparency and monotonicity made under that program. Examples of such operations include covert movement at the level of logical form, abstraction or storage mechanisms, and proliferating type-changing operations.

The paper examines some interactions of scope alternation with syntactic phenomena including coordination, binding, and relativization. Starting from the assumption of Fodor and Sag, and others, that many expressions that have been treated as generalized quantifiers are in reality non-quantificational, expressions, and using Combinatory Categorial Grammar (CCG) as a grammatical framework, the paper presents an account of quantifier scope ambiguities according to which the available readings are projected directly from the lexicon by the combinatorics of the syntactic derivation, without any independent manipulation of logical form and without recourse to syntactically unmotivated type-changing operations.

The logic that results has a number of surprising features. It makes extensive use of (generalized) Skolem terms. The treatment of negation is unusual. The Philonian identification of the conditional with material implication is avoided. Some implications for natural language processing are considered.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, p. 43, 2008.

Conjunctive Grammars and Alternating Pushdown Automata
(Extended Abstract)

Tamar Aizikowitz and Michael Kaminski

Department of Computer Science
Technion – Israel Institute of Technology
{tamarai,kaminski}@cs.technion.ac.il

Abstract. In this paper we introduce a variant of alternating pushdown automata, *Synchronized Alternating Pushdown Automata*, which accept the same class of languages as those generated by conjunctive grammars.

1 Introduction

Many well known computational models support non-deterministic computations with existential acceptance conditions, thereby leading to an inherent disjunctive quality of the class of languages accepted. When looking at the dual form of these models (e.g., Co-NP), universal acceptance conditions lead to conjunction: all computations must accept. This type of acceptance is useful in such fields as concurrent programming where *all* processes must meet correctness demands. In this paper we explore several extensions of models for context-free languages which combine both the notion of conjunction and of disjunction, leading to a richer set of languages.

Conjunctive Grammars (CG) are an example of such a model. Introduced by Okhotin in [1], CG are a generalization of context-free grammars. Explicit intersection operations are allowed in rules thereby adding the power of conjunction. CG were shown by Okhotin to accept all finite conjunctions of context-free languages, as well as some additional languages. However, there is no known non-trivial technique to prove a language cannot be derived by a CG, so their exact placing in the Chomsky hierarchy is unknown. Okhotin proved the languages generated by these grammars to be polynomially parsable [1,2], making the model practical from a computational standpoint, and therefore of interest for applications in various fields such as programming languages, etc. In this paper we introduce a new model of alternating automata, *Synchronized Alternating Pushdown Automata* (SAPDA), which is equivalent to the Conjunctive Grammar model.[1]

The concept of alternating automata models was first introduced by Chandra et.al. in [3]. In these models, computations alternate between existential and universal modes of acceptance, hence their name. This behavior is achieved

[1] We call two models equivalent if they accept/generate the same class of languages.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 44–55, 2008.

by splitting the state-set into two disjoint groups, existential states and universal states. The acceptance model dictates that all possible computations from universal states must be accepting whereas only one must be accepting from existential states. Thus, for a word to be accepted it must meet both disjunctive and conjunctive conditions. In the case of Alternating Finite State Automata and Alternating Turing Machines, the alternating models have been shown to be equivalent in expressive power to their non-alternating counterparts, see [3].

Alternating Pushdown Automata (APDA) were also introduced in [3] and were further explored in [4]. Like Conjunctive Grammars, APDA add the power of conjunction over context-free languages. Therefore, unlike Finite Automata and Turing Machines, here alternation increases the expressiveness of the model. In fact, APDA accept exactly the exponential-time languages, see [3,4].

It is well known that Context-Free Grammars and Pushdown Automata are equivalent, e.g., see [5, pp. 115–119]. Yet, the APDA model is stronger than the CG model [1]. Our *Synchronized Alternating Pushdown Automata* are weaker than general APDA, and accept exactly the class of languages derived by Okhotin's Conjunctive Grammars. Okhotin showed in [6,7] that Linear Conjunctive Grammars, a subfamily of the Conjunctive Grammars, are equivalent to Trellis Automata [8], however SAPDA are the first class of automata shown to be equivalent to general CG.

The paper is organized as follows. In Section 2 we define the Conjunctive Grammar model. In Section 3 we introduce our SAPDA model. Section 4 details our main results, namely the equivalence of the CG and SAPDA models. Sections 5 and 6 contain discussions of mildly context-sensitive languages and related work respectively, and Section 7 is a short conclusion of our work.

2 Conjunctive Grammars

The following definitions are taken from [1].

Definition 1. *A* Conjunctive Grammar *is a quadruple* $G = (V, \Sigma, P, S)$ *where:*

1. V, Σ *are disjoint finite sets of non-terminal and terminal symbols respectively.*
2. $S \in V$ *is the designated start symbol.*
3. P *is a finite set of rules of the form* $A \to (\alpha_1 \& \ldots \& \alpha_n)$ *s.t.* $A \in V$ *and* $\alpha_i \in (V \cup \Sigma)^*$. *If* $n = 1$ *then we write* $A \to \alpha$.

Definition 2. Conjunctive Formulas *are defined over the alphabet* $V \cup \Sigma \cup \{(,), \&\}$. *The set of conjunctive formulas corresponding to a grammar* G *is defined as follows:*

1. ϵ *is a formula.*
2. *Every symbol in* $V \cup \Sigma$ *is a formula.*
3. *If* $\mathcal{A}$ *and* $\mathcal{B}$ *are formulas then* $\mathcal{AB}$ *is a formula.*
4. *If* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *are formulas then* $(\mathcal{A}_1 \& \cdots \& \mathcal{A}_n)$ *is a formula.*

Notation. Below we use the following notation: σ, τ denote terminal symbols, u, w, y denote terminal words, X, Y denote non-terminal symbols, α, β denote non-terminal words, and $\mathcal{A}, \mathcal{B}$ denote conjunctive formulas.

Definition 3. *For every conjunctive formula* $\mathcal{A} = (\mathcal{B}_1 \& \ldots \& \mathcal{B}_n)$, $\mathcal{B}_i s$, $i = 1, \ldots, n$, *are called are called* conjuncts *of* $\mathcal{A}$,[2] *and* $\mathcal{A}$ *is called the* enclosing formula. *If* $\mathcal{B}_i$ *contains no* &*-s, then* $\mathcal{B}_i$ *is a* simple conjunct.

Definition 4. *Given a grammar* G, *the relation of* immediate derivability *on the set of conjunctive formulas,* $\Rightarrow_G$, *is defined as follows:*

1. $s_1 X s_2 \Rightarrow_G s_1(\alpha_1 \& \cdots \& \alpha_n) s_2$ *for all* $X \to (\alpha_1 \& \cdots \& \alpha_n) \in P$
2. $s_1(w \& \cdots \& w) s_2 \Rightarrow_G s_1 w s_2$ *for all* $w \in T^*$

where $s_i \in (V \cup T \cup \{(,), \&\})^*$. *As usual,* $\Rightarrow_G^*$ *is the reflexive transitive closure of* $\Rightarrow_G$, *and the language of a grammar* G *is defined as* $L(G) = \{w \in T^* | S \Rightarrow_G^* w\}$. *We refer to (1) as* production *rules and to (2) as* contraction *rules.*

Informally, a terminal word w is derived from a formula $(\mathcal{A}_1 \& \cdots \& \mathcal{A}_n)$ if and only if it is derived from each $\mathcal{A}_i$.

Example 1. The following conjunctive grammar derives the non-context-free *multiple-agreement* language $\{a^n b^n c^n | n \in \mathbb{N}\}$. $G = (V, T, P, S)$ where:

- $V = \{S, A, B, C, X, Y\}$, $T = \{a, b, c\}$,
- P contains the following derivation rules:

$$\begin{aligned} S &\to (C \ \& \ A) \\ C &\to Cc \mid X \ ; \ A \to aA \mid Y \\ X &\to aXb \mid \epsilon \ ; \ Y \to bYc \mid \epsilon \end{aligned}$$

The derivation of the word *aabbcc* is as follows:

$$\begin{aligned} S &\Rightarrow (C \ \& \ A) \Rightarrow (Cc \ \& \ A) \Rightarrow (Ccc \ \& \ A) \Rightarrow (Xcc \ \& \ A) \\ &\Rightarrow (aXbcc \ \& \ A) \Rightarrow (aaXbbcc \ \& \ A) \Rightarrow (aabbcc \ \& \ A) \\ &\Rightarrow (aabbcc \ \& \ aA) \Rightarrow (aabbcc \ \& \ aaA) \Rightarrow (aabbcc \ \& \ aaY) \\ &\Rightarrow (aabbcc \ \& \ aabYc) \Rightarrow (aabbcc \ \& \ aabbYcc) \Rightarrow (aabbcc \ \& \ aabbcc) \Rightarrow aabbcc \end{aligned}$$

Example 2. The following linear conjunctive grammar derives the non-context-free *cross-agreement* language $\{a^n b^m c^n d^m | n, m \in \mathbb{N}\}$. $G = (V, T, P, S)$ where:

- $V = \{S, A, B, C, D, X, Y\}$, $T = \{a, b, c, d\}$,
- P contains the following derivation rules:

$$\begin{aligned} S &\to (A \ \& \ D) \\ A &\to aA \mid X \ ; \ D \to Dd \mid Y \\ X &\to bXd \mid C \ ; \ Y \to aYc \mid B \\ C &\to cC \mid \epsilon \ ; \ B \to bB \mid \epsilon \end{aligned}$$

[2] Note that this definition is different from Okhotin's definition in [1].

The non-terminal A derives words of the form $a^i b^m c^j d^m$ for $m, i, j \in \mathbb{N}$, while D derives words of the form $a^n b^i c^n d^j$ for $n, i, j \in \mathbb{N}$. Therefore, the conjunction of the two generates the cross-agreement language.

Example 3. The following linear conjunctive grammar, taken from [1] derives the non-context-free *reduplication* language with a center marker $\{wcw \mid w \in \{a,b\}^*\}$. $G = (V, T, P, S)$ where:

- $V = \{S, A, B, C, D, E\}$, $T = \{a, b, c\}$,
- P contains the following derivation rules:

$$S \rightarrow (C\&D) \quad ; \quad C \rightarrow aCa \mid aCb \mid bCa \mid bCb \mid c$$
$$D \rightarrow (aA\&aD) \mid (bB\&bD) \mid cE \quad ; \quad A \rightarrow aAa \mid aAb \mid bAa \mid bAb \mid cEa$$
$$B \rightarrow aBa \mid aBb \mid bBa \mid bBb \mid cEb \quad ; \quad E \rightarrow aE \mid bE \mid \epsilon$$

The non-terminal C verifies that the lengths of the words before and after the c marker are equal, while D validates that the letters are the same. For a more detailed description see [1].

2.1 Linear Conjunctive Grammars

Okhotin defines in [1] a sub-family of conjunctive grammars called *Linear Conjunctive Grammars* (LCG). The definition is analogues to the definition of linear grammars as a sub-family of context-free grammars. LCG are an interesting sub-family of CG as they have particularly efficient parsing algorithms [6], making them practical from a computational standpoint. Okhotin proved in [7] that LCGs are equivalent to Trellis Automata.

Definition 5. *A conjunctive grammar $G = (V, T, P, S)$ is said to be* linear *if all rules in P are of the forms:*

- $X \rightarrow (u_1 Y_1 v_1 \ \& \cdots \& \ u_n Y_n v_n)$; $u_i, v_i \in T^*$, $X, Y_i \in V$
- $X \rightarrow w$; $w \in T^*$, $X \in V$

The grammars presented in Examples 1, 2, 3 are all linear.

3 Synchronized Alternating Pushdown Automata

We define a class of automata called *Synchronized Alternating Pushdown Automata* (SAPDA) as a variation on the standard PDA model. Similarly to standard Alternating Pushdown Automata [3,4], SAPDA have both the power of existential and universal choice.

We use a different (equivalent) definition from the existential and universal state-sets one presented in [3]. Instead, transitions are made to a conjunction of states. The model is non-deterministic, therefore several different conjunctions may be possible from a given configuration. If all conjunctions are of one state only, the automaton is a standard PDA.

The stack memory of an SAPDA is a tree. Each leaf has a processing head which reads the input and writes to its branch independently. When a multiple-state conjunctive transition is applied, the stack branch splits into multiple branches, one for each conjunct.[3] The branches process the input independently, however sibling branches must empty synchronously, after which the computation continues from the parent branch.

Definition 6. *A* synchronized alternating pushdown automaton *is a tuple* $A = (Q, \Sigma, \Gamma, \delta, q_0, \bot)$ *where the domain of* δ *is* $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$*. For every such* (q, σ, X)*,* δ *is a finite subset of*

$$\{(q_1, \alpha_1) \wedge \cdots \wedge (q_n, \alpha_n) \mid q_i \in Q,\ \alpha_i \in \Gamma^*,\ n \in \mathbb{N}\}\ .$$

Everything else is defined as in the standard PDA model; Q *is a finite set of states,* Γ, Σ *are the stack and input alphabets respectively,* $q_0 \in Q$ *is the initial state and* $\bot \in \Gamma$ *is the initial stack symbol, see, e.g., [5, pp. 107–112].*

We describe the current state of the automaton as a labelled tree. The tree encodes the stack contents, the current states of the stack-branches, and the remaining input to be read for each stack-branch. States and remaining inputs are saved in leaves only, as these encode the stack-branches currently processed.

Definition 7. *A* configuration *of an SAPDA is a labelled tree. Each internal node is labelled* $\alpha \in \Gamma^*$ *denoting the stack-branch contents, and each leaf node is labelled* (q, w, α) *where* $q \in Q$ *denotes the current state,* $w \in \Sigma^*$ *denotes the remaining input to be read and* $\alpha \in \Gamma^*$ *denotes the stack-branch contents.*

For a node v in a configuration T, we denote the label of v in T by $T(v)$. If a configuration has a single node only, it is denoted by the label of that node. For example, if a configuration T has a single node labelled (q, w, α) then T is denoted by (q, w, α).

At each computation step, a transition is applied to one stack-branch. If a branch empties, it cannot be chosen for the next transition (because it has no top symbol). If all sibling branches are empty, and each branch emptied with the *same* remaining input (i.e., after processing the same portion of the input) and with the same state, the branches are collapsed back to the parent branch.

Definition 8. *Let* A *be an SAPDA and let* T*,* T' *be two configurations of* A*. We write* $T \vdash_A T'$ *(*A *is omitted if understood from the context), if:*

- *There exists a leaf node* v *in* T *s.t.* $T(v) = (q, \sigma w, X\alpha)$ *and a transition* $(q_1, \alpha_1) \wedge \cdots \wedge (q_k, \alpha_k) \in \delta(q, \sigma, X)$ *s.t.:*
  - *If* $k = 1$ *then* T' *can be obtained from* T *by relabelling* v *s.t.* $T'(v) = (q_1, w, \alpha_1\alpha)$*.*

[3] This is similar to the concept of a transition from a universal state in the standard formulation of alternating automata, as all branches must accept.

  - *If $k > 1$ then T' can be obtained from T by relabelling v s.t. $T'(v) = \alpha$, and adding k child nodes to v, $v_1, \ldots, v_k$ s.t. $T'(v_j) = (q_j, w, \alpha_j)$ for $j = 1, \ldots, k$.*
- *There is a node v in T s.t. $T(v) = \alpha$, v has k children $v_1, \ldots, v_k$ s.t. all v_js, $j = 1, \ldots, k$, are leaves labelled with the same (p, w, ϵ), and T' can be obtained from T by removing nodes v_j and relabelling v s.t. $T'(v) = (p, w, \alpha)$.*

We denote by $T \vdash_A^* T'$ the reflexive transitive closure of $\vdash_A$.

Definition 9. *Let A be an SAPDA and let $w \in \Sigma^*$.*

- *An* initial configuration *of A on w is the configuration $(q_0, w, \bot)$.*
- *An* accepting configuration *of A is a configuration of the form (q, ϵ, ϵ).*
- *A* computation *of A on w is a series of configurations $T_0, \ldots, T_n$ where T_0 is the initial configuration, $T_{i-1} \vdash_A T_i$ for $i = 1, \ldots, n$, and all leaves v of T_n are labelled (q, ϵ, α), i.e., the entire input string has been read.*
- *An* accepting computation *of A on w is a computation where the final configuration T_n is accepting.*[4]

The language of A, denoted $L(A)$, is the set of all $w \in \Sigma^*$ s.t. A has an accepting computation on w.

Example 4. The SAPDA, $A = (Q, \Sigma, \Gamma, \delta, q_0, \bot)$, accepts the non-context-free language $\{w \mid \#_a(w) = \#_b(w) = \#_c(w)\}$ over $\Sigma = \{a, b, c\}$, where $Q = \{q_0, q_1, q_2\}$, $\Gamma = \{\bot, \bot_1, \bot_2, a, b, c\}$ and δ is defined as follows:

- $\delta(q_0, \epsilon, \bot) = \{(q_1, \bot_1) \wedge (q_2, \bot_2)\}$
- $\delta(q_i, \sigma, \bot_i) = \{(q_i, \sigma\bot_i)\}$, $(i, \sigma) \in \{1\} \times \{a, b\} \cup \{2\} \times \{b, c\}$
- $\delta(q_i, \sigma, \sigma) = \{(q_i, \sigma\sigma)\}$, $(i, \sigma) \in \{1\} \times \{a, b\} \cup \{2\} \times \{b, c\}$
- $\delta(q_1, \sigma_j, \sigma_k) = \{(q_1, \epsilon)\}$, $(\sigma_j, \sigma_k) \in \{(a, b), (b, a)\}$
- $\delta(q_2, \sigma_j, \sigma_k) = \{(q_2, \epsilon)\}$, $(\sigma_j, \sigma_k) \in \{(b, c), (c, b)\}$
- $\delta(q_i, \epsilon, \bot_i) = \{(q_0, \epsilon)\}$, $i \in \{1, 2\}$

The first step of the computation opens two branches, one for verifying that $\#_a = \#_b$ and one for verifying that $\#_b = \#_c$. If both branches manage to empty their stack then the word is accepted.

Figure 1 shows the contents of the stack-tree at an intermediate stage of a computation on the word *abbcccabb*. The left branch has read *abbcc* and shows that one more *b*-s than *a*-s have been read, while the right branch has read *abbccc* and shows that two more *c*-s than *b*-s have been read. Figure 2 shows the configuration describing the state of the automaton.

[4] Note that this is acceptance by empty stack. It is possible to define acceptance by accepting states. Let $F \subseteq Q$ be a set of accepting states. An accepting configuration is of the form (q, ϵ, α) where $q \in F$. Both models of acceptance are equivalent.

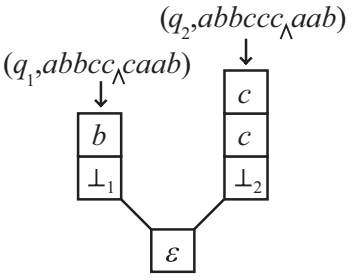


Fig. 1. Intermediate state of a computation on *abbcccaab*

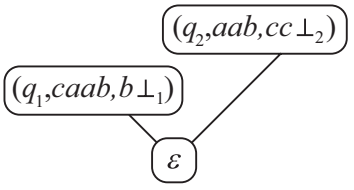


Fig. 2. The configuration matching the state in Fig. 1

4 Main Results

In this section we state the main results of our paper, namely that the SAPDA and CG models are equivalent.

Theorem 1. *If a language is generated by a CG then it is accepted by a SAPDA.*

Theorem 2. *If a language is accepted by an SAPDA then it is generated by a CG.*

The proofs of Theorems 1 and 2 are extensions of the classical ones,[5] see, e.g., [5, Theorem 5.3, pp. 115–116] and [5, Theorem 5.4, pp. 116–119] respectively. Both proofs are omitted due to lack of space.

5 Mildly Context-Sensitive Languages

The field of computational linguistics focuses on defining a computational model for natural languages. Originally, context-free languages were considered, and many natural language models are in fact models for context-free languages. However, certain natural language structures that cannot be expressed in context free languages, led to an interest in a slightly wider class of languages which came to be known as *mildly context-sensitive languages* (MCSL). Several formalisms for grammar specification are known to converge to this class [9].

Mildly context sensitive languages are loosely categorized as having the following properties: (1) They contain the context-free languages; (2) They contain such languages as multiple-agreement, cross-agreement and reduplication; (3) They are polynomially parsable; (4) They are semi-linear[6]. It is clear that there is a strong relation between the class of languages derived by conjunctive grammars (and accepted by SAPDA) and the class of mildly context sensitive languages. The first criterion of MCSL is obviously met, as both CG and SAPDA

[5] The proof of Theorem 1 is more involved, and requires several preliminary steps.

[6] A language L is *semi-linear* if $\{|w| \mid w \in L\}$ is a finite union of sets of integers of the form $\{l + im \mid i = 0, 1, \ldots\},\ l, m \geq 0$.

contain their context free counterparts, CFG and PDA respectively. The third criterion is also met by Okhotin's proof that CG membership is polynomial.

Multiple-agreement and cross-agreement are covered as shown in Examples 1, 2 respectively. Reduplication with a center marker is shown in Example 3. Okhotin has conjectured that reduplication without a center marker cannot be generated by any CG. However, this is still an open problem.

Surprisingly, it is the fourth criterion of semi-linearity which is not met, as demonstrated by the following example due to Okhotin [10].

Example 5. The following linear conjunctive grammar derives the non-context-free language $\{ba^2ba^4\cdots ba^{2^n}b | n \in \mathbb{N}\}$. $G = (V, T, P, S)$ where:

- $V = \{S, A, B, C, D, U, V\}$, $T = \{a, b\}$,
- P contains the following derivation rules:

$$S \rightarrow (U \ \& \ V) \mid b$$
$$U \rightarrow Ua \mid Ub \mid b \ ; \ V \rightarrow Ab \mid (B \ \& \ D)$$
$$A \rightarrow aA \mid a \ ; \ B \rightarrow Ba \mid Bb \mid Cb$$
$$C \rightarrow aCa \mid baa \ ; \ D \rightarrow aD \mid bV$$

For more details regarding this example see [10].

The language generated in Example 5 has super-linear growth, which means that, in this respect, CG and SAPDA accept some languages not characterized as mildly context-sensitive. In this respect, it may be that the CG and SAPDA models are too strong for natural language processing.

6 Related Work

6.1 Alternating Grammars

Moriya introduced in [11] the concept of Alternating Context-Free Grammars (ACFG), as a suggested grammatization for APDA. In ACFG, when a conjunctive rule is applied in a derivation, the currently derived formula is duplicated, and each duplicate continues its derivation independently. Therefore, a derivation of a grammar is in fact a tree where a conjunctive rule with k conjuncts yields k child nodes in the derivation tree. The root of the derivation tree is always labelled with the start symbol S. If all leaves of a derivation tree are labelled w then the tree is a derivation of w.

The difference between Moriya's ACFGs and Okhotin's CGs is that conjunctions in CGs are *local* leaving the rest of the thus far derived formula untouched whereas in ACFGs, when a conjunctive rule is applied, the entire formula is duplicated. It is the *locality* of conjunctions in CGs which renders them so similar in many respects to context-free grammars.

In [11], Moriya claimed that ACFG are equivalent to APDA. However, Ibarra et.al. showed in [12] that the equivalence proof was flawed. Namely, the proof was based on the claim that leftmost derivations of ACFGs are equivalent to general derivations. This claim, however, is surprisingly false. The question of

whether ACFG and APDA are equivalent remains an open one. It would seem that ACFG are stronger than CGs (as APDAs are stronger than SAPDAs) but this too has yet to be proven. Possibly the introduction of an automata model for CG can help solve some of these questions.

6.2 Conjunction in Lambek Categorial Grammars

Categorial Grammar is a formal system for analyzing the syntax and semantics of both formal and natural languages. Categorial grammars contain only a small set of universal rules, which are applicable to *all* languages; the differences between languages stemming solely from the lexicon. The universal rules of categorial grammars are treated as a logical calculus. Therefore, the syntactic analysis of an expression is reduced to a logical derivation. For more information see [13].

There are several frameworks for categorial grammars, each defining a set of universal derivation rules. One of the most widely accepted is the (associative) Lambek-calculus (**L**) as defined in [14]. Figure 3 shows the calculus, where Γ (Γ_i) denotes a finite sequence of categories, and c (c_i) denotes a single category. The expression $\Gamma \triangleright c$ is understood as: Γ is reducible to c in the given calculus.

$$(Ax) \quad c \triangleright c$$

$$\frac{\Gamma_1 \triangleright c_1 \qquad \Gamma_2\ c_2\ \Gamma_3 \triangleright c_3}{\Gamma_2\ \Gamma_1\ (c_1 \rightarrow c_2)\ \Gamma_3 \triangleright c_3}\ (\rightarrow \mathrm{L}) \qquad \frac{c_1\ \Gamma \triangleright c_2}{\Gamma \triangleright (c_1 \rightarrow c_2)}\ (\rightarrow \mathrm{R})$$

$$\frac{\Gamma_1 \triangleright c_1 \qquad \Gamma_2\ c_2\ \Gamma_3 \triangleright c_3}{\Gamma_2\ (c_2 \leftarrow c_1)\ \Gamma_1\ \Gamma_3 \triangleright c_3}\ (\leftarrow \mathrm{L}) \qquad \frac{\Gamma\ c_1 \triangleright c_2}{\Gamma \triangleright (c_2 \leftarrow c_1)}\ (\leftarrow \mathrm{R})$$

Fig. 3. Lambek Calculus – Sequent calculus style

Definition 10. *A* Lambek categorial grammar *is a tuple* $G = (\Sigma, \mathcal{B}, c_0, \alpha)$ *where:*

- Σ *is a finite set, the* alphabet
- $\mathcal{B}$ *is a set of* basic categories *with an associated category system* $\mathcal{C}$ *(the reflexive-transitive closure of* $\mathcal{B}$ *under* $\rightarrow$ *and* $\leftarrow$*)*
- c_0 *is the* target category
- $\alpha : \Sigma \longrightarrow P_f(\mathcal{C})$ *is the* lexicon*, a mapping assigning each terminal symbol in* Σ *a finite non-empty subset of categories from* $\mathcal{C}$

The language generated by G *is defined to be:*

$$L(G) = \{w = \sigma_1 \dots \sigma_n \in \Sigma^+ \mid \exists c_1 \dots c_n : c_i \in \alpha[\sigma_i], i = 1 \dots n, \vdash_L c_1 \dots c_n \triangleright c_0\}$$

Lambek categorial grammars have been proven to derive exactly the class of context free languages, see [15,16,17]. However, as context-free languages do not cover all natural language structures, several extensions have been explored. One

$$\frac{\Gamma_1\ c_1\ \Gamma_2 \triangleright c_3}{\Gamma_1\ c_1\ \cap\ c_2\ \Gamma_2 \triangleright c_3}\ (\cap\ \mathrm{L}_1) \qquad \frac{\Gamma_1\ c_1\ \Gamma_2 \triangleright c_3}{\Gamma_1\ c_2\ \cap\ c_1\ \Gamma_2 \triangleright c_3}\ (\cap\ \mathrm{L}_2)$$

$$\frac{\Gamma \triangleright c_1 \qquad \Gamma \triangleright c_2}{\Gamma \triangleright c_1\ \cap\ c_2}\ (\cap\ \mathrm{R})$$

Fig. 4. Kanazawa's intersective conjunction rules

such extension is Kanazawa's work [18] in which he suggests an enrichment of the Lambek calculus with intersective conjunction (see Fig. 4).

Similarly to CG, Kanazawa's grammars accept all finite intersections of context-free languages, e.g., the multiple-agreement and cross-agreement languages etc. Kanazawa also shows that his extended grammars accept languages not obtained as a finite intersection of context-free languages by proving that they can accept the langauge $L = \{a^{2^{n^2}} \mid n \in \mathbb{N}\}$.[7] This is a super-linear language, similarly to the CG generated language from Example 5.

It would be interesting to compare Kanazawa's model to CG and SAPDA as there seem to be many similarities between them. Categorial grammars are used mainly in the field of computational linguistics, so such a comparison could have a bearing on natural language processing as well as formal language theory.

7 Concluding Remarks

We have introduced a synchronized model of Alternating Pushdown Automata, SAPDA, which is equivalent to the CG model. As the exact class of languages generated by CG-s is not yet known, the exact class of languages accepted by SAPDA is not known either. Perhaps the formalization as an automaton will help find methods to prove that languages are *not* accepted by the model, thus answering some open questions.

An interesting direction for further research is the exploration of the relation between LCG and SAPDA. It is a well known result, due to Ginsberg and Spanier [19], that linear grammars are equivalent to 1turn-PDA. 1turn-PDA are a subfamily of PDA where in each computation the stack hight switches only once from non-decreasing to decreasing. A similar notion of 1turn-SAPDA can be defined, where each stack branch can make only one turn in the course of a computation. Our initial results point towards an equivalence between 1turn-SAPDA and LCG. If this equivalence holds, it will deepen the correlation between SAPDA and CG, strengthening the claim that SAPDA are a natural model for CG.

In [20], Kutrib and Malcher explore a wide range of finite-turn automata with and without turn conditions, and their relationships with closures of linear context-free languages under regular operations. It would also prove interesting to explore the general case of finite-turn SAPDA, perhaps finding models for closures of linear conjunctive languages under regular operations.

[7] L is not a finite intersection of context-free languages, as all unary context-free languages are regular, and regular languages are closed under intersection [18].

Acknowledgments

The authors are grateful to Nissim Francez for his remarks on the first draft of this paper. The comments of the anonymous referee which considerably improved the presentation are greatly appreciated.

The work of Michael Kaminski was supported by the fund for promotion of research at the Technion.

References

1. Okhotin, A.: Conjunctive grammars. Journal of Automata, Languages and Combinatorics 6(4), 519–535 (2001)
2. Okhotin, A.: A recognition and parsing algorithm for arbitrary conjunctive grammars. Theoretical Computer Science 302, 81–124 (2003)
3. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. Journal of the ACM 28(1), 114–133 (1981)
4. Ladner, R.E., Lipton, R.J., Stockmeyer, L.J.: Alternating pushdown and stack automata. SIAM Journal on Computing 13(1), 135–155 (1984)
5. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (1979)
6. Okhotin, A.: Efficient automaton-based recognition for linear conjunctive languages. International Journal of Foundations of Computer Science 14(6), 1103–1116 (2003)
7. Okhotin, A.: On the equivalence of linear conjunctive grammars and trellis automata. RAIRO Theoretical Informatics and Applications 38(1), 69–88 (2004)
8. Culik II, K., Gruska, J., Salomaa, A.: Systolic trellis automata, i and ii. International Journal of Computer Mathematics 15 & 16(1 & 3–4), 3–22, 195–212 (1984)
9. Vijay-Shanker, K., Weir, D.J.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory 27(6), 511–546 (1994)
10. Okhotin, A.: On the closure properties of linear conjunctive languages. Theor. Comput. Sci. 299(1-3), 663–685 (2003)
11. Moriya, E.: A grammatical characterization of alternating pushdown automata. Theoretical Computer Science 67(1), 75–85 (1989)
12. Ibarra, O.H., Jiang, T., Wang, H.: A characterization of exponential-time languages by alternating context-free grammars. Theoretical Computer Science 99(2), 301–313 (1992)
13. Moortgat, M.: Categorial type logics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language, Elsevier, Amsterdam (1997)
14. Lambek, J.: The mathematics of sentance structure. American Mathematical Monthly 65(3), 154–170 (1958)
15. Bar-Hillel, Y., Gaifman, C., Shamir, E.: On categorial and phrase structure grammars. Bulletin of the Research Council of Israel 9(F), 1–16 (1960)
16. Cohen, J.M.: The equivalence of two concepts of categorial grammar. Information and Control 10, 475–484 (1967)
17. Pentus, M.: Lambek grammars are context free. In: Proc. of 8th Ann. IEEE Symp. on Logic in Computer Science, pp. 429–433 (1993)

18. Kanazawa, M.: The lambek calculus enriched with additional connectives. Journal of Logic, Language and Information 1(2), 141–171 (1992)
19. Ginsgurg, S., Spanier, E.H.: Finite-turn pushdown automata. SIAM Journal on Control 4(3), 429–453 (1966)
20. Kutrib, M., Malcher, A.: Finite-turn pushdown automata. Discreet Applied Mathematics 155, 2152–2164 (2007)

Expressive Power and Decidability for Memory Logics

Carlos Areces[1], Diego Figueira[2], Santiago Figueira[3,4], and Sergio Mera[3,⋆]

[1] INRIA Nancy Grand Est, France
[2] LSV, ENS Cachan, CNRS, INRIA, France
[3] Departamento de Computación, FCEyN, UBA, Argentina
[4] CONICET, Argentina

Abstract. Taking as inspiration the hybrid logic $\mathcal{HL}(\downarrow)$, we introduce a new family of logics that we call *memory logics*. In this article we present in detail two interesting members of this family defining their formal syntax and semantics. We then introduce a proper notion of bisimulation and investigate their expressive power (in comparison with modal and hybrid logics). We will prove that in terms of expressive power, the memory logics we discuss in this paper are more expressive than orthodox modal logic, but less expressive than $\mathcal{HL}(\downarrow)$. We also establish the undecidability of their satisfiability problems.

1 Memory Logics: Hybrid Logics with a Twist

Hybrid languages have been extensively investigated in the past years. $\mathcal{HL}$, the simplest hybrid language, is usually presented as the basic modal language $\mathcal{K}$ extended with special symbols (called *nominals*) to name individual states in a model. These new symbols are simply a new sort of atomic symbols $\{i, j, k, \ldots\}$ disjoint from the set of standard propositional variables. While they behave syntactically exactly as propositional variables do, their semantic interpretation differ: nominals denote elements in the model, instead of sets of elements. This simple addition already results in increased expressive power. For example the formula $i \wedge \langle r \rangle i$ is true in a state w, only if w is a reflexive point named by the nominal i. As the basic modal language is invariant under unraveling, there is no equivalent modal formula [1].

But as we said above, $\mathcal{HL}$ is just the simplest hybrid language. Once nominals have been added to the language, other natural extensions arise. Having names for states at our disposal we can introduce, for each nominal i, an operator $@_i$ that allows us to jump to the point named by i obtaining the language $\mathcal{HL}(@)$. The formula $@_i\varphi$ (read 'at i, φ') moves the point of evaluation to the state named by i and evaluates φ there. Intuitively, the $@_i$ operators internalize the satisfaction relation '$\models$' into the logical language: $\mathcal{M}, w \models \varphi$ iff $\mathcal{M} \models @_i\varphi$, where i is a nominal naming w. For this reason, these operators are usually called *satisfaction operators*.

⋆ S. Mera is partially supported by a grant of Fundación YPF.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 56–68, 2008.

If nominals are names for individual states, why not introduce also binders. We would then be able to write formulas like $\forall i.\langle r\rangle i$, which will be true at a state w if it is related to all states in the domain. The $\forall$ quantifier is very expressive: the satisfiability problem of $\mathcal{HL}(\forall)$ ($\mathcal{HL}$ extended with the universal binder $\forall$) is undecidable [2]. Moreover, $\mathcal{HL}(@, \forall)$ is expressively equivalent to full first-order logic (over the appropriate signature).

From a modal perspective, other binders besides $\forall$ are possible. The $\downarrow$ binder binds nominals to the *current* point of evaluation. In essence, it enables us to create a name for the here-and-now, and refer to it later in the formula. For example, the formula $\downarrow i.\langle r\rangle i$ is true at a state w if and only if it is related to itself. The intuitive reading is quite straightforward: the formula says "call the current state i and check that i is reachable". The logic $\mathcal{HL}(\downarrow)$ is also very expressive but weaker than $\mathcal{HL}(\forall)$. Sadly, its satisfiability problem is also undecidable.

Different binders for hybrid logics have been investigated in detail (see [2]), but in this article we want to take a look at $\downarrow$ from a slightly different perspective: we will consider nominals and $\downarrow$ as ways for storing and retrieving information in the model.

Models as Information Storage. We should note that nominals and $\downarrow$ work nicely together. Whereas $\downarrow i$ stores the current point of evaluation in the nominal i, nominals act as checkpoints enabling us to retrieve stored information by verifying if the current point is named by a given nominal i. To make this point clear, let's define formally the semantics of $\mathcal{HL}(\downarrow)$.

Definition 1. *A hybrid signature $\mathcal{S}$ is a tuple $\langle \text{PROP}, \text{REL}, \text{NOM}\rangle$ where* PROP, REL, NOM *are mutually disjoint infinite enumerable sets (the sets of propositional symbols, relational symbols and nominals, respectively).*

Formulas of $\mathcal{HL}(\downarrow)$ are defined over a given $\mathcal{S}$ by the following rules

$$\text{FORMS} ::= p \mid i \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle r\rangle\varphi \mid \downarrow i.\varphi,$$

where $p \in$ PROP, $i \in$ NOM, $r \in$ REL and $\varphi, \varphi_1, \varphi_2 \in$ FORMS. Formulas in which any nominal i appears in the scope of a binder $\downarrow i$ are called sentences.

A model for $\mathcal{HL}(\downarrow)$ over a signature $\mathcal{S}$ is a tuple $\langle W, (R_r)_{r\in\text{REL}}, V, g\rangle$ where $\langle W, (R_r)_{r\in\text{REL}}, V\rangle$ is a standard Kripke model (i.e., W is a non empty set, each R_r is a binary relation over W, and V is a valuation), and g is an assignment function from NOM *to W.*

Given a model $\mathcal{M} = \langle W, (R_r)_{r\in\text{REL}}, V, g\rangle$ the semantic conditions for the propositional and modal operators are defined as usual (see [1]), and in addition:

$$\begin{aligned} \langle W, (R_r)_{r\in\text{REL}}, V, g\rangle, w &\models i \text{ iff } g(i) = w \\ \langle W, (R_r)_{r\in\text{REL}}, V, g\rangle, w &\models \downarrow i.\varphi \text{ iff } \langle W, (R_r)_{r\in\text{REL}}, V, g^i_w\rangle, w \models \varphi \\ &\qquad \text{where } g^i_w \text{ is the assignment identical to } g \\ &\qquad \text{except perhaps in that } g^i_w(i) = w. \end{aligned}$$

We can think that $\downarrow i$ is *modifying* the model (by storing the current point of evaluation into i), and that i is being evaluated in the modified model. We can

see the assignment g as a particular type of 'information storage' in our model, and consider $\downarrow$ and i as our way to access this information storage for reading and writing.

But let us take a step back and consider the new picture. When we introduced the $\downarrow$ binder, our main aim was to define a binder which was weaker than the first-order quantifier. We thought of the semantics of $\downarrow$ first, and we suitably adjusted the way we updated the assignment later. But why do we need to restrict ourselves to binders and assignments?

Let us start with a standard Kripke models $\langle W, (R_r)_{r\in\mathrm{REL}}, V\rangle$, and let us consider a very simple addition: just a set $S \subseteq W$. We can, for example, think of S as a set of states that are, for some reason, 'known' to us. Already in this very simple set up we can define the following operators

$$\begin{array}{l}\langle W, (R_r)_{r\in\mathrm{REL}}, V, S\rangle, w \models \textcircled{r}\varphi \text{ iff } \langle W, (R_r)_{r\in\mathrm{REL}}, V, S\cup\{w\}\rangle, w \models \varphi\\ \langle W, (R_r)_{r\in\mathrm{REL}}, V, S\rangle, w \models \textcircled{k} \text{ iff } w \in S.\end{array}$$

As it is clear from the semantic definition, the 'remember' operator $\textcircled{r}$ (a unary modality) just marks the current state as being 'already visited', by storing it in our 'memory' S. On the other hand, the zero-ary operator $\textcircled{k}$ (for 'known') queries S to check if the current state has already been visited.

In this simple language we would have that $\langle W, (R_r)_{r\in\mathrm{REL}}, V, \emptyset\rangle, w \models \textcircled{r}\langle r\rangle\textcircled{k}$ will be true only if w is reflexive. Is this new logic equivalent to $\mathcal{HL}(\downarrow)$? As we will prove in this article, the answer is negative: the new language is less expressive than $\mathcal{HL}(\downarrow)$ but more expressive than $\mathcal{K}$. Intuitively, in the new language we cannot discern between states stored in S, while an assignment g keeps a complete mapping between states and nominals.

Naturally, we can include structures which are richer than a simple set, in our models. Let us consider one example. Let S be now a stack of elements that we will represent as a list that 'grows to the right' (we will denote the act of pushing w in S as $S \cdot w$). Let us define the operators:

$$\begin{array}{rcl}\langle W, (R_r)_{r\in\mathrm{REL}}, V, S\rangle, w \models \texttt{(push)}\varphi & \text{iff} & \langle W, (R_r)_{r\in\mathrm{REL}}, V, S\cdot w\rangle, w \models \varphi\\ \langle W, (R_r)_{r\in\mathrm{REL}}, V, S\cdot w'\rangle, w \models \texttt{(pop)}\varphi & \text{iff} & \langle W, (R_r)_{r\in\mathrm{REL}}, V, S\rangle, w \models \varphi\\ \langle W, (R_r)_{r\in\mathrm{REL}}, V, []\rangle, w \models \texttt{(pop)}\varphi & \text{never} & \\ \langle W, (R_r)_{r\in\mathrm{REL}}, V, S\cdot w'\rangle, w \models \texttt{top} & \text{iff} & w = w'.\end{array}$$

We will call this new family of logics *memory logics* ($\mathcal{M}$) and in this article we will focus on $\mathcal{M}(\textcircled{r}, \textcircled{k})$, i.e., the logic $\mathcal{K}$ extended with the operators $\textcircled{r}$ and $\textcircled{k}$ introduced above, and investigate two possible variations.

More generally, our proposal is to take seriously the usual saying that 'modal languages are languages to talk about labeled graphs' but give us the freedom to choose what we want to 'remember' about a given graph and how we are going to store it.

To close this section, we formally define the syntax and semantics of the logics we will investigate in the rest of the article.

Syntax and semantics for $\mathcal{M}(\textcircled{r}, \textcircled{k})$. Syntactically, we obtain $\mathcal{M}(\textcircled{r}, \textcircled{k})$ by extending the basic modal language $\mathcal{K}$ with the $\textcircled{r}$ and $\textcircled{k}$ modalities.

Definition 2 (Syntax). *Let* PROP $= \{p_1, p_2, \ldots\}$ *(the propositional symbols) and* REL $= \{r_1, r_2, \ldots\}$ *(the relational symbols) be pairwise disjoint, countable infinite sets of symbols. The set* FORMS *of formulas of* $\mathcal{M}(ⓡ, ⓚ)$ *in the signature* $\langle \text{PROP}, \text{REL} \rangle$ *is defined as:*

$$\text{FORMS} ::= p \mid ⓚ \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle r \rangle \varphi \mid ⓡ\varphi,$$

where $p \in$ PROP, $r \in$ REL *and* $\varphi, \varphi_1, \varphi_2 \in$ FORMS.

While the syntax of the logics that we will discuss in this article is the same, they differ subtly in their semantics.

Definition 3 (Semantics). *Given a signature* $\mathcal{S} = \langle \text{PROP}, \text{REL} \rangle$*, a model for* $\mathcal{M}(ⓡ, ⓚ)$ *is a tuple* $\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle$*, where* $\langle W, (R_r)_{r \in \text{REL}}, V \rangle$ *is a standard Kripke model and* $S \subseteq W$*. The semantics is defined as:*

$$\begin{aligned}
\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w &\models p \text{ iff } w \in V(p) \\
\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w &\models \neg\varphi \text{ iff } \langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w \not\models \varphi \\
\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w &\models \varphi \wedge \psi \text{ iff } \langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w \models \varphi \\
&\qquad \text{and } \langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w \models \psi \\
\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w &\models \langle r \rangle \varphi \text{ iff there is } w' \text{ such that } R_r(w, w') \\
&\qquad \text{and } \langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w' \models \varphi \\
\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w &\models ⓡ\varphi \text{ iff } \langle W, (R_r)_{r \in \text{REL}}, V, S \cup \{w\} \rangle, w \models \varphi \\
\langle W, (R_r)_{r \in \text{REL}}, V, S \rangle, w &\models ⓚ \text{ iff } w \in S
\end{aligned}$$

In this paper, we will be especially interested in the case where formulas are evaluated in models with no previously 'remembered' states, that is, the case where $S = \emptyset$. We will call $\mathcal{M}_\emptyset(ⓡ, ⓚ)$ the logic that results from restricting the class of models to those with $S = \emptyset$.

2 Bisimulation

Here we will define a proper notion of bisimulation for $\mathcal{M}(ⓡ, ⓚ)$ and $\mathcal{M}_\emptyset(ⓡ, ⓚ)$, and use it to investigate their expressive power. We will use a presentation in terms of Ehrenfeucht games [3], but a relational presentation is also possible.

We start with some notation. Given $\mathcal{M} = \langle W, (R_r)_{r \in \text{REL}}, V, S \rangle$ and states $w_1, \ldots, w_n$, we define $\mathcal{M}[w_1, \ldots, w_n] = \langle W, (R_r)_{r \in \text{REL}}, V, S \cup \{w_1, \ldots, w_n\} \rangle$. The set of propositions that are true at a given state w is defined as $\mathsf{props}(w) = \{p \in \text{PROP} \mid w \in V(p)\}$. Given two models $\mathcal{M} = \langle W, (R_r)_{r \in \text{REL}}, V, S \rangle$ and $\mathcal{M}' = \langle W', (R'_r)_{r \in \text{REL}}, V', S' \rangle$, and states $w \in W$ and $w' \in W'$, we say that they *agree* if $\mathsf{props}(w) = \mathsf{props}(w')$ and $w \in S$ iff $w' \in S'$.

Bisimulation Games for $\mathcal{M}(ⓡ, ⓚ)$. Let $S = \langle \text{PROP}, \text{REL} \rangle$ be a standard modal signature. Let $\mathcal{M}_1 = \langle W_1, (R^1_r)_{r \in \text{REL}}, V_1, S_1 \rangle$ and $\mathcal{M}_2 = \langle W_2, (R^2_r)_{r \in \text{REL}}, V_2, S_2 \rangle$ be models and let $w_1 \in W_1$ and $w_2 \in W_2$ be agreeing states. We define the *Ehrenfeucht game* $E(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ as follows. There are two players called *Spoiler* and *Duplicator*. In a play of the game, the players move alternatively.

Spoiler always makes the first move. At every move, Spoiler starts by choosing in which model he will make a move. Let us set $s = 1$ and $d = 2$ in case he chooses $\mathcal{M}_1$; otherwise, let $s = 2$ and $d = 1$. He can then either:

1. Make a *memorizing step*. I.e., he extends S_s to $S_s \cup \{w_s\}$. The game then continues with $E(\mathcal{M}_1[w_1], \mathcal{M}_2[w_2], w_1, w_2)$.
2. Make a *move step*. I.e., he chooses $r \in \text{REL}$, and v_s, an R^s_r-successor of w_s. If w_s has no R^s_r-successors, then Duplicator wins. Duplicator has to chose v_d, an R^d_r-successor of w_d, such that v_s and v_d agree. If there is no such successor, Spoiler wins. Otherwise the game continues with $E(\mathcal{M}_1, \mathcal{M}_2, v_1, v_2)$.

In the case of an infinite game, Duplicator wins. Note that with this definition, exactly one of Spoiler or Duplicator wins each game.

Definition 4 (Bisimulation). *We say that two models $\mathcal{M}_1$ and $\mathcal{M}_2$ are bisimilar (and we write $\mathcal{M}_1 \underline{\leftrightarrow} \mathcal{M}_2$) when there exist $w_1 \in \mathcal{M}_1$ and $w_2 \in \mathcal{M}_2$ such that they agree and Duplicator has a winning strategy on $E(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$. In this case we also say that w_1 and w_2 are bisimilar ($\mathcal{M}_1, w_1 \underline{\leftrightarrow} \mathcal{M}_2, w_2$).*

We are now ready to prove that the notion of bisimulation we just introduced is adequate. We will show that formulas of $\mathcal{M}(ⓡ, ⓚ)$ are preserved under bisimulation.

Definition 5 (Logic equivalence). *Given $\mathcal{M}_1, \mathcal{M}_2$ two models, $w_1 \in \mathcal{M}_1$, $w_2 \in \mathcal{M}_2$, we say that w_1 is* equivalent *(for some logic $\mathcal{L}$) to w_2 ($w_1 \leftrightsquigarrow w_2$) if for all φ (in $\mathcal{L}$) we have $\mathcal{M}_1, w_1 \models \varphi$ iff $\mathcal{M}_2, w_2 \models \varphi$.*

Theorem 1. *Let $\mathcal{M}_1, \mathcal{M}_2$ be two models, $w_1 \in \mathcal{M}_1, w_2 \in \mathcal{M}_2$. If $w_1 \underline{\leftrightarrow} w_2$ then $w_1 \leftrightsquigarrow w_2$.*

Proof. We prove that if w_1 and w_2 agree and Duplicator has a winning strategy on $E(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ then $\forall \varphi \in \mathcal{M}(ⓡ, ⓚ)$, $\mathcal{M}_1, w_1 \models \varphi$ iff $\mathcal{M}_2, w_2 \models \varphi$. We proceed by induction on φ.

- The propositional and boolean cases are trivial.
- $\varphi = ⓚ$. This case follows from Definition 3 and because w_1 and w_2 agree.
- $\varphi = \langle r \rangle \psi$. This is the standard modal case. Preservation is ensured thanks to the *move steps* in the definition of the game.
- $\varphi = ⓡ\psi$. We prove that $\mathcal{M}_1, w_1 \models ⓡ\psi$ implies $\mathcal{M}_2, w_2 \models ⓡ\psi$. Suppose $\mathcal{M}_1, w_1 \models ⓡ\psi$ then $\mathcal{M}_1[w_1], w_1 \models \psi$. The following claim is clear.

  Claim. Let $\mathcal{M}_1, \mathcal{M}_2$ be two models, $w_1 \in \mathcal{M}_1, w_2 \in \mathcal{M}_2$. If Duplicator has a winning strategy on $E(\mathcal{M}_1, \mathcal{M}_2, w_1, w_2)$ then he has a winning strategy on $E(\mathcal{M}_1[w_1], \mathcal{M}_2[w_2], w_1, w_2)$.

  By this claim, Duplicator has a winning strategy on $E(\mathcal{M}_1[w_1], \mathcal{M}_2[w_2], w_1, w_2)$. Applying inductive hypothesis and the fact that $\mathcal{M}_1[w_1], w_1 \models \psi$, we conclude $\mathcal{M}_2[w_2], w_2 \models \psi$ and then $\mathcal{M}_2, w_2 \models ⓡ\psi$. The other direction is identical.

This concludes the proof.

The converse of Theorem 1 holds for image-finite models (i.e., models in which the set of successors of any state in the domain is finite). The proof is exactly the same as for $\mathcal{K}$, as ⓡ and ⓚ do not interact with the accessibility relation [1].

Theorem 2 (Hennessy-Milner Theorem). *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two image finite models. Then for every $w_1 \in \mathcal{M}_1$ and $w_2 \in \mathcal{M}_2$, $w_1 \leftrightsquigarrow w_2$ then $w_1 \underline{\leftrightarrow} w_2$.*

Clearly, as Theorems 1 and 2 hold for arbitrary models, the results hold also for $\mathcal{M}_\emptyset(ⓡ, ⓚ)$.

3 Expressivity

In this section we compare the expressive power of memory logics with respect to both the modal and hybrid logics. But comparing the expressive power of these logics poses a complication because, strictly speaking, each of them uses a different class of models. We would like to be able to define a natural mapping between models of each logic, similar to the natural mapping that exists between Kripke models and first-order models [1].

Such a mapping is easy to define in the case of $\mathcal{M}_\emptyset(ⓡ, ⓚ)$: each Kripke model $\langle W, (R_r)_{r\in \mathrm{REL}}, V\rangle$ can be identified with the $\mathcal{M}_\emptyset(ⓡ, ⓚ)$ model $\langle W, (R_r)_{r\in \mathrm{REL}}, V, \emptyset\rangle$. Similarly, for formulas which are sentences, the $\mathcal{M}_\emptyset(ⓡ, ⓚ)$ model $\langle W, (R_r)_{r\in \mathrm{REL}}, V, \emptyset\rangle$ can be identified with the hybrid model $\langle W, (R_r)_{r\in \mathrm{REL}}, V, g\rangle$ (for g arbitrary). As we will discuss below, it is harder to find such a natural way to transform models for the case of $\mathcal{M}(ⓡ, ⓚ)$: the most natural way seems to involve a shift in the signature of the language.

Definition 6 ($\mathcal{L} \leq \mathcal{L}'$). *We say that $\mathcal{L}$ is* not more expressive than *$\mathcal{L}'$ (notation $\mathcal{L} \leq \mathcal{L}'$) if it is possible to define a function* Tr *between formulas of $\mathcal{L}$ and $\mathcal{L}'$ such that for every model $\mathcal{M}$ and every formula φ of $\mathcal{L}$ we have that*

$$\mathcal{M} \models_{\mathcal{L}} \varphi \text{ iff } \mathcal{M} \models_{\mathcal{L}'} \mathsf{Tr}(\varphi).$$

We say that $\mathcal{L}$ is strictly less expressive than $\mathcal{L}'$ (notation $\mathcal{L} < \mathcal{L}'$) if $\mathcal{L} \leq \mathcal{L}'$ but not $\mathcal{L}' \leq \mathcal{L}$.

$\mathcal{K}$ is strictly less expressive than $\mathcal{M}_\emptyset(ⓡ, ⓚ)$. It is easy to see intuitively that ⓡ and ⓚ do bring additional expressive power into the language: with their help we can detect cycles in a given model, while formulas of $\mathcal{K}$ are invariant under unraveling.

Showing that $\mathcal{K} \leq \mathcal{M}_\emptyset(ⓡ, ⓚ)$ is straightforward as $\mathcal{K}$ is a sublanguage of $\mathcal{M}_\emptyset(ⓡ, ⓚ)$. Hence, we can take Tr to be the identity function.

Theorem 3. $\mathcal{K} \leq \mathcal{M}_\emptyset(ⓡ, ⓚ)$.

Proving that $\mathcal{M}_\emptyset(ⓡ, ⓚ)$ is strictly more expressive is only slightly harder.

Theorem 4. $\mathcal{K} \neq \mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$

Proof. Let $\mathcal{M}_1 = \langle\{w\}, \{(w,w)\}, \emptyset\rangle$ and $\mathcal{M}_2 = \langle\{u,v\}, \{(u,v),(v,u)\}, \emptyset\rangle$ be two Kripke models. It is known that they are $\mathcal{K}$ bisimilar (see [1]). On the other hand, the equivalent $\mathcal{M}(\textcircled{r}, \textcircled{k})$ models are distinguishable by $\varphi = \textcircled{r}\langle r\rangle\textcircled{k}$.

$\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ *is strictly less expressive than* $\mathcal{HL}(\downarrow)$. We will define a translation that maps formulas of $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ into sentences of $\mathcal{HL}(\downarrow)$. Intuitively, it is clear that we can use $\downarrow$ to simulate $\textcircled{r}$, but $\textcircled{k}$ does not distinguishes between different memorized states (while nominals binded by $\downarrow$ do distinguish them). We can solve this using disjunction to gather together all previously remembered states.

Theorem 5. $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k}) \leq \mathcal{HL}(\downarrow)$.

Proof. See the technical appendix.

Finally we arrive to the most interesting question in this section: as we already mentioned, $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ seems to be weaker than $\mathcal{HL}(\downarrow)$ because it allows us to remember that we have already visited a given state, but we cannot distinguish among different visited states. Indeed, we can prove that $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ is strictly less expressive than $\mathcal{HL}(\downarrow)$, but the proof is slightly involved.

Theorem 6. $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k}) \neq \mathcal{HL}(\downarrow)$.

Proof. Let $\mathcal{M}_1 = \langle\omega, R_1, \emptyset, \emptyset\rangle$ and $\mathcal{M}_2 = \langle\omega, R_2, \emptyset, \emptyset\rangle$, where $R_1 = \{(n,m) \mid n \neq m\} \cup \{(0,0)\}$ and $R_2 = \{(n,m) \mid n \neq m\} \cup \{(0,0),(1,1)\}$ (the models are shown in Figure 1, the accessibility relation is the non-reflexive transitive closure of the arrows shown in the picture).

We prove that $\mathcal{M}_1, 0 \underline{\leftrightarrow} \mathcal{M}_2, 0$ showing the winning strategy for duplicator. Intuitively, the strategy for Duplicator consists in the following idea: whenever one player is in $(\mathcal{M}_1, 0)$ the other will be in $(\mathcal{M}_2, 0)$ or $(\mathcal{M}_2, 1)$, and conversely whenever a player is in $(\mathcal{M}_1, n)$, $n > 0$ the other will be in $(\mathcal{M}_2, m)$, $m > 1$. This is maintained until Spoiler (if ever) decides to remember a state. Once this is done, then any strategy will be a winning one for Duplicator.

Being a bit more formal, the winning strategy will have two stages. While Spoiler does not remember any reflexive state, Duplicator plays with the following strategy: if Spoiler chooses 0 in any model, Duplicator chooses 0 in the other

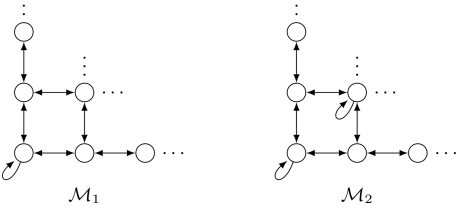


Fig. 1. Two $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$-bisimilar models

one; if Spoiler chooses $n > 0$ in $\mathcal{M}_1$, Duplicator plays $n + 1$ in $\mathcal{M}_2$; if Spoiler chooses $n > 0$ in $\mathcal{M}_2$, Duplicator plays $n - 1$ in $\mathcal{M}_1$.

Notice that with this strategy Spoiler chooses a reflexive state if and only if Duplicator answers with a reflexive one. This is clearly a winning strategy. If ever Spoiler decides to remember a reflexive state, Duplicator starts using the following strategy: if Spoiler selects a state n, Duplicator answers with an agreeing state m of the opposite model. Notice that this is always possible since both n and m see infinitely many non remembered states and at least one remembered state. Therefore $\mathcal{M}_1, w \underline{\leftrightarrow} \mathcal{M}_2, w$.

On the other hand, let φ be the formula $\downarrow i.\langle r\rangle(i \wedge \langle r\rangle(\neg i \wedge \downarrow i.\langle r\rangle i))$. It is easy to see that $\mathcal{M}_1, w \not\models \varphi$ but $\mathcal{M}_2, w \models \varphi$.

The basic idea behind the previous proof is that if the relations R_1 and R_2 extend the set $\{(n, m) \mid n \neq m\}$, then $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ can distinguish between irreflexive and non irreflexive frames, but it cannot distinguish frames with a different number of reflexive nodes.

There is a number of interesting remarks to be made above the previous proof. First, notice that it is essential for the winning strategy of Duplicator that each state in a model is related to infinitely many others. The question of whether $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k}) < \mathcal{HL}(\downarrow)$ on image-finite models is still open. Second, notice that the $\mathcal{HL}(\downarrow)$ sentence that we used in the proof uses only one nominal. Hence, we have actually proved that $\mathcal{HL}_1(\downarrow) \not\leq \mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$, where $\mathcal{HL}_1(\downarrow)$ is $\mathcal{HL}(\downarrow)$ restricted to only one nominal. But actually, it is also the case that $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k}) \not\leq \mathcal{HL}_1(\downarrow)$.

Proposition 1. *The logics $\mathcal{HL}_1(\downarrow)$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ are incomparable in terms of expressive power.*

Proof. See technical appendix.

Actually, this incomparability result can be extended to $\mathcal{HL}(\downarrow)$ restricted to any fixed number of nominals, by taking cliques of the appropriate size.

Theorem 7. *For any fixed k, the logics $\mathcal{HL}_k(\downarrow)$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ are incomparable in terms of expressive power.*

We will now briefly discuss the case of $\mathcal{M}(\textcircled{r}, \textcircled{k})$. As we already mentioned at the beginning of this section, the first required step to compare expressivity is to be able to define a natural mapping between models of the different logics involved. Consider a model $\langle W, (R_r)_{r \in \mathrm{REL}}, V, S\rangle$ for $\mathcal{M}(\textcircled{r}, \textcircled{k})$; if we want to associate a Kripke model we have to decide how to deal with the set S. The only natural choice seems to be to extend the signature with a special propositional variable *known*, and let V' be identical to V excepts that $V'(known) = S$. And the same can be done to obtain a hybrid model from a $\mathcal{M}(\textcircled{r}, \textcircled{k})$ model.

Theorem 8. *The following results concerning expressive power can be established*

1. *$\mathcal{K}$ over the signature $\langle \mathrm{PROP} \cup \{known\}, \mathrm{REL}\rangle$ is strictly less expressive than $\mathcal{M}(\textcircled{r}, \textcircled{k})$ over the signature $\langle \mathrm{PROP}, \mathrm{REL}\rangle$.*

2. $\mathcal{M}(\textcircled{r}, \textcircled{k})$ *over the signature* $\langle \text{PROP}, \text{REL} \rangle$ *is strictly less expressive than* $\mathcal{HL}(\downarrow)$ *over the signature* $\langle \text{PROP} \cup \{known\}, \text{REL}, \text{NOM} \rangle$.
3. $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ *over the signature* $\langle \text{PROP} \cup \{known\}, \text{REL} \rangle$ *is equivalent to* $\mathcal{M}(\textcircled{r}, \textcircled{k})$ *over the signature* $\langle \text{PROP}, \text{REL} \rangle$

Proof. See technical appendix for details.

To close this section, we mention that the satisfaction preserving translations defined in the proof can actually be used to transfer known results, for example, from $\mathcal{HL}(\downarrow)$ to $\mathcal{M}(\textcircled{r}, \textcircled{k})$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$. For instance, both logics are compact and their formulas are preserved by generated submodels (see [4]).

4 Infinite Models and Undecidability

The last issue that we will discuss in this paper is the undecidability of the satisfiability problem for both $\mathcal{M}(\textcircled{r}, \textcircled{k})$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$. The proof is an adaptation of the proof of undecidability of $\mathcal{HL}(\downarrow)$ presented in [2].

We first prove that both languages lack the finite model property [1].

Theorem 9. *There is a formula* $Inf \in \mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ *such that* $\mathcal{M}, w \models Inf$ *implies that the domain of* $\mathcal{M}$ *is an infinite set.*

Proof. The formula *Inf* states that there is a nonempty subset of W that is an unbounded strict partial order. See the technical appendix for details.

To prove failure of the finite model property for the case $\mathcal{M}(\textcircled{r}, \textcircled{k})$ we first notice that the following lemma is easy to establish (we only state it for the monomodal case; a similar result is true in the multimodal case). Failure of the finite model property is then a direct consequence.

Lemma 1. *Let* φ *be a formula of modal depth* d. *If* $\langle W, R_r, V, S \rangle, w \models \left(\bigwedge_{i=0}^{d} [r]^i \neg \textcircled{k}\right) \wedge \varphi$ *then* $\langle W, R_r, V, \emptyset \rangle, w \models \varphi$.

Corollary 1. $\mathcal{M}(\textcircled{r}, \textcircled{k})$ *lacks the finite model property.*

Proof. Using Lemma 1, one can easily see that the formula $Inf \wedge \left(\bigwedge_{i=0}^{4} [r]^i \neg \textcircled{k}\right)$, where *Inf* is the one in the proof of Theorem 9, forces an infinite model.

We now turn to undecidability. We show that $\mathcal{M}(\textcircled{r}, \textcircled{k})$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ are undecidable by encoding the $\omega \times \omega$ tiling problem (see [5]). Following the idea in [2], we construct a spy point over the relation S which has access to every tile. The relations U and R represent moving up and to the right, respectively, from one tile to the other. We code each type of tile with a fixed propositional symbol t_i. With this encoding we define for each tiling problem T, a formula φ^T such that the set of tiles T tiles $\omega \times \omega$ iff φ^T has a model.

Theorem 10. *The satisfiability problem for* $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ *is undecidable.*

Proof. See the technical appendix for details.

Corollary 2. *The satisfiability problem for* $\mathcal{M}(\textcircled{r}, \textcircled{k})$ *is undecidable.*

Proof. Using Lemma 1 and the formula φ^T in Theorem 10, we obtain a formula ψ such that if $\mathcal{M}, w \models \psi$ then $\mathcal{M}$ is a tiling of $\omega \times \omega$. For the converse, we can build exactly the same model as in the above proof.

5 Conclusions and Further Work

In this paper we investigate two members of a family of logics that we called *memory logics*. These logics were inspired by the hybrid logic $\mathcal{HL}(\downarrow)$: the $\downarrow$ operator can be thought of as a storage command, and our aim is to carry this idea further investigating different ways in which information can be stored. We have proved that, in terms of expressive power, the memory logics $\mathcal{M}(\textcircled{r}, \textcircled{k})$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ lay between the basic modal logic $\mathcal{K}$ and the hybrid logic $\mathcal{HL}(\downarrow)$. Unluckily, the reduced expressive power is not sufficient to ensure good computational behavior: both $\mathcal{M}(\textcircled{r}, \textcircled{k})$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ fail to have the finite model property and moreover their satisfiability problems are undecidable.

Despite the negative result concerning decidability, we believe that the new perspective we pursue in this paper is appealing. Clearly, it opens up the way to many new interesting modal languages (we discuss some examples in Sect. 1). As in the case of modal and hybrid languages, all of them seem to share some common behavior, and the challenge is now to discover and understand it.

Much work rest to be done. We are currently working on complete axiomatizations of $\mathcal{M}(\textcircled{r}, \textcircled{k})$ and $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$, and on model theoretic characterizations. Extending the language with nominals is a natural step, and then adapting the internalized hybrid tableau method [6] to the new languages is straightforward. More interesting is to explore new languages of the family (like `(push)`, `(pop)`, or `(forget)`), and interaction between the memory operators and the modalities.

For example, if we restrict the class of models to those in which we are forced to memorize the current state each time we take a step via the accessibility relation, then the logic turns decidable (even though it is still strictly more expressive than $\mathcal{K}$). More precisely, changing the semantic definition of $\langle r \rangle$ to be

$$\langle W, (R_r)_{r \in \mathrm{REL}}, V, S \rangle, w \models \langle r \rangle \varphi \text{ iff } \exists w' \in W, R_r(w, w') \text{ and } \langle W, (R_r)_{r \in \mathrm{REL}}, V, S \cup \{w\} \rangle, w' \models \varphi$$

and calling the resulting logic $\mathcal{M}^-(\textcircled{r}, \textcircled{k})$, then $\mathcal{K} < \mathcal{M}^-(\textcircled{r}, \textcircled{k}) < \mathcal{M}(\textcircled{r}, \textcircled{k})$. Moreover, $\mathcal{M}^-(\textcircled{r}, \textcircled{k})$ has the bounded tree model property: every satisfiable formula φ of $\mathcal{M}^-(\textcircled{r}, \textcircled{k})$ is satisfied in a tree of size bounded by a computable funcion over the size of φ. Hence, the satisfiability problem of $\mathcal{M}^-(\textcircled{r}, \textcircled{k})$ is decidable.

The work presented in this paper is somehow related in spirit with the work on Dynamic Epistemic Logic and other update logics [7,8], but as we discuss in the introduction, our inspiration was rooted in a new interpretation of the $\downarrow$ binder.

References

1. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
2. Blackburn, P., Seligman, J.: Hybrid languages. Journal of Logic, Language and Information 4, 251–272 (1995)
3. Ebbinghaus, H., Flum, J., Thomas, W.: Mathematical Logic. Springer, Heidelberg (1984)
4. Areces, C., Blackburn, P., Marx, M.: Hybrid logics: characterization, interpolation and complexity. The Journal of Symbolic Logic 66(3), 977–1010 (2001)
5. Börger, E., Grädel, E., Gurevich, Y.: The classical decision problem. Springer, Heidelberg (1997)
6. Blackburn, P.: Internalizing labelled deduction. Journal of Logic and Computation 10(1), 137–168 (2000)
7. van Benthem, J.: An essay on sabotage and obstruction. In: Mechanizing Mathematical Reasoning, pp. 268–276 (2005)
8. Gerbrandy, J.: Bisimulations on Planet Kripke. PhD thesis, University of Amsterdam, ILLC Dissertation series DS-1999-01 (1999)

Technical Appendix

Proof (Theorem 5). The translation Tr, taking $\mathcal{M}(\textcircled{r}, \textcircled{k})$ formulas over the signature $\langle \text{PROP}, \text{REL}\rangle$ to $\mathcal{HL}(\downarrow)$ sentences over the signature $\langle \text{PROP}, \text{REL}, \text{NOM}\rangle$ is defined for any finite set $N \subseteq \text{NOM}$ as follows:

$$\begin{aligned}
\mathsf{Tr}_N(p) &= p \quad p \in \text{PROP}\\
\mathsf{Tr}_N(\textcircled{k}) &= \textstyle\bigvee_{i\in N} i\\
\mathsf{Tr}_N(\neg\varphi) &= \neg\mathsf{Tr}_N(\varphi)\\
\mathsf{Tr}_N(\varphi_1 \wedge \varphi_2) &= \mathsf{Tr}_N(\varphi_1) \wedge \mathsf{Tr}_N(\varphi_2)\\
\mathsf{Tr}_N(\langle r\rangle\varphi) &= \langle r\rangle\mathsf{Tr}_N(\varphi)\\
\mathsf{Tr}_N(\textcircled{r}\varphi) &= \downarrow i.\mathsf{Tr}_{N\cup\{i\}}(\varphi) \quad \text{where } i \notin N.
\end{aligned}$$

A simple induction shows that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, g, w \models \mathsf{Tr}_\emptyset(\varphi)$, for any g.

Proof (Proposition 1). As we said, $\mathcal{HL}_1(\downarrow) \not\leq \mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k})$ is a direct consequence of the proof of Theorem 6. To prove $\mathcal{M}_\emptyset(\textcircled{r}, \textcircled{k}) \not\leq \mathcal{HL}_1(\downarrow)$, let $\mathcal{M}_1 = \langle\{1,2,3\}, \{(i,j) \mid 1 \leq i,j \leq 3\}, \emptyset, \emptyset\rangle$ (a clique of size 3) and $\mathcal{M}_2 = \langle\{1,2\}, \{(i,j) \mid 1 \leq i,j \leq 2\}, \emptyset, \emptyset\rangle$ (a clique of size 2). It is easy to check that $\mathcal{M}_1, 1 \underline{\leftrightarrow}_{\mathcal{HL}_1(\downarrow)} \mathcal{M}_2, 1$. However, the formula $\varphi = \textcircled{r}\langle r\rangle(\neg\textcircled{k} \wedge \textcircled{r}\langle r\rangle(\neg\textcircled{k} \wedge \textcircled{r}\langle r\rangle\neg\textcircled{k}))$ distinguishes the models: $\mathcal{M}_1, 1 \models \varphi$ but $\mathcal{M}_2, 1 \not\models \varphi$.

Proof (Theorem 8). All proofs are similar to (and sometimes easier than) the ones presented above. We only discuss 2. To prove $\mathcal{M}(\textcircled{r}, \textcircled{k}) \leq \mathcal{HL}(\downarrow)$ (over the appropriate signatures) we adapt the translation Tr with the following clause for $\textcircled{k}$

$$\mathsf{Tr}_N(\textcircled{k}) = \big(\textstyle\bigvee_{i\in N} i\big) \vee \mathit{known}.$$

$\mathcal{HL}(\downarrow) \not\leq \mathcal{M}(\textcircled{r}, \textcircled{k})$ can be shown using the following models. Let $\mathcal{M}_1 = \langle\{w\}, \{(w,w)\}, \emptyset, \{w\}\rangle$ and $\mathcal{M}_2 = \langle\{u,v\}, \{(u,v),(v,u)\}, \emptyset, \{u,v\}\rangle$. Duplicator always

wins on $E(\mathcal{M}_1, \mathcal{M}_2, w, u)$ and thus $\mathcal{M}_1, w \underline{\leftrightarrows}_{\mathcal{M}(\textcircled{r},\textcircled{k})} \mathcal{M}_2, u$. On the other hand, $\mathcal{M}'_1, w \models_{\mathcal{HL}(\downarrow)} \downarrow i.\langle r\rangle i$ but $\mathcal{M}'_2, u \not\models_{\mathcal{HL}(\downarrow)} \downarrow i.\langle r\rangle i$, for $\mathcal{M}'_1, \mathcal{M}'_2$ the models corresponding to $\mathcal{M}_1$ and $\mathcal{M}_2$.

Proof (Theorem 9). Consider the following formulas:

$$
\begin{aligned}
&(\mathit{Back})\ p \wedge [r]\neg p \wedge \langle r\rangle \top \wedge \textcircled{r}([r]\langle r\rangle \textcircled{k})\\
&(\mathit{Spy})\ \textcircled{r}([r][r](\neg p \rightarrow \textcircled{r}(\langle r\rangle(p \wedge \textcircled{k} \wedge \langle r\rangle(\neg p \wedge \textcircled{k})))))\\
&(\mathit{Irr})\ [r]\textcircled{r}\neg\langle r\rangle\textcircled{k}\\
&(\mathit{Succ})\ [r]\langle r\rangle\neg p\\
&(\mathit{3cyc})\ \neg(\langle r\rangle\textcircled{r}\langle r\rangle(\neg p \wedge \langle r\rangle(\neg p \wedge \neg\textcircled{k} \wedge \langle r\rangle\textcircled{k})))\\
&(\mathit{Tran})\ [r]\textcircled{r}[r](\neg p \rightarrow ([r](\neg p \rightarrow (\textcircled{r}\langle r\rangle(p \wedge \langle r\rangle(\textcircled{k} \wedge \langle r\rangle\textcircled{k}))))))
\end{aligned}
$$

Let *Inf* be $\mathit{Back} \wedge \mathit{Spy} \wedge \mathit{Irr} \wedge \mathit{Succ} \wedge \mathit{3cyc} \wedge \mathit{Tran}$. Let $\mathcal{M} = \langle W, R, V, \emptyset\rangle$. We show that if $\mathcal{M}, w \models \mathit{Inf}$, then W is infinite.

Suppose $\mathcal{M}, w \models \mathit{Inf}$. Notice that if $\textcircled{k}$ holds in a state, is because it was previously remembered by the evaluating formula. Let $B = \{b \in W \mid wRb\}$. Because *Back* is satisfied, $w \notin B$, $B \neq \emptyset$ and for all $b \in B$, bRw. Because *Spy* is satisfied, if $a \neq w$ and a is a successor of an element of B then a is also an element of B. As *Irr* is satisfied at w, every state in B is irreflexive. As *Succ* is satisfied at w, every point in B has a successor distinct from w. As *3cyc* is satisfied, there cannot be 3 different elements in B forming a cycle, and this sentence together with *Tran* force R to transitively order B.

It follows that B is an unbounded strict partial order, hence infinite, and so is W.

Proof (Theorem 10). Let $T = \{T_1, \ldots, T_n\}$ be a set of tile types. Given a tile type T_i, $u(T_i)$, $r(T_i)$, $d(T_i)$, $l(T_i)$ will represent the colors of the up, right, down and left edges of T_i r espectively. Define

$$
\begin{aligned}
&(\mathit{Back})\ p \wedge [S]\neg p \wedge \langle S\rangle \top \wedge \textcircled{r}([S]\langle S\rangle\textcircled{k}) \wedge \textcircled{r}([S][S]\textcircled{k})\\
&(\mathit{Spy})\ \textcircled{r}[S][\dagger]\textcircled{r}\langle S\rangle(\textcircled{k} \wedge p \wedge \langle S\rangle(\textcircled{k} \wedge \neg p)), \qquad \text{where } \dagger \in \{U, R\}\\
&(\mathit{Grid})\ [S][U]\neg p \wedge [S][R]\neg p \wedge [S]\langle U\rangle \top \wedge [S]\langle r\rangle \top\\
&(\mathit{Func})\ \textcircled{r}[S]\textcircled{r}\langle\dagger\rangle\textcircled{r}\langle S\rangle\langle S\rangle(\textcircled{k} \wedge \langle\dagger\rangle\textcircled{k} \wedge [\dagger]\textcircled{k}), \qquad \text{where } \dagger \in \{U, R\}\\
&(\mathit{Irr})\ [S]\textcircled{r}[\dagger]\neg\textcircled{k}, \qquad \text{where } \dagger \in \{U, R\}\\
&(\mathit{2cyc})\ [S]\textcircled{r}[\dagger][\dagger]\neg\textcircled{k}, \qquad \text{where } \dagger \in \{U, R\}\\
&(\mathit{Confluent})\ [S]\textcircled{r}\langle U\rangle\langle r\rangle\textcircled{r}\langle S\rangle\langle S\rangle(\textcircled{k} \wedge \langle U\rangle\langle r\rangle\textcircled{k} \wedge \langle r\rangle\langle U\rangle\textcircled{k})\\
&(\mathit{UR\text{-}Irr})\ [S]\textcircled{r}[U][R]\neg\textcircled{k}\\
&(\mathit{UR\text{-}2cyc})\ [S]\textcircled{r}[U][R][U][R]\neg\textcircled{k}\\
&(\mathit{Unique})\ [S]\left(\bigvee_{1\leq i\leq n} t_i \wedge \bigwedge_{1\leq i<j\leq n}(t_i \rightarrow \neg t_j)\right)\\
&(\mathit{Vert})\ [S]\bigwedge_{1\leq i\leq n}\left(t_i \rightarrow \langle U\rangle \bigvee_{1\leq j\leq n, u(T_i)=d(T_j)} t_j\right)\\
&(\mathit{Horiz})\ [S]\bigwedge_{1\leq i\leq n}\left(t_i \rightarrow \langle r\rangle \bigvee_{1\leq j\leq n, r(T_i)=l(T_j)} t_j\right)
\end{aligned}
$$

Let the formula φ^T be the conjunction of all the above formulas. We show that T tiles $\omega \times \omega$ iff φ^T is satisfiable.

Suppose $\mathcal{M}, w \models \varphi^T$. Observe that (*Back*) and (*Spy*) impose w to be a spy point over all its S-accessible states of $\mathcal{M}$. These S-accessible states will be the tiles. From this it follows that $[S]\psi$ holds at w iff ψ is true at every tile. Additionally, $\langle S\rangle\langle S\rangle\psi$ holds at tile v iff ψ is true at some tile (maybe the same one).

Taking the above points into account, one can establish the following. (*Grid*) states that from every tile there is another tile moving up (that is, following the U-relation). The same holds for the right direction (following the R-relation). (*Func*) forces that U and R are both functionals, given that (*Irr*) and (*2cyc*) guarantee irreflexivity and asymmetry of U and R respectively. (*Confluent*) imposes that the tiles are arranged in a grid pattern. To make its job, (*Confluent*) needs the composed relation $U \circ R$ to be irreflexive and asymmetric, and this is done by (*UR-Irr*) and (*UR-2cyc*) respectively.

All the formulas we discuss up to now configure the grid. The last three ensure that every tile has a unique type t_i, and that the colors of the tiles match properly. From this, it easily follows that $\mathcal{M}$ is a tiling of $\omega \times \omega$.

For the converse, suppose $f : \omega \times \omega \to T$ is a tiling of $\omega \times \omega$. We define the model $\mathcal{M} = \langle W, \{S, U, R\}, V, \emptyset\rangle$ as follows:

- $W = \omega \times \omega \cup \{w\}$
- $S = \{(w, v), (v, w) \mid v \in \omega \times \omega\}$ (hence w will act as the spy point)
- $U = \{((x, y), (x, y + 1)) \mid x, y \in \omega\}$
- $R = \{((x, y), (x + 1, y)) \mid x, y \in \omega\}$
- $V(p) = \{w\}$; $V(t_i) = \{x \mid x \in \omega \times \omega, f(x) = T_i\}$

The reader may verify that, by construction, $\mathcal{M}, w \models \varphi^T$.

Reasoning with Uncertainty by Nmatrix–Metric Semantics

Ofer Arieli[1] and Anna Zamansky[2,⋆]

[1] Department of Computer Science, The Academic College of Tel-Aviv, Israel
oarieli@mta.ac.il
[2] Department of Computer Science, Tel-Aviv University, Israel
annaz@post.tau.ac.il

Abstract. Non-deterministic matrices, a natural generalization of many-valued matrices, are semantic structures in which the value assigned to a complex formula may be chosen non-deterministically from a given set of options. We show that by combining Nmatrices and preferential metric-based considerations, one obtains a family of logics that are useful for reasoning with uncertainty. We investigate the basic properties of these logics and demonstrate their usefulness in handling incomplete and inconsistent information.

1 Introduction

One of the main challenges of commonsense reasoning is dealing with phenomena that are inherently non-deterministic. The causes of non-determinism may vary: partially unknown information, faulty behavior of devices and ambiguity of natural languages are just a few cases in point. It is clear that truth-functional semantics, in which the truth-value of a complex formula is completely determined by the truth-values of its subformulas, cannot capture non-deterministic behaviour, the very essence of which is, in some sense, contradictory to the principle of truth-functionality. One possible solution is to borrow the idea of non-deterministic computations from automata and computability theory and apply it to evaluations of formulas. This idea led to introducing *non-deterministic matrices* (Nmatrices) in [8]. These structures are a natural generalization of standard multi-valued matrices [13,25], in which the truth-value of a complex formula can be chosen *non-deterministically* out of some non-empty set of options. The use of Nmatrices preserves many attractive properties of logics with ordinary finite-valued logics, such as decidability and compactness. Moreover, as in many-valued logics, the consequence relations induced by Nmatrices are monotonic (i.e., the set of conclusions monotonically grow in the size of the premises), and are trivialized in the presence of inconsistency (i.e., any inconsistent set of premises entails every formula). In real life, however, both of these properties are not always desirable as, e.g., it is often the case that information systems are exposed to contradictory evidence and that new information

⋆ Supported by the Israel Science Foundation, grant No. 809–06.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 69–82, 2008.

requires a retraction of old assertions. To cope with this, Shoham [22] introduced the notion of *preferential semantics* (see also [20]), according to which an order relation, reflecting some condition or preference criteria, is defined on a set of valuations, and only the valuations that are minimal with respect to this order are relevant for making inferences from a given theory. Following this idea, we use metric-like considerations as our primary preference criteria. Such distance minimization considerations are a cornerstone behind many paradigms of handling incomplete or inconsistent information, such as belief revision [9,14,18,23] database integration systems [1,5,10,19], and formalisms for commonsense reasoning in the context of social choice theory [16,21]. In [2,3,7] this approach is described in terms of entailment relations, based on a standard truth-functional semantics. As argued above, this cannot capture non-deterministic behavior, so instead, in this paper, we use logics based on Nmatrices as the underlying formalism for a preferential metric-based approach. We also consider some of the properties of the entailment relations that are obtained, demonstrate their applicability for reasoning under uncertainty by some case studies, and show the relation between reasoning in these cases and some well-known SAT problems.

2 Distance-Based Non-deterministic Semantics

2.1 Non-deterministic Matrices

In what follows, $\mathcal{L}$ denotes a propositional language with a set Atoms of atomic formulas. A theory Γ is a *finite multiset* of $\mathcal{L}$-formulas, for which $\mathsf{Atoms}(\Gamma)$ and $\mathsf{SF}(\Gamma)$ denote, respectively, the atomic formulas of Γ and the subformulas of Γ. Below, we shortly reproduce the main definitions from [8].

Definition 1. A *non-deterministic matrix* (henceforth, *Nmatrix*) for $\mathcal{L}$ is a tuple $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, where $\mathcal{V}$ is a non-empty set of truth values, $\mathcal{D}$ is a non-empty proper subset of $\mathcal{V}$, and for every n-ary connective $\diamond$ of $\mathcal{L}$, $\mathcal{O}$ includes an n-ary function $\widetilde{\diamond}$ from $\mathcal{V}^n$ to $2^{\mathcal{V}} - \{\emptyset\}$.

Definition 2. An $\mathcal{M}$-*valuation* is a function $\nu : \mathcal{L} \to \mathcal{V}$ that satisfies the following condition for every n-ary connective $\diamond$ of $\mathcal{L}$ and every $\psi_1, \ldots, \psi_n \in \mathcal{L}$,

$$\nu(\diamond(\psi_1, \ldots, \psi_n)) \in \widetilde{\diamond}(\nu(\psi_1), \ldots, \nu(\psi_n)).$$

We denote by $\Lambda_{\mathcal{M}}$ the space of all the $\mathcal{M}$-valuations.

Note that in Nmatrices the truth-values assigned to $\psi_1, \ldots, \psi_n$ do not uniquely determine the truth-value assigned to $\diamond(\psi_1, \ldots, \psi_n)$, as ν makes a non-deterministic choice out of the set of options $\tilde{\diamond}(\nu(\psi_1), \ldots, \nu(\psi_n))$. Thus, the non-deterministic semantics is non-truth-functional, as opposed to standard many-valued logics.

Example 1. Let $\mathcal{M} = \langle \{t, f\}, \{t\}, \mathcal{O} \rangle$, where $\mathcal{O}$ contains the following operators:

	$\neg$
t	$\{f\}$
f	$\{t\}$

$\rightarrow$	t	f
t	$\{t\}$	$\{f\}$
f	$\{t\}$	$\{t\}$

$\leftrightarrow$	t	f
t	$\{t\}$	$\{f\}$
f	$\{f\}$	$\{t\}$

$\vee$	t	f
t	$\{t\}$	$\{t\}$
f	$\{t\}$	$\{f\}$

$\curlywedge$	t	f
t	$\{t, f\}$	$\{f\}$
f	$\{f\}$	$\{f\}$

Let $p, q \in \mathsf{Atoms}$ and $\nu_1, \nu_2 \in \Lambda_{\mathcal{M}}$, such that $\nu_1(p) = \nu_2(p) = \nu_1(q) = \nu_2(q) = t$, $\nu_1(p \curlywedge q) = t$ and $\nu_2(p \curlywedge q) = f$. While ν_1 and ν_2 coincide on, e.g., $p \vee q$, and on the proper subformulas of $p \curlywedge q$, they make different non-deterministic choices for $p \curlywedge q$.

Definition 3. A valuation $\nu \in \Lambda_{\mathcal{M}}$ is a *model* of (or *satisfies*) a formula ψ in $\mathcal{M}$ if $\nu(\psi) \in \mathcal{D}$. ν is a *model* in $\mathcal{M}$ of a set Γ of formulas if it satisfies every formula in Γ. A formula ψ is *$\mathcal{M}$-satisfiable* if it is satisfied by a valuation in $\Lambda_{\mathcal{M}}$. ψ is an *$\mathcal{M}$-tautology* if it is satisfied by every valuation in $\Lambda_{\mathcal{M}}$.

Definition 4. For an Nmatrix $\mathcal{M}$, a formula ψ, and a theory Γ in $\mathcal{L}$, denote: $mod_{\mathcal{M}}(\psi) = \{\nu \in \Lambda_{\mathcal{M}} \mid \nu(\psi) \in \mathcal{D}\}$ and $mod_{\mathcal{M}}(\Gamma) = \cap_{\psi \in \Gamma} mod_{\mathcal{M}}(\psi)$.

Definition 5. The consequence relation that is induced by an Nmatrix $\mathcal{M}$ is defined by: $\Gamma \models_{\mathcal{M}} \psi$ if $mod_{\mathcal{M}}(\Gamma) \subseteq mod_{\mathcal{M}}(\psi)$.

In this paper we concentrate on two-valued Nmatrices with $\mathcal{V} = \{t, f\}$ and $\mathcal{D} = \{t\}$, and denote by $\mathcal{M}$ such an Nmatrix.

2.2 Preferential Distance-Based Entailments

Next, we augment non-deterministic semantics with preferential considerations. The idea is simple: given a distance function d on a space of valuations, reasoning with a set of premises Γ is based on those valuations that are 'd-closest' to Γ (called the *most plausible valuations* of Γ). For instance, under the standard interpretation of negation, it is intuitively clear that valuations in which q is true should be closer to $\Gamma = \{p, \neg p, q\}$ than valuations in which q is false, and so q should follow from Γ while $\neg q$ should *not* follow from Γ, although Γ is not consistent. The formal details are given in [2,3] and are adapted to the non-deterministic case in what follows.

Definition 6. A *pseudo-distance* on a set U is a total function $d : U \times U \to \mathbb{R}^+$, satisfying the following conditions:

- *symmetry:* for all $\nu, \mu \in U$ $d(\nu, \mu) = d(\mu, \nu)$,
- *identity preservation:* for all $\nu, \mu \in U$ $d(\nu, \mu) = 0$ iff $\nu = \mu$.

A pseudo-distance d is a *distance* (*metric*) on U if it has the following property:

- *triangular inequality:* for all $\nu, \mu, \sigma \in U$ $d(\nu, \sigma) \leq d(\nu, \mu) + d(\mu, \sigma)$.

Example 2. For every $\mathcal{M}$, the following two functions are distances on $\Lambda_{\mathcal{M}}$.

- *The drastic distance:* $d_U(\nu, \mu) = 0$ if $\nu = \mu$ and $d_U(\nu, \mu) = 1$ otherwise.
- *The Hamming distance:* $d_H(\nu, \mu) = |\{p \in \mathsf{Atoms} \mid \nu(p) \neq \mu(p)\}|$. [1,2]

[1] Here, the set Atoms of the atomic formulas in the language is assumed to be *finite*.

[2] The drastic distance is also known as the discrete metric, and Hamming distance is sometimes called Dalal distance [11], or the symmetric difference. For other representations of distances between propositional valuations see, e.g., [16].

The non-deterministic character of our framework induces some further restrictions on the distances that we shall use. This is so, since two valuations for an Nmatrix can agree on all the atoms of a formula, but still assign two different values to that formula, thus for computing distances between valuations it is not enough to consider only atomic formulas.[3] It follows that even under the assumption that the set of atoms is finite, there are infinitely many complex formulas to consider. To handle this, the distance computations in the sequel are *context dependent*, that is: restricted to a certain set of relevant formulas.

Definition 7. A *context* C is a finite set of $\mathcal{L}$-formulas closed under subformulas. The *restriction to* C of a valuation $\nu \in \Lambda_{\mathcal{M}}$ is a valuation $\nu^{\downarrow\mathsf{C}}$ on C, such that $\nu^{\downarrow\mathsf{C}}(\psi) = \nu(\psi)$ for every ψ in C. The restriction to C of $\Lambda_{\mathcal{M}}$ is the set $\Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}} = \{\nu^{\downarrow\mathsf{C}} \mid \nu \in \Lambda_{\mathcal{M}}\}$, that is, $\Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}}$ consists of all the $\mathcal{M}$-valuations on C.

Example 3. Consider the following functions on $\Lambda_{\mathcal{M}}^{\downarrow\mathsf{SF}(\Gamma)} \times \Lambda_{\mathcal{M}}^{\downarrow\mathsf{SF}(\Gamma)}$:

- $d_U^{\downarrow\mathsf{SF}(\Gamma)}(\nu,\mu) = \begin{cases} 0 & \text{if } \nu(\psi) = \mu(\psi) \text{ for every } \psi \in \mathsf{SF}(\Gamma), \\ 1 & \text{otherwise.} \end{cases}$
- $d_H^{\downarrow\mathsf{SF}(\Gamma)}(\nu,\mu) = |\{\psi \in \mathsf{SF}(\Gamma) \mid \nu(\psi) \neq \mu(\psi)\}|$.

Proposition 1. *$d_U^{\downarrow\mathsf{SF}(\Gamma)}$ and $d_H^{\downarrow\mathsf{SF}(\Gamma)}$ are distance functions on $\Lambda_{\mathcal{M}}^{\downarrow\mathsf{SF}(\Gamma)}$.* [4]

Definition 8. Let d be a function on $\cup_{\mathcal{M}} = \bigcup_{\{\mathsf{C}=\mathsf{SF}(\Gamma) \mid \Gamma \in 2^{\mathcal{L}}\}} \Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}} \times \Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}}$

- The *restriction* of d to a context C is a function $d^{\downarrow\mathsf{C}}$ on $\Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}} \times \Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}}$, defined for every $\nu, \mu \in \Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}}$ by $d^{\downarrow\mathsf{C}}(\nu,\mu) = d(\nu,\mu)$.
- d is a *generic (pseudo) distance on* $\Lambda_{\mathcal{M}}$, if for every context C, $d^{\downarrow\mathsf{C}}$ is a (pseudo) distance on $\Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}}$.

Example 4. Given an Nmatrix $\mathcal{M}$ for $\mathcal{L}$, define the functions d_U and d_H on $\cup_{\mathcal{M}}$ as follows: for every context C and every $\nu, \nu \in \Lambda_{\mathcal{M}}^{\downarrow\mathsf{C}}$,

- $d_U(\nu,\mu) = \begin{cases} 0 & \text{if } \nu = \mu, \\ 1 & \text{otherwise.} \end{cases}$
- $d_H(\nu,\mu) = |\{\psi \in \mathsf{C} \mid \nu(\psi) \neq \mu(\psi)\}|$.

The restrictions of the two functions to a context $\mathsf{C} = \mathsf{SF}(\Gamma)$ are given in Example 3. By Proposition 1, then, both of these functions are generic distances on $\Lambda_{\mathcal{M}}$ for every Nmatrix $\mathcal{M}$.

[3] Thus, e.g., the Hamming distance defined in the last example should be adjusted to the non-deterministic case, so that differences in the truth assignment of complex formulas will be taken into consideration as well.

[4] This proposition is easily verifiable. Proofs of some other propositions in this paper appear in the appendix.

Note 1. Denote by $\mathcal{M}_c$ the Nmatrix for the language $\{\neg, \wedge, \vee, \rightarrow\}$ with the classical interpretations of the connectives (i.e., $\mathcal{M}_c$ is similar to the classical deterministic matrix, except that its valuation functions return singletons of truth-values instead of truth-values). Under the assumption that the set of atoms is finite, the distance functions in Example 2 can be represented in the non-deterministic case as metrics on $\Lambda_{\mathcal{M}_c}^{\downarrow \mathsf{Atoms}}$; In the notations of Example 4, they are generic distances on $\Lambda_{\mathcal{M}_c}$, denoted by $d_U^{\downarrow \mathsf{Atoms}}$ and $d_H^{\downarrow \mathsf{Atoms}}$.

Definition 9. A *numeric aggregation function* is total function f whose argument is a multiset of real numbers and whose values are real numbers, such that: (i) f is non-decreasing in the value of its argument,[5] (ii) $f(\{x_1, \ldots, x_n\}) = 0$ iff $x_1 = x_2 = \ldots x_n = 0$, and (iii) $f(\{x\}) = x$ for every $x \in \mathbb{R}$.

Definition 10. A (distance-based, non-deterministic) *setting* for a language $\mathcal{L}$, is a triple $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$, where $\mathcal{M}$ is a non-deterministic matrix for $\mathcal{L}$, d is a generic distance on $\Lambda_{\mathcal{M}}$, and f is an aggregation function.

Definition 11. Given a setting $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$ for a language $\mathcal{L}$, a valuation $\nu \in \Lambda_{\mathcal{M}}$, and a set $\Gamma = \{\psi_1, \ldots, \psi_n\}$ of formulas in $\mathcal{L}$, define:

- $d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_i) =$
$$\begin{cases} \min\{d^{\downarrow \mathsf{SF}(\Gamma)}(\nu^{\downarrow \mathsf{SF}(\Gamma)}, \mu^{\downarrow \mathsf{SF}(\Gamma)}) \mid \mu \in mod_{\mathcal{M}}(\psi_i)\} & \text{if } mod_{\mathcal{M}}(\psi_i) \neq \emptyset, \\ 1 + \max\{d^{\downarrow \mathsf{SF}(\Gamma)}(\mu_1^{\downarrow \mathsf{SF}(\Gamma)}, \mu_2^{\downarrow \mathsf{SF}(\Gamma)}) \mid \mu_1, \mu_2 \in \Lambda_{\mathcal{M}}\} & \text{otherwise.} \end{cases}$$
- $\delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \Gamma) = f(\{d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_1), \ldots, d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_n)\})$.

Note 2. In every setting $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$, the following properties hold:

1. In the two extreme degenerate cases, when ψ is either a tautology or a contradiction w.r.t. $\mathcal{M}$, all the valuations are equally distant from ψ. Otherwise, the valuations that are closest to ψ are its models and their distance to ψ is zero. This also implies that $\delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \Gamma) = 0$ iff $\nu \in mod_{\mathcal{M}}(\Gamma)$ (see also [3]).
2. A natural property of distances between valuations and formulas is that they are not affected (biased) by irrelevant formulas (those that are not part of the relevant context):

Proposition 2 (unbiasedness). *For every* $\nu_1, \nu_2 \in \Lambda_{\mathcal{M}}$, $\mathsf{C} = \mathsf{SF}(\Gamma)$, *and* $\psi \in \Gamma$, *if* $\nu_1^{\downarrow \mathsf{C}} = \nu_2^{\downarrow \mathsf{C}}$ *then* $d^{\downarrow \mathsf{C}}(\nu_1, \psi) = d^{\downarrow \mathsf{C}}(\nu_2, \psi)$ *and* $\delta_{d,f}^{\downarrow \mathsf{C}}(\nu_1, \Gamma) = \delta_{d,f}^{\downarrow \mathsf{C}}(\nu_2, \Gamma)$.

Now we define entailment relations based on distance minimization.

Definition 12. Given a setting $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$, the *most plausible valuations* of a theory Γ are defined as follows:

$$\Delta_{\mathcal{S}}(\Gamma) = \begin{cases} \{\nu \in \Lambda_{\mathcal{M}} \mid \forall \mu \in \Lambda_{\mathcal{M}} \ \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \Gamma) \leq \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \Gamma)\} & \text{if } \Gamma \neq \emptyset, \\ \Lambda_{\mathcal{M}} & \text{otherwise.} \end{cases}$$

Definition 13. Let $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$. Define: $\Gamma \models_{\mathcal{S}} \psi$ if $\Delta_{\mathcal{S}}(\Gamma) \subseteq mod_{\mathcal{M}}(\psi)$.

[5] That is, the function value is non-decreasing when an element in the multiset is replaced by a larger element.

2.3 Examples of Reasoning with $\models_{\mathcal{S}}$

Notation. Given a theory Γ with $\mathsf{SF}(\Gamma) = \{\psi_1, \psi_2, \ldots, \psi_n\}$, a valuation $\nu \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{SF}(\Gamma)}$ is represented by $\{\psi_1 : \nu(\psi_1), \psi_2 : \nu(\psi_2), \ldots, \psi_n : \nu(\psi_n)\}$.

Example 5. Let $\mathcal{S} = \langle \mathcal{M}, d_U, \Sigma \rangle$, where $\mathcal{M}$ is the Nmatrix considered in Example 1. Let $\Gamma = \{p, \neg p, q, \neg(p \barwedge q)\}$. Then:

$$\Delta_{\mathcal{S}}(\Gamma) = \left\{ \begin{array}{l} \{p{:}t,\ \neg p{:}f,\ q{:}t,\ p \barwedge q{:}f,\ \neg(p \barwedge q){:}t\}, \\ \{p{:}f,\ \neg p{:}t,\ q{:}t,\ p \barwedge q{:}f,\ \neg(p \barwedge q){:}t\} \end{array} \right\}.$$

Thus, $\Gamma \models_{\mathcal{S}} q$ and $\Gamma \models_{\mathcal{S}} \neg(p \barwedge q)$, while $\Gamma \not\models_{\mathcal{S}} p$ and $\Gamma \not\models_{\mathcal{S}} \neg p$.

Example 6. A reasoner wants to learn as much as possible about a (black-box) circuit, the structure of which is assumed to be the following:

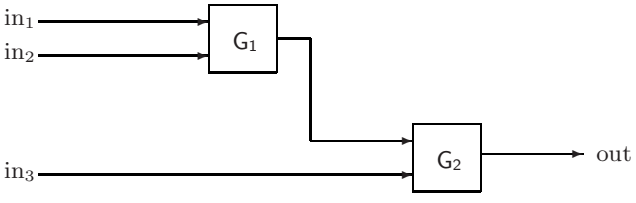


Fig. 1.

Here, $\mathsf{G_1}$ and $\mathsf{G_2}$ are two AND gates that are faulty or behave unpredictably when both of their input lines are 'on'.[6] After experimenting with the circuit, the reasoner concludes that if one of the input lines is 'on' then so is the output line. This situation may be represented by the Nmatrix $\mathcal{M}$ of Example 1 as follows:

$$\Gamma = \left\{ (in_1 \vee in_2 \vee in_3) \rightarrow out \right\},$$

where *out* denotes the formula $((in_1 \barwedge in_2) \barwedge in_3)$. Here, $\Lambda_{\mathcal{M}}^{\downarrow \mathsf{SF}(\Gamma)}$ has 11 elements (see the appendix), two of them are models of Γ. Thus, by Lemma 1 below, for *every* setting $\mathcal{S}$,

$$\Delta_{\mathcal{S}}(\Gamma) = mod_{\mathcal{M}}(\Gamma) = \left\{ \begin{array}{l} \{in_1{:}t,\ in_2{:}t,\ in_3{:}t,\ in_1 \barwedge in_2{:}t,\ out{:}t\}, \\ \{in_1{:}f,\ in_2{:}f,\ in_3{:}f,\ in_1 \barwedge in_2{:}f,\ out{:}f\} \end{array} \right\},$$

so the reasoner may conclude that when all the input lines have the same value, the output line of the circuit preserves this value.

Suppose now that the reasoner learns that the value of the output line is always different than the value of $\mathsf{G_1}$. The new situation can be represented by

$$\Gamma' = \Gamma \cup \left\{ (in_1 \barwedge in_2) \leftrightarrow \neg out \right\}.$$

[6] This may happen due to noises on or off chip, variations in the manufacturing process, adversary operations, etc.

It is easy to verify that Γ' is not $\mathcal{M}$-satisfiable anymore, i.e. the new information is inconsistent with the reasoner's previous knowledge. In such cases the usual $\models_{\mathcal{M}}$ entailment is trivialized: everything can be inferred from Γ'. This, however, is not the case for $\models_{\mathcal{S}}$. For instance, when $\mathcal{S} = \langle \mathcal{M}, d_U, \Sigma \rangle$, we have that

$$\Delta_{\mathcal{S}}(\Gamma') = \left\{ \begin{array}{l} \{in_1{:}t,\ in_2{:}t,\ in_3{:}t,\ in_1 \curlywedge in_2{:}t,\ out{:}t\}, \\ \{in_1{:}t,\ in_2{:}t,\ in_3{:}t,\ in_1 \curlywedge in_2{:}t,\ out{:}f\}, \\ \{in_1{:}t,\ in_2{:}t,\ in_3{:}f,\ in_1 \curlywedge in_2{:}t,\ out{:}f\}, \\ \{in_1{:}f,\ in_2{:}f,\ in_3{:}f,\ in_1 \curlywedge in_2{:}f,\ out{:}f\} \end{array} \right\}.$$

Using $\models_{\mathcal{S}}$, the reasoner may still conclude from Γ' that if the value of all the input lines is 'off', this is also the value of the output line. This shows that $\models_{\mathcal{S}}$ is inconsistency-tolerant (see Proposition 4 below). On the other hand, a stronger assertion, that when the values of all input lines coincide the value of the output line is the same, is no longer a valid consequence of Γ'. This shows that $\models_{\mathcal{S}}$ is non-monotonic (see Proposition 6 below).

3 General Properties of $\models_{\mathcal{S}}$

In this section, we consider some basic properties of the entailments that are induced by a setting $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$. First, we consider the relation between basic and distance-based entailments.

Proposition 3. [6] *For every setting $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$, if $\Gamma \models_{\mathcal{S}} \psi$ then $\Gamma \models_{\mathcal{M}} \psi$. Moreover, if Γ is $\mathcal{M}$-satisfiable, then $\Gamma \models_{\mathcal{S}} \psi$ iff $\Gamma \models_{\mathcal{M}} \psi$.*

Proposition 3 follows from the fact that if Γ is not $\mathcal{M}$-satisfiable then $\Gamma \models_{\mathcal{M}} \psi$ for *every* ψ, and from the following lemma:

Lemma 1. [6] *Γ is $\mathcal{M}$-satisfiable iff $\Delta_{\mathcal{S}}(\Gamma) = mod_{\mathcal{M}}(\Gamma)$.*

Thus, $\models_{\mathcal{S}}$ coincides with $\models_{\mathcal{M}}$ w.r.t. $\mathcal{M}$-consistent premises. In contrast to $\models_{\mathcal{M}}$, however, $\models_{\mathcal{S}}$ tolerates inconsistent information in a non-trivial way, thus, as Proposition 4 shows, $\models_{\mathcal{S}}$ is paraconsistent.

Definition 14. Γ_1 and Γ_2 are called *independent* if $\mathsf{Atoms}(\Gamma_1) \cap \mathsf{Atoms}(\Gamma_2) = \emptyset$.

The next proposition is an improvement of a similar proposition in [6].

Proposition 4 (paraconsistency). *For every Γ and every ψ such that Γ and $\{\psi\}$ are independent, $\Gamma \models_{\mathcal{S}} \psi$ iff ψ is an $\mathcal{M}$-tautology.*

Corollary 1 (weak paraconsistency). *For every Γ there is a ψ s.t. $\Gamma \not\models_{\mathcal{S}} \psi$.*

A related property is that $\models_{\mathcal{S}}$ preserves the consistency of its conclusions:

Definition 15. An Nmartix $\mathcal{M} = \langle \{t, f\}, \{t\}, \mathcal{O} \rangle$ is *with negation*, if there is a unary function $\widetilde{\neg}$ in $\mathcal{O}$ such that $\widetilde{\neg}(t) = \{f\}$ and $\widetilde{\neg}(f) = \{t\}$. A setting $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$ is with negation if its Nmatrix $\mathcal{M}$ is with negation.

Proposition 5. [6] *Let $\mathcal{S}$ be a setting with negation. Then for every Γ and every ψ, if $\Gamma \models_{\mathcal{S}} \psi$ then $\Gamma \not\models_{\mathcal{S}} \neg\psi$.*

We now consider to what extent the entailment relations of our framework are non-monotonic (i.e., whether conclusions may be revised in light of new information).

Proposition 6 (non-monotonicity). *Let $\mathcal{S} = \langle \mathcal{M}, d, f\rangle$ be a setting with negation. Then $\models_{\mathcal{S}}$ is non-monotonic.*

In spite of Proposition 6, even for settings with negation, one may specify conditions under which the entailment relations have some monotonic characteristics.

Definition 16. An aggregation function f is *hereditary*, if $f(\{x_1,\ldots,x_n\}) < f(\{y_1,\ldots,y_n\})$ entails $f(\{x_1,\ldots,x_n,z_1,\ldots,z_m\}) < f(\{y_1,\ldots,y_n,z_1,\ldots,z_m\})$.

Example 7. The aggregation function Σ is hereditary, while max is not.

The following proposition shows that in light of new information that is unrelated to the premises, previously drawn conclusions should not be retracted.[7]

Proposition 7 (rational monotonicity). *Let $\mathcal{S} = \langle \mathcal{M}, d, f\rangle$ be a setting in which f is hereditary. If $\Gamma \models_{\mathcal{S}} \psi$, then $\Gamma, \phi \models_{\mathcal{S}} \psi$ for every formula ϕ such that $\Gamma \cup \{\psi\}$ and $\{\phi\}$ are independent.*

The discussion above, on the non-monotonicity of $\models_{\mathcal{S}}$, brings us to the question to what extent these entailments can be considered as consequence relations.

Definition 17. A Tarskian *consequence relation* [24] for a language $\mathcal{L}$ is a binary relation $\vdash$ between sets of formulas of $\mathcal{L}$ and formulas of $\mathcal{L}$ that satisfies the following conditions:

Reflexivity:	if $\psi \in \Gamma$, then $\Gamma \vdash \psi$.
Monotonicity:	if $\Gamma \vdash \psi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \vdash \psi$.
Transitivity:	if $\Gamma \vdash \psi$ and $\Gamma', \psi \vdash \phi$, then $\Gamma, \Gamma' \vdash \phi$.

As follows from Example 5 and Proposition 6, entailments of the form $\models_{\mathcal{S}}$ are, in general, neither reflexive nor monotonic. It is also not difficult to verify that in general $\models_{\mathcal{S}}$ is not transitive either. In the context of non-monotonic reasoning, however, it is usual to consider the following weaker conditions that guarantee a 'proper behaviour' of nonmonotonic entailments in the presence of inconsistency (see, e.g., [4,15,17,20]):

Definition 18. A *cautious* consequence relation for $\mathcal{L}$ is a relation $\mid\!\sim$ between sets of $\mathcal{L}$-formulas and $\mathcal{L}$-formulas, that satisfies the following conditions:

Cautious Reflexivity:	if Γ is $\mathcal{M}$-satisfiable and $\psi \in \Gamma$, then $\Gamma \mid\!\sim \psi$.
Cautious Monotonicity [12]:	if $\Gamma \mid\!\sim \psi$ and $\Gamma \mid\!\sim \phi$, then $\Gamma, \psi \mid\!\sim \phi$.
Cautious Transitivity [15]:	if $\Gamma \mid\!\sim \psi$ and $\Gamma, \psi \mid\!\sim \phi$, then $\Gamma \mid\!\sim \phi$.

[7] This type of monotonicity is a kind of *rational monotonicity*, considered in [17].

The next result is another improvement of a similar proposition in [6].

Proposition 8. *Let $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$ be a setting where f is hereditary. Then $\models_{\mathcal{S}}$ is a cautious consequence relation.*

Regarding the computability of our entailments, we show that in most of the practical cases entailment checking is decidable.

Definition 19. A setting $\mathcal{S} = \langle \mathcal{M}, d, f \rangle$ is *computable*, if f is computable, and there is an algorithm that computes $d(\mu, \nu)$ for every context C and $\mu, \nu \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{C}}$.

Note 3. Clearly, all the distance and aggregation functions considered in this paper are computable. Yet, as the following example shows, this is not always the case. Let $\mathcal{L} = \{\wedge\}$ be a propositional language and L a first-order language with a constant c, a unary function g and a binary relation R. Consider the following one-to-one mapping Θ from L-formulas to $\mathcal{L}$-formulas: every symbol s in L is associated with an atomic formula p_s in $\mathcal{L}$, and every L-formula ψ is mapped to the $\mathcal{L}$-formula $\Theta(\psi)$, obtained by taking the conjunction of all the atomic formulas to which the symbols of ψ are mapped. For instance, the formula $\forall x_1 \forall x_2 R(x_1, x_2)$ is mapped to $p_\forall \wedge p_{x_1} \wedge p_\forall \wedge p_{x_2} \wedge p_R \wedge p_(\wedge p_{x_1} \wedge p_, \wedge p_{x_2} \wedge p_)$. A formula ψ in $\mathcal{L}$ is called *proper* if there is an L-formula ψ' s.t. $\psi = \Theta(\psi')$. Now, consider the following pseudo distance:

$$d(\nu, \mu) = \begin{cases} 0 & \text{if } \nu = \mu, \\ 1 & \text{if } \nu \neq \mu \text{ and there is a proper } \psi \text{ s.t. } \nu, \mu \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{SF}(\psi)}, \\ & \text{and } \Theta^{-1}(\psi) \text{ is satisfiable,} \\ 2 & \text{otherwise.} \end{cases}$$

Since $\mathsf{SF}(\psi) \neq \mathsf{SF}(\phi)$ whenever $\psi \neq \phi$, the pseudo distance above is well defined. Now, as the satisfiability problem for L-formulas is undecidable, d is not computable.

Proposition 9. *For every computable setting $\mathcal{S}$, the question whether $\Gamma \models_{\mathcal{S}} \psi$ is decidable.*

4 Some Particular Cases of Reasoning with $\models_{\mathcal{S}}$

In this section we focus on drastic settings, i.e., settings with a drastic distance (see Examples 2 and 4). In this context we investigate the following aggregation functions:

Definition 20. An aggregation function f is *range restricted* if $f(\{x_1, \ldots, x_n\}) \in \{x_1, \ldots, x_n\}$; f is called *additive* if for any non-empty set S it can be represented as $f(S) = g(|S|) \cdot \Sigma_{x \in S}\, x$, for some function $g : \mathbb{N}^+ \to \mathbb{R}^+$.

Example 8. The maximum function is a range-restricted but not additive, while the summation (respectively, the average) is additive where g is uniformly 1 (respectively, $g(n) = \frac{1}{n}$), but it is not range-restricted.

The next proposition should be compared with Proposition 4.

Proposition 10. *Let $\mathcal{S} = \langle \mathcal{M}, d_U, f\rangle$ be a drastic setting in which f is range restricted. Let Γ be a set of formulas that is not $\mathcal{M}$-satisfiable. Then $\Gamma \models_{\mathcal{S}} \psi$ iff ψ is an $\mathcal{M}$-tautology.*

Corollary 2. *Let $\mathcal{S}$ be a drastic setting with a range-restricted aggregation function. If $\Gamma \models_{\mathcal{S}} \psi$ then either $\Gamma \models_{\mathcal{M}} \psi$ or ψ is an $\mathcal{M}$-tautology.*

The last corollary shows that reasoning with drastic distances and range-restricted functions has a somewhat 'crude nature': either the set of premises is $\mathcal{M}$-consistent, in which case the set of conclusions coincides with that of the basic entailment, or, in case of contradictory premises, only tautologies are entailed.

The behavior of drastic settings with additive functions is completely different: entailments in this case are closely related to the maximum satisfiability problem:

Definition 21. Let $\mathsf{SAT}_{\mathcal{M}}(\Gamma)$ be the set of all the $\mathcal{M}$-satisfiable subsets of Γ. The set $\mathsf{mSAT}_{\mathcal{M}}(\Gamma)$ of the *maximally $\mathcal{M}$-satisfiable* subsets of Γ consistent of all the elements $\Upsilon \in \mathsf{SAT}_{\mathcal{M}}(\Gamma)$ such that $|\Upsilon'| \leq |\Upsilon|$ for every $\Upsilon' \in \mathsf{SAT}_{\mathcal{M}}(\Gamma)$.

Note 4. Clearly, $\mathsf{mSAT}_{\mathcal{M}}(\Gamma)$ is nonempty whenever Γ contains an $\mathcal{M}$-satisfiable element.

Proposition 11. *Let $\mathcal{S} = \langle \mathcal{M}, d_U, f\rangle$ be a drastic setting with additive f and let Γ be a finite set of formulas. Then:*

$$\Delta_{\mathcal{S}}(\Gamma) = \begin{cases} \{\nu \in mod_{\mathcal{M}}(\Upsilon) \mid \Upsilon \in \mathsf{mSAT}_{\mathcal{M}}(\Gamma)\} & \text{if } \mathsf{mSAT}_{\mathcal{M}}(\Gamma) \neq \emptyset, \\ \Lambda_{\mathcal{M}} & \text{otherwise.} \end{cases}$$

Corollary 3. *Let $\mathcal{S}$ be a drastic setting with additive f. If $\mathsf{mSAT}_{\mathcal{M}}(\Gamma) \neq \emptyset$ and $\Gamma' \models_{\mathcal{M}} \psi$ for every $\Gamma' \in \mathsf{mSAT}_{\mathcal{M}}(\Gamma)$, then $\Gamma \models_{\mathcal{S}} \psi$.*

Example 9. By taking $\mathcal{S} = \langle \mathcal{M}_c, d_U, \Sigma\rangle$ in the last corollary, we get that reasoning with summation of drastic distances is equivalent to checking classical entailments from the maximally consistent subsets of the premises.

Note 5. It is easy to verify that all the results in this section still hold for settings $\mathcal{S} = \langle \mathcal{M}, d, f\rangle$, where for every context $\mathsf{C} = \mathsf{SF}(\Gamma)$ there is some constant $k_{\mathsf{C}} > 0$, such that for all $\psi \in \Gamma$ and $\nu \in \Lambda_{\mathcal{M}}$,

$$d^{\downarrow \mathsf{C}}(\nu, \psi) = \begin{cases} 0 & \text{if } \nu \in mod_{\mathcal{M}}(\psi), \\ k_{\mathsf{C}} & \text{otherwise.} \end{cases}$$

Note that drastic settings $\mathcal{S} = \langle \mathcal{M}, d_U, f\rangle$ are a particular instance of this definition, in which $k_{\mathsf{C}} = 1$ for every context C.

References

1. Arenas, M., Bertossi, L., Chomicki, J.: Answer sets for consistent query answering in inconsistent databases. Theory and Practice of Logic Programming 3(4–5), 393–424 (2003)
2. Arieli, O.: Commonsense reasoning by distance semantics. In: Proc. TARK 2007, pp. 33–41 (2007)
3. Arieli, O.: Distance-based paraconsistent logics. International Journal of Approximate Reasoning (in press, 2008), doi:10.1016/j.ijar.2007.07.002
4. Arieli, O., Avron, A.: General patterns for nonmonotonic reasoning: from basic entailments to plausible relations. Logic Journal of the IGPL 8(2), 119–148 (2000)
5. Arieli, O., Denecker, M., Bruynooghe, M.: Distance semantics for database repair. Annals of Mathematics and Artificial Intelligence 50(3–4), 389–415 (2007)
6. Arieli, O., Zamansky, A.: Non-deterministic distance-based semantics. In: Proc. AGI 2008. Frontiers in Artificial Intelligence, vol. 171, pp. 39–50. IOS Press, Amsterdam (2008)
7. Arieli, O., Zamansky, A.: Some simplified forms of reasoning with distance-based entailments. In: Bergler, S. (ed.) Canadian AI 2008. LNCS (LNAI), vol. 5032, pp. 36–47. Springer, Heidelberg (2008)
8. Avron, A., Lev, I.: Non-deterministic multi-valued structures. Journal of Logic and Computation 15, 241–261 (2005)
9. Ben Naim, J.: Lack of finite characterizations for the distance-based revision. In: Proc. KR 2006, pp. 239–248 (2006)
10. Chomicki, J., Marchinkowski, J.: Minimal-change integrity maintenance using tuple deletion. Information and Computation 197(1–2), 90–121 (2005)
11. Dalal, M.: Investigations into a theory of knowledge base revision. In: Proc. AAAI 1988, pp. 475–479. AAAI Press, Menlo Park (1988)
12. Gabbay, D.: Theoretical foundation for non-monotonic reasoning, Part II: Structured non-monotonic theories. In: Proc. SCAI 1991. IOS Press, Amsterdam (1991)
13. Gottwald, S.: A Treatise on Many-Valued Logics. Studies in Logic and Computation, vol. 9. Research Studies Press (2001)
14. Grove, A.: Two modellings for theory change. J. Phil. Logic 17, 157–180 (1988)
15. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1–2), 167–207 (1990)
16. Lafage, C., Lang, J.: Propositional distances and preference representation. In: Benferhat, S., Besnard, P. (eds.) ECSQARU 2001. LNCS (LNAI), vol. 2143, pp. 48–59. Springer, Heidelberg (2001)
17. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55, 1–60 (1992)
18. Lehmann, D., Magidor, M., Schlechta, K.: Distance semantics for belief revision. Journal of Symbolic Logic 66(1), 295–317 (2001)
19. Lopatenko, A., Bertossi, L.: Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 179–193. Springer, Heidelberg (2006)
20. Makinson, D.: General patterns in nonmonotonic reasoning. In: Gabbay, D., Hogger, C., Robinson, J. (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3, pp. 35–110 (1994)
21. Pigozzi, G.: Two aggregation paradoxes in social decision making: the ostrogorski paradox and the discursive dilemma. Episteme 2(2), 33–42 (2005)
22. Shoham, Y.: Reasoning about change. MIT Press, Cambridge (1988)

23. Spohn, W.: Ordinal conditional functions: a dynamic theory of epistemic states. In: Belief Change and Statistics, vol. II, pp. 105–134. Kluwer, Dordrecht (1988)
24. Tarski, A.: Introduction to Logic. Oxford University Press, Oxford (1941)
25. Urquhart, A.: Many-valued logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. II, pp. 249–295. Kluwer, Dordrecht (2001)

A Supplementary Material

Elaboration on Example 6: Below, we use the following abbreviations:

$$\mathsf{G}_1 = (in_1 \nrightarrow in_2), \qquad \psi_1 = (in_1 \vee in_2 \vee in_3) \rightarrow out,$$
$$out = ((in_1 \nrightarrow in_2) \nrightarrow in_3), \qquad \psi_2 = (in_1 \nrightarrow in_2) \leftrightarrow \neg out.$$

In these notations, $\Gamma = \{\psi_1\}$ and $\Gamma' = \{\psi_1, \psi_2\}$. Distances to elements of $\Lambda_{\mathcal{M}}^{\downarrow \mathsf{SF}(\Gamma)}$ are given below, where $\delta(\cdot)$ abbreviates $\delta_{d_U,\Sigma}(\nu, \cdot)$ for the relevant valuation ν.

	in_1	in_2	in_3	G_1	out	$\delta(\psi_1)$	$\delta(\psi_2)$	$\delta(\Gamma)$	$\delta(\Gamma')$
ν_1	t	t	t	t	t	0	1	**0**	**1**
ν_2	t	t	t	t	f	1	0	1	**1**
ν_3	t	t	t	f	f	1	1	1	2
ν_4	t	t	f	t	f	1	0	1	**1**
ν_5	t	t	f	f	f	1	1	1	2
ν_6	t	f	t	f	f	1	1	1	2
ν_7	t	f	f	f	f	1	1	1	2
ν_8	f	t	t	f	f	1	1	1	2
ν_9	f	t	f	f	f	1	1	1	2
ν_{10}	f	f	t	f	f	1	1	1	2
ν_{11}	f	f	f	f	f	0	1	**0**	**1**

Thus, $\Delta_{\mathcal{S}}(\Gamma) = \{\nu_1, \nu_{11}\}$ and $\Delta_{\mathcal{S}}(\Gamma') = \{\nu_1, \nu_2, \nu_4, \nu_{11}\}$.

We turn now to the proofs of the propositions in the paper: Proposition 1 and Proposition 2 are easy. The proofs of Propositions 3, 5, and 6 appear in [6]. The proof of Proposition 4 is a variation of the proof of Proposition 39 in [6]. Below, we show the other results:

Proof of Proposition 7: Let $\Gamma = \{\psi_1, \ldots, \psi_n\}$ and $\mu \in \Lambda_{\mathcal{M}}$, s.t. $\mu(\psi) = f$. As $\Gamma \models_{\mathcal{S}} \psi$, $\mu \notin \Delta_{\mathcal{S}}(\Gamma)$, so there is $\nu \in \Delta_{\mathcal{S}}(\Gamma)$ with $\delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \Gamma) < \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \Gamma)$, i.e., $f(\{d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_1), \ldots, d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_n)\}) < f(\{d^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \psi_1), \ldots, d^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \psi_n)\})$. As $\Gamma \models_{\mathcal{S}} \psi$, it follows that $\nu(\psi) = t$. Now, as $\mathsf{Atoms}(\Gamma \cup \{\psi\}) \cap \mathsf{Atoms}(\{\phi\}) = \emptyset$, one can easily define an $\mathcal{M}$-valuation σ such that $\sigma(\varphi) = \nu(\varphi)$ for every $\varphi \in \mathsf{SF}(\Gamma \cup \{\psi\})$ and $\sigma(\varphi) = \mu(\varphi)$ for every $\varphi \in \mathsf{SF}(\{\phi\})$. By Proposition 2, and since f is hereditary, we have:

$$\begin{aligned}
\delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\sigma, \Gamma \cup \{\phi\}) &= f(\{d^{\downarrow \mathsf{SF}(\Gamma)}(\sigma, \psi_1), \ldots, d^{\downarrow \mathsf{SF}(\Gamma)}(\sigma, \psi_n), d^{\downarrow \mathsf{SF}(\Gamma)}(\sigma, \phi)\}) \\
&= f(\{d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_1), \ldots, d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_n), d^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \phi)\}) \\
&< f(\{d^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \psi_1), \ldots, \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \psi_n), d^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \phi)\}) \\
&= \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \Gamma \cup \{\phi\})
\end{aligned}$$

Thus, for every $\mu \in \Lambda_{\mathcal{M}}$ such that $\mu(\psi) = f$, there is some $\sigma \in \Lambda_{\mathcal{M}}$ such that $\sigma(\psi) = t$ and $\delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\sigma, \Gamma \cup \{\phi\}) < \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\mu, \Gamma \cup \{\phi\})$. It follows that the elements of $\Delta_{\mathcal{S}}(\Gamma \cup \{\phi\})$ must satisfy ψ, and so $\Gamma, \phi \models_{\mathcal{S}} \psi$. □

Proof of Proposition 8: Cautious reflexivity follows from Proposition 3. The proofs for cautious monotonicity and cautious transitivity are an adaptation of the ones for the deterministic case (see [3]):

For cautious monotonicity, let $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ and suppose that $\Gamma \models_{\mathcal{S}} \psi$, $\Gamma \models_{\mathcal{S}} \phi$, and $\nu \in \Delta_{\mathcal{S}}(\Gamma \cup \{\psi\})$. We show that $\nu \in \Delta_{\mathcal{S}}(\Gamma)$ and since $\Gamma \models_{\mathcal{S}} \phi$ this implies that $\nu \in mod_{\mathcal{M}}(\{\phi\})$. Indeed, if $\nu \notin \Delta_{\mathcal{S}}(\Gamma)$, there is a valuation $\mu \in \Delta_{\mathcal{S}}(\Gamma)$ so that $\delta_{d,f}(\mu, \Gamma) < \delta_{d,f}(\nu, \Gamma)$, i.e., $f(\{d(\mu, \gamma_1), \ldots, d(\mu, \gamma_n)\}) < f(\{d(\nu, \gamma_1), \ldots, d(\nu, \gamma_n)\})$. Also, as $\Gamma \models_{\mathcal{S}} \psi$, $\mu \in mod_{\mathcal{M}}(\{\psi\})$, thus $d(\mu, \psi) = 0$. By these facts, then,

$$\begin{aligned}\delta_{d,f}(\mu, \Gamma \cup \{\psi\}) &= f(\{d(\mu, \gamma_1), \ldots, d(\mu, \gamma_n), 0\}) \\ &< f(\{d(\nu, \gamma_1), \ldots, d(\nu, \gamma_n), 0\}) \\ &\leq f(\{d(\nu, \gamma_1), \ldots, d(\nu, \gamma_n), d(\nu, \psi)\}) = \delta_{d,f}(\nu, \Gamma \cup \{\psi\}),\end{aligned}$$

a contradiction to $\nu \in \Delta_{\mathcal{S}}(\Gamma \cup \{\psi\})$.

For cautious transitivity, let again $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ and assume that $\Gamma \models_{\mathcal{S}} \psi$, $\Gamma, \psi \models_{\mathcal{S}} \phi$, and $\nu \in \Delta_{\mathcal{S}}(\Gamma)$. We have to show that $\nu \in mod_{\mathcal{M}}(\{\phi\})$. Indeed, since $\nu \in \Delta_{\mathcal{S}}(\Gamma)$, for all $\mu \in \Lambda_{\mathcal{M}}$, $f(\{d(\nu, \gamma_1), \ldots, d(\nu, \gamma_n)\}) \leq f(\{d(\mu, \gamma_1), \ldots, d(\mu, \gamma_n)\})$. Moreover, since $\Gamma \models_{\mathcal{S}} \psi$, $\nu \in mod_{\mathcal{M}}(\{\psi\})$, and so $d(\nu, \psi) = 0 \leq d(\mu, \psi)$. It follows, then, that for every $\mu \in \Lambda_{\mathcal{M}}$,

$$\begin{aligned}\delta_{d,f}(\nu, \Gamma \cup \{\psi\}) &= f(\{d(\nu, \gamma_1), \ldots, d(\nu, \gamma_n), d(\nu, \psi)\}) \\ &\leq f(\{d(\mu, \gamma_1), \ldots, d(\mu, \gamma_n), d(\nu, \psi)\}) \\ &\leq f(\{d(\mu, \gamma_1), \ldots, d(\mu, \gamma_n), d(\mu, \psi)\}) = \delta_{d,f}(\mu, \Gamma \cup \{\psi\}).\end{aligned}$$

Thus, $\nu \in \Delta_{\mathcal{S}}(\Gamma \cup \{\psi\})$, and since $\Gamma, \psi \models_{\mathcal{S}} \phi$, necessarily $\nu \in mod_{\mathcal{M}}(\{\phi\})$. □

Proof outline of Proposition 9: Suppose that $\mathcal{S}$ is a computable setting. By Definition 17, in order to check whether $\Gamma \models_{\mathcal{S}} \psi$, one has to check whether $\Delta_{\mathcal{S}}(\Gamma) \subseteq mod_{\mathcal{M}}(\psi)$. For decidability, we show that this condition, which involves infinite sets, can be reduced to an equivalent condition in terms of finite sets. For this, we denote by $mod_{\mathcal{M}}^{\downarrow \mathsf{C}}(\psi)$ the set $\{\mu^{\downarrow \mathsf{C}} \mid \mu \in mod_{\mathcal{M}}(\psi)\}$. Next, we extend the notions of distance between a valuation and a formula and distance between a valuation and a theory to *partial valuations* as follows: for every context C such that $\mathsf{SF}(\Gamma) \subseteq \mathsf{C}$, define, for every $\nu \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{SF}(\Gamma)}$ and every $\psi \in \Gamma$,

- $d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi) =$
$$\begin{cases} \min\{d^{\downarrow \mathsf{SF}(\Gamma)}(\nu^{\downarrow \mathsf{SF}(\Gamma)}, \mu^{\downarrow \mathsf{SF}(\Gamma)}) \mid \mu \in mod_{\mathcal{M}}^{\downarrow \mathsf{C}}(\psi)\} & \text{if } mod_{\mathcal{M}}^{\downarrow \mathsf{C}}(\psi) \neq \emptyset, \\ 1 + \max\{d^{\downarrow \mathsf{SF}(\Gamma)}(\mu_1^{\downarrow \mathsf{SF}(\Gamma)}, \mu_2^{\downarrow \mathsf{SF}(\Gamma)}) \mid \mu_1, \mu_2 \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{C}}\} & \text{otherwise.} \end{cases}$$
- $\delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \Gamma) = f(\{d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_1), \ldots, d^{\downarrow \mathsf{SF}(\Gamma)}(\nu, \psi_n)\})$.

Note that since all the partial valuations involved in the definitions above are defined on finite contexts, there are finitely many such valuations to check, and so $d^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi)$ and $\delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\Gamma)$ are computable for every $\nu \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{C}}$. Now, consider the following set of partial valuations on C:

$$\Delta_{\mathcal{S}}^{\downarrow \mathsf{C}}(\Gamma) = \begin{cases} \{\nu \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{C}} \mid \forall \mu \in \Lambda_{\mathcal{M}}\ \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\Gamma) \leq \delta_{d,f}^{\downarrow \mathsf{SF}(\Gamma)}(\mu,\Gamma)\} & \text{if } \Gamma \neq \emptyset, \\ \Lambda_{\mathcal{M}}^{\downarrow \mathsf{C}} & \text{otherwise.} \end{cases}$$

Clearly, $\Delta_{\mathcal{S}}^{\downarrow \mathsf{C}}(\Gamma)$ and $mod_{\mathcal{M}}^{\downarrow \mathsf{C}}(\psi)$ are computable. Decidability now follows from the fact that $\Delta_{\mathcal{S}}(\Gamma) \subseteq mod_{\mathcal{M}}(\psi)$ if and only if $\Delta_{\mathcal{S}}^{\downarrow \mathsf{C}}(\Gamma) \subseteq mod_{\mathcal{M}}^{\downarrow \mathsf{C}}(\psi)$. □

Proof of Proposition 10: Let $\mu \in \Lambda_{\mathcal{M}}$. As $\Gamma = \{\varphi_1, \ldots, \varphi_n\}$ is not $\mathcal{M}$-satisfiable, μ is not a model of Γ, and so there is some formula $\varphi_j \in \Gamma$ such that $d_U^{\downarrow \mathsf{SF}(\Gamma)}(\mu,\varphi_j) = 1$. Moreover, for every $\varphi_i \in \Gamma$ we have that $d_U^{\downarrow \mathsf{SF}(\Gamma)}(\mu,\varphi_i) \in \{0,1\}$ and so, since f is range-restricted,

$$\delta_{d_U,f}^{\downarrow \mathsf{SF}(\Gamma)}(\mu,\Gamma) = f(\{d_U^{\downarrow \mathsf{SF}(\Gamma)}(\mu,\varphi_1), \ldots, d_U^{\downarrow \mathsf{SF}(\Gamma)}(\mu,\varphi_n)\}) = 1.$$

This shows that all the valuations in $\Lambda_{\mathcal{M}}$ are equally distant from Γ and so $\Delta_{\mathcal{S}}(\Gamma) = \Lambda_{\mathcal{M}}$. Thus, $\Gamma \models_{\mathcal{S}} \psi$ iff $\Delta_{\mathcal{S}}(\Gamma) \subseteq mod_{\mathcal{M}}(\psi)$, iff $mod_{\mathcal{M}}(\psi) = \Lambda_{\mathcal{M}}$, iff ψ is a tautology. □

Proof of Proposition 11: Consider a theory $\Gamma = \{\psi_1, \ldots, \psi_n\}$, and assume first that $\mathsf{mSAT}_{\mathcal{M}}(\Gamma) \neq \emptyset$. Since $\mathcal{S}$ is drastic, for every $\psi \in \Gamma$ and every $\nu \in \Lambda_{\mathcal{M}}$, $d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi) = 0$ if $\nu \in mod_{\mathcal{M}}(\psi)$, and otherwise $d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi) = 1$. Now, since f is additive, we have that

$$\begin{aligned} \delta_{d_U,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\Gamma) &= f\{d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_1), \ldots, d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_n)\} \\ &= g(n) \cdot (d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_1) + \ldots + d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_n)) \\ &= g(n) \cdot |\{\psi \in \Gamma \mid \nu \notin mod_{\mathcal{M}}(\psi)\}|. \end{aligned}$$

Thus, $\nu \in \Delta_{\mathcal{S}}(\Gamma)$ iff the set $\{\psi \in \Gamma \mid \nu \notin mod_{\mathcal{M}}(\psi)\}$ is minimal in its size, iff $\{\psi \in \Gamma \mid \nu \in mod_{\mathcal{M}}(\psi)\}$ is maximal in its size, iff this set belongs to $\mathsf{mSAT}_{\mathcal{M}}(\Gamma)$.

Now assume that $\mathsf{mSAT}_{\mathcal{M}}(\Gamma) = \emptyset$. In this case none of the formulas in Γ is $\mathcal{M}$-satisfiable (see Note 4). Thus, as

$$\mathsf{M}_{d_U}(\Gamma) = \max\{d_U^{\downarrow \mathsf{SF}(\Gamma)}(\mu_1^{\downarrow \mathsf{SF}(\Gamma)}, \mu_2^{\downarrow \mathsf{SF}(\Gamma)}) \mid \mu_1, \mu_2 \in \Lambda_{\mathcal{M}}^{\downarrow \mathsf{SF}(\Gamma)}\} = 1,$$

we have that for every $\nu \in \Lambda_{\mathcal{M}}$,

$$\begin{aligned} \delta_{d_U,f}^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\Gamma) &= f\{d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_1), \ldots, d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_n)\} \\ &= g(n) \cdot (d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_1) + \ldots + d_U^{\downarrow \mathsf{SF}(\Gamma)}(\nu,\psi_n)) \\ &= g(n) \cdot n \cdot (1 + \mathsf{M}_{d_U}(\Gamma)) \\ &= 2n \cdot g(n). \end{aligned}$$

Thus, all the elements in $\Lambda_{\mathcal{M}}$ are equally distant from Γ, and so $\Delta_{\mathcal{S}}(\Gamma) = \Lambda_{\mathcal{M}}$. □

A Propositional Dynamic Logic for CCS Programs

Mario R.F. Benevides[1,2] and L. Menasché Schechter[1]

[1] Systems and Computer Engineering Program, Federal University of Rio de Janeiro, Brazil
{mario,luis}@cos.ufrj.br
[2] Computer Science Department, Federal University of Rio de Janeiro, Brazil

Abstract. This work presents a Propositional Dynamic Logic in which the programs are CCS terms (CCS-PDL). Its goal is to reason about properties of concurrent systems specified in CCS. CCS is a process algebra that models the concurrency and interaction between processes through individual acts of communication. At a first step, we consider only CCS processes without constants and give a complete axiomatization for this logic, which is very similar to $*$-free PDL. Then, we proceed to include CCS processes with constants. In this case, we impose some restrictions on the form of the recursive equations that can be built with those constants. We also give an axiomatization for this second logic and prove its completeness using a Fischer-Ladner construction. Unlike Concurrent PDL (with channels) [1,2], our logic has a simple Kripke semantics, a complete axiomatization and the finite model property.

1 Introduction

Propositional Dynamic Logic (PDL) plays an important role in formal specification and reasoning about programs and actions. PDL is a multi-modal logic with one modality $\langle\pi\rangle$ for each program π. The logic has a finite set of basic programs and a set of operators (sequential composition, iteration and nondeterministic choice) that can be used to build more complex programs from simpler ones. PDL has been used in formal specification to reason about properties of programs and their behaviour. Correctness, termination, fairness, liveness and equivalence of programs are among the properties that one usually wants to verify. A Kripke semantics can be provided, with a frame $\mathcal{F} = \langle W, R_\pi\rangle$, where W is a set of possible program states and, for each program π, R_π is a binary relation on W such that $(s,t) \in R_\pi$ if and only if there is a computation of π starting in s and terminating in t.

The Calculus for Communicating Systems (CCS) is a well known process algebra, proposed by Robin Milner [3], for the specification of concurrent systems. It models the concurrency and interaction between processes through individual acts of communication. A pair of processes can communicate through a common channel and each act of communication consists simply of a signal being sent at one end of the channel and (immediately) being received at the other.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 83–97, 2008.

A CCS specification is a description (in the form of algebraic equations) of the behaviour expected from a system, based on the communication events that may occur. As in PDL, CCS has a set of operators (action prefix, parallel composition, nondeterministic choice and restriction on acts of communication) to build more complex specifications from simpler ones. Iteration can also be described through the use of recursive equations.

This work presents a Propositional Dynamic Logic in which the programs are CCS terms (CCS-PDL). Its goal is to reason about properties of concurrent systems specified in CCS. The idea is to bring together the logical and the algebraic formalisms. Quoting from Milner [3]:

> "The calculus of this book is indeed largely algebraic, but I make no claim that everything can be done by algebra. (...) It is perhaps equally true that not everything can be done by logic; thus one of the outstanding challenges in concurrency is to find the right marriage between logical and behavioural approaches."

CCS-PDL is related to Concurrent PDL (with channels) [1,2] and the logic developed in [4]. Both of these logics are expressive enough to represent interesting properties of concurrent systems. However, neither of them has a simple Kripke semantics. The first has a semantics based on *super-states and super-processes* and its satisfiability problem can be proved undecidable (in fact, it is Π^1_1-hard) [1]. Also, it does not have a complete axiomatization [1]. The second makes a semantic distinction between *final* and *non-final* states, which makes its semantics and its axiomatization rather complex. On the other hand, due to the full use of the CCS mechanism of communication, CCS-PDL has a simple Kripke semantics and the finite model property.

The rest of this paper is organized as follows. In section 2, we introduce the necessary background concepts: Propositional Dynamic Logic and the Calculus for Communicating Systems. A first version of our logic, together with an axiomatic system, is presented in section 3. In this preliminary version, we do not use constants in the CCS processes. In section 4, we present the full logic, in which we allow the presence of constants in the CCS processes. We also give an axiomatization for this second logic and prove its completeness using a Fischer-Ladner construction. Finally, in section 5, we state our final remarks.

2 Background

This section presents two important subjects. First, we make a brief review of the syntax and semantics of PDL. Second, we present the process algebra CCS and the Expansion Law, which is closely related to the way we deal with the parallel composition operator.

2.1 Propositional Dynamic Logic

In this section, we present the syntax and semantics of PDL.

Definition 1. *The PDL language consists of a set Φ of countably many propositional symbols, a finite set of basic programs Π, the boolean connectives $\neg$ and $\wedge$, the program constructors ;, $\cup$ and $*$ and a modality $\langle\pi\rangle$ for every program π. The formulas are defined as follows:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\pi\rangle\varphi, \text{ with } \pi ::= a \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \ ,$$

where $p \in \Phi$ and $a \in \Pi$.

In this logic and in every other logic defined in this paper, we use the standard abbreviations $\bot \equiv \neg\top$, $\varphi \vee \phi \equiv \neg(\neg\varphi \wedge \neg\phi)$, $\varphi \rightarrow \phi \equiv \neg(\varphi \wedge \neg\phi)$ and $[\pi]\varphi \equiv \neg\langle\pi\rangle\neg\varphi$.

Definition 2. *A* frame *for PDL is a tuple $\mathcal{F} = (W, R_a, R_\pi)$ where*

- *W is a non-empty set of states;*
- *R_a is a binary relation for each basic program a;*
- *R_π is a binary relation for each non-basic program π, inductively built using the rules $R_{\pi_1;\pi_2} = R_{\pi_1} \circ R_{\pi_2}$, $R_{\pi_1\cup\pi_2} = R_{\pi_1} \cup R_{\pi_2}$ and $R_{\pi^*} = R^*_\pi$, where R^*_π denotes the reflexive transitive closure of R_π.*

Definition 3. *A* model *for PDL is a pair $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, where $\mathcal{F}$ is a PDL frame and $\mathbf{V}$ is a valuation function $\mathbf{V} : \Phi \mapsto 2^W$.*

The semantical notion of satisfaction for PDL is defined as follows:

Definition 4. *Let $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ be a model. The notion of* satisfaction *of a formula φ in a model $\mathcal{M}$ at a state w, notation $\mathcal{M}, w \Vdash \varphi$, can be inductively defined as follows:*

- *$\mathcal{M}, w \Vdash p$ iff $w \in \mathbf{V}(p)$;*
- *$\mathcal{M}, w \Vdash \top$ always;*
- *$\mathcal{M}, w \Vdash \neg\varphi$ iff $\mathcal{M}, w \nVdash \varphi$;*
- *$\mathcal{M}, w \Vdash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, w \Vdash \varphi_1$ and $\mathcal{M}, w \Vdash \varphi_2$;*
- *$\mathcal{M}, w \Vdash \langle\pi\rangle\varphi$ iff there is $w' \in W$ such that $wR_\pi w'$ and $\mathcal{M}, w' \Vdash \varphi$.*

2.2 Calculus for Communicating Systems

The Calculus for Communicating Systems (CCS) is a well known process algebra, proposed by Robin Milner [3], for the specification of concurrent systems. It models the concurrency and interaction between processes through individual actions of communication. A CCS specification is a description (in the form of algebraic equations) of the behaviour expected from a system, based on the communication events that may occur.

In CCS, a pair of processes can communicate through a common channel and each act of communication consists simply of a signal being sent at one end of the channel and (immediately) being received at the other. Each process connects itself to a channel through a port.

Let $\mathcal{N} = \{a, b, c, \ldots\}$ be a set of names and $\overline{\mathcal{N}} = \{\overline{a}, \overline{b}, \overline{c}, \ldots\}$ be the correspondent set of co-names. Each port in a CCS specification is labelled by either a name or a co-name. Using the convention that $\overline{\overline{a}} = a$, if a port connected to an end of a channel is labelled with $\alpha \in \mathcal{N} \cup \overline{\mathcal{N}}$, then a port connected to the other end of this same channel must be labelled with $\overline{\alpha}$. Moreover, two ports of a process cannot have the same label. Channels in CCS can only transmit signals in one direction. By convention, signals are sent through the ports labelled by co-names and received through ports labelled by names.

The labels of the ports are also used to describe the communication actions performed by the processes. For example, if a process is connected to a channel through port $\overline{\alpha}$, the action of sending a signal into this channel is also denoted by $\overline{\alpha}$. So, in terms of actions, α means that the process receives a signal through the port labelled with α and $\overline{\alpha}$ means that the process sends a signal through the port labelled with $\overline{\alpha}$. A process can never perform a communication action α without its complementary action $\overline{\alpha}$ also being performed at the same time by some other process.

Besides the actions denoted by the elements of $\mathcal{N} \cup \overline{\mathcal{N}}$, CCS admits only one other action: the silent action, denoted by τ. The silent action is used to represent any internal action performed by any of the processes that does not involve any act of communication (e.g.: a memory update). Thus, we have that the set of CCS actions is $\mathcal{A} = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$.

There are two different ways in CCS to consider the τ action: it can be regarded as being observable, in the same way as the communication actions, or it can be regarded as being invisible. We will adopt the first point of view, since it simplifies the semantics considerably.

In CCS, process specifications can be built using the following operations ($\alpha \in \mathcal{A}$ is an action, P, P_1 and P_2 are process specifications, A is a constant and $L \subseteq \mathcal{N}$):

$$P ::= \alpha \mid \alpha.P \mid \alpha.A \mid P_1 + P_2 \mid P_1|P_2 \mid P\backslash L \ ,^{1}$$

where every constant A has a (unique) *defining equation* $A \stackrel{def}{=} P_A$, where P_A is a process specification. In this work, every time that a process is linked to a constant A through a defining equation, it will be denoted by P_A.

The *prefix* operator (.) denotes that the process will first perform the action α and then behave as P or A. The *summation* (or *nondeterministic choice*) operator (+) denotes that the process will make a nondeterministic choice to behave as either P_1 or P_2. The *parallel composition* operator (|) denotes that the processes P_1 and P_2 may proceed independently or may communicate through a pair of complementary ports (one performing an action α and the other $\overline{\alpha}$). Finally, the *restriction* operator ($\backslash$) denotes that for all ports α such that $\alpha \in L$ or $\overline{\alpha} \in L$, α is unreachable outside P. Iteration in CCS is modeled through recursive defining equations, i.e., equations $A \stackrel{def}{=} P_A$ where A occurs in P_A.

We write $P \stackrel{\alpha}{\rightarrow} P'$ to express that the process P can perform the action α and after that behave as P'. We write $P \stackrel{\alpha}{\rightarrow} \mathbf{0}$ to express that the process P

[1] Originally, CCS has also a relabelling operator.

Table 1. Transition Relations of CCS

Action	$\frac{}{\alpha.\mathbf{0} \xrightarrow{\alpha} \mathbf{0}}$	Prefix	$\frac{}{\alpha.P \xrightarrow{\alpha} P}$	Constant	$\frac{A \stackrel{def}{=} P_A}{\alpha.A \xrightarrow{\alpha} P_A}$
Summation (1)	$\frac{E \xrightarrow{\alpha} E'}{E+F \xrightarrow{\alpha} E'}$	Summation (2)	$\frac{F \xrightarrow{\beta} F'}{E+F \xrightarrow{\beta} F'}$	Restriction	$\frac{E \xrightarrow{\alpha} E', \alpha, \overline{\alpha} \notin L}{E\backslash L \xrightarrow{\alpha} E'\backslash L}$
Par. Comp. (1)	$\frac{E \xrightarrow{\alpha} E'}{E\|F \xrightarrow{\alpha} E'\|F}$	Par. Comp. (2)	$\frac{F \xrightarrow{\beta} F'}{E\|F \xrightarrow{\beta} E\|F'}$	Par. Comp. (3)	$\frac{E \xrightarrow{\lambda} E', F \xrightarrow{\overline{\lambda}} F'}{E\|F \xrightarrow{\tau} E'\|F'}$

finishes after performing the action α. A process only finishes when there is not any possible action left for it to perform. For example, $\beta \xrightarrow{\beta} \mathbf{0}$. When a process finishes inside a parallel composition, we write P instead of $P|\mathbf{0}$. In table 1, we present the semantics for the operators based on this notation.

The notion of equality between process specifications is defined through the concept of *strong bisimulation*.

Definition 5. *Let $\mathcal{P}$ be the set of all possible process specifications. A set $Z \subseteq \mathcal{P} \times \mathcal{P}$ is a* strong bisimulation *if $(P, Q) \in Z$ implies, for all $\alpha \in \mathcal{A}$,*

- *Whenever $P \xrightarrow{\alpha} P'$ then, for some Q', $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in Z$*
- *Whenever $Q \xrightarrow{\alpha} Q'$ then, for some P', $P \xrightarrow{\alpha} P'$ and $(P', Q') \in Z$*

Definition 6. *Two process specifications P and Q are* strongly bisimilar *(or simply* bisimilar*), denoted by $P \sim Q$, if there is a strong bisimulation Z such that $(P, Q) \in Z$. Two process specifications are considered "equal" if they are bisimilar.*

Bisimilarity is preserved by all of CCS operators:

Theorem 1 ([3]). *Let $P_1 \sim P_2$. Then*

1. $\alpha.P_1 \sim \alpha.P_2$;
2. $P_1 + Q \sim P_2 + Q$;
3. $P_1|Q \sim P_2|Q$;
4. $P_1\backslash L \sim P_2\backslash L$.

A very useful property of CCS processes is that they can be rewritten as a summation of all their possible actions. This is what states the Expansion Law below. This theorem will be very important in the definition of the semantics of our logic in the next section.

Theorem 2 (Expansion Law [3]). *Let $P = (P_1 \mid P_2)$. Then*

$$P \sim \sum\{\alpha.(P_1' \mid P_2) : P_1 \xrightarrow{\alpha} P_1'\} + \sum\{\alpha.(P_1 \mid P_2') : P_2 \xrightarrow{\alpha} P_2'\} +$$

$$+ \sum\{\tau.(P_1' \mid P_2') : P_1 \xrightarrow{\lambda} P_1', P_2 \xrightarrow{\overline{\lambda}} P_2'\} \ .$$

In the rest of this paper, we consider that the restriction operator does not occur in the processes. In order to motivate the use of CCS, we present a simple example below:

Example 1 (Vending Machine [3,5]). Consider a vending machine where one can put coins of one or two euro and buy a little or a big chocolate bar. After inserting the coins, one must press the little button for a little chocolate or the big button for a big chocolate. The machine is also programmed to shutdown on its own following some internal logic (represented by a τ action). A CCS term describing the behaviour of this machine is the following:

$$V = 1e.little.\overline{collect}.A + 1e.1e.big.\overline{collect}.A + 2e.big.\overline{collect}.A$$

$$A \stackrel{def}{=} 1e.little.\overline{collect}.A + 1e.1e.big.\overline{collect}.A + 2e.big.\overline{collect}.A + \tau$$

Let us now suppose that Chuck wants to use this vending machine. We could describe Chuck as

$$C = \overline{1e}.\overline{little}.collect + \overline{1e}.\overline{1e}.\overline{big}.collect + \overline{2e}.\overline{big}.collect \ .$$

Notice that Chuck does not have an iterative behaviour. Once he collects the chocolate, he is done. Now, if we want to model the process of Chuck buying a chocolate from the vending machine, we could write $(V|C)$.

3 PDL for CCS Programs without Constants

This section presents a suitable fragment of CCS-PDL. In this fragment, all CCS processes do not use constants. We call this fragment Small CCS-PDL or SCCS-PDL. Our goal is to introduce a smaller version of our logic and discuss some of the issues concerning the axioms and the relational interpretation of formulas.

3.1 Language and Semantics

Definition 7. *The SCCS-PDL language consists of a set Φ of countably many propositional symbols, a finite set of actions $\mathcal{A}$ that includes the silent action τ, the boolean connectives $\neg$ and $\wedge$, the CCS operators ., + and | and a modality $\langle P\rangle$ for every process P. The formulas are defined as follows:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle P\rangle\varphi, \ \textit{with} \ P ::= \alpha \mid \alpha.P \mid P_1 + P_2 \mid P_1|P_2 \ ,$$

where $p \in \Phi$ and $\alpha \in \mathcal{A}$.

Going back to the example of the vending machine, we could use this language to express that after the insertion of two one euro coins, the machine does not accept coins anymore. This can be expressed with the formula $[1e.1e][1e+2e]\bot$.

Definition 8. *We define the length of a process P, denoted by $||P||$, as the number of symbols in P. We also define the length of a finished process as 0.*

Theorem 3. *If P is a process without any constants and $P \stackrel{\alpha}{\rightarrow} P'$, then $||P'|| < ||P||$.*

Definition 9. *A* frame *for SCCS-PDL is a tuple* $\mathcal{F} = (W, R_\alpha, R_P)$ *where*

- W *is a non-empty set of states;*
- R_α *is a binary relation for each basic action* α*, including* τ*;*
- R_P *is a binary relation for each non-basic process* P*, inductively built as:*
  - $R_{\alpha.P} = R_\alpha \circ R_P$*;*
  - $R_{(P_1+P_2)} = R_{P_1} \cup R_{P_2}$*;*
  - $R_{(P_1|P_2)} = \bigcup_{P_1 \xrightarrow{\alpha} P_1'} R_\alpha \circ R_{(P_1'|P_2)} \cup \bigcup_{P_2 \xrightarrow{\alpha} P_2'} R_\alpha \circ R_{(P_1|P_2')} \cup \bigcup_{\substack{P_1 \xrightarrow{\lambda} P_1' \\ P_2 \xrightarrow{\overline{\lambda}} P_2'}} R_\tau \circ R_{(P_1'|P_2')}$

It should be noticed that the Expansion Law stated in theorem 2 is what allows us to define a simple relational semantic to the parallel composition operator. Besides that, theorem 3 guarantees that we can fully define the relation R_P in terms of the relations R_α, for any process P, applying the above rules a finite number of times.

Definition 10. *A* model *for SCCS-PDL is a pair* $\mathcal{M} = (\mathcal{F}, \mathbf{V})$*, where* $\mathcal{F}$ *is a SCCS-PDL frame and* $\mathbf{V}$ *is a valuation function* $\mathbf{V} : \Phi \mapsto 2^W$*.*

The semantical notion of satisfaction for SCCS-PDL is defined as follows:

Definition 11. *Let* $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ *be a model. The notion of* satisfaction *of a formula* φ *in a model* $\mathcal{M}$ *at a state* w*, notation* $\mathcal{M}, w \Vdash \varphi$*, can be inductively defined as follows:*

- $\mathcal{M}, w \Vdash p$ *iff* $w \in \mathbf{V}(p)$*;*
- $\mathcal{M}, w \Vdash \top$ *always;*
- $\mathcal{M}, w \Vdash \neg\varphi$ *iff* $\mathcal{M}, w \nVdash \varphi$*;*
- $\mathcal{M}, w \Vdash \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}, w \Vdash \varphi_1$ *and* $\mathcal{M}, w \Vdash \varphi_2$*;*
- $\mathcal{M}, w \Vdash \langle P \rangle \varphi$ *iff there is* $w' \in W$ *such that* $w R_P w'$ *and* $\mathcal{M}, w' \Vdash \varphi$*.*

If $\mathcal{M}, w \Vdash \varphi$ for every state w, we say that φ is *globally satisfied* in the model $\mathcal{M}$, notation $\mathcal{M} \Vdash \varphi$. If φ is globally satisfied in all models $\mathcal{M}$ of a frame $\mathcal{F}$, we say that φ is *valid* in $\mathcal{F}$, notation $\mathcal{F} \Vdash \varphi$. Finally, if φ is valid in all frames, we say that φ is valid, notation $\Vdash \varphi$.

3.2 Proof Theory

We consider the following set of axioms and rules, where p and q are proposition symbols and φ and ψ are formulas.

(PL) Enough propositional logic tautologies
(K) $[P](p \to q) \to ([P]p \to [P]q)$
(Pr) $\langle \alpha.P \rangle p \leftrightarrow \langle \alpha \rangle \langle P \rangle p$,
(NC) $\langle P_1 + P_2 \rangle p \leftrightarrow \langle P_1 \rangle p \vee \langle P_2 \rangle p$

(PC) $\langle P_1 \mid P_2\rangle p \leftrightarrow \bigvee_{P_1 \xrightarrow{\alpha} P_1'} \langle\alpha\rangle\langle P_1' \mid P_2\rangle p \vee \bigvee_{P_2 \xrightarrow{\alpha} P_2'} \langle\alpha\rangle\langle P_1 \mid P_2'\rangle p \vee \vee \bigvee_{\substack{P_1 \xrightarrow{\lambda} P_1' \\ P_2 \xrightarrow{\overline{\lambda}} P_2'}} \langle\tau\rangle\langle P_1' \mid P_2'\rangle p$

(Sub) If $\vdash \varphi$, then $\vdash \varphi^\sigma$, where σ uniformly substitutes proposition symbols by arbitrary formulas.

(MP) If $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$, then $\vdash \psi$.

(Gen) If $\vdash \varphi$, then $\vdash [P]\varphi$.

These axioms are closely related to the conditions imposed in Definition 9. The second axiom is the K axiom for modal logics. The third and fourth axioms are well-known from PDL literature and define sequential composition and choice operators respectively. The fifth axiom corresponds to the Expansion Law for the parallel composition operator.

The logic presented in this section is sound and complete w.r.t. the class of frames described in Definition 9. It also has the finite model property. We omit the proofs here, because they are analogous to the proofs presented in section 4, where constants are added to the language.

4 PDL for CCS Programs

The logic presented in this section uses the same CCS operators as in the previous section plus constants. This is the full CCS-PDL logic. Our goal in this section is to build an axiomatic system to CCS-PDL and prove its completeness.

4.1 Language and Semantics

Definition 12. *The CCS-PDL language consists of a set Φ of countably many propositional symbols, a finite set of actions $\mathcal{A}$ that includes the silent action τ, the boolean connectives $\neg$ and $\wedge$, the CCS operators $.$, $+$ and $|$, constants, with its correspondent defining equations, and a modality $\langle P\rangle$ for every process P. The formulas are defined as follows:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle P\rangle\varphi, \text{ with } P ::= \alpha \mid \alpha.P \mid \alpha.A \mid P_1 + P_2 \mid P_1|P_2 \ ,$$

where $p \in \Phi$ and $\alpha \in \mathcal{A}$.

Let P be a process and $\{A_1, \ldots, A_n\}$ be the constants that occur in P. We define $Cons(P)$ as the smallest set of constants such that $Cons(P) \supseteq \{A_1, \ldots, A_n\}$ and, for every constant $A_i \in Cons(P)$, if A_k occurs in P_{A_i}, then $A_k \in Cons(P)$. We make the restriction that $Cons(P)$ must be a finite set for every process P.

Another restriction concerns the construction of defining equations. We only allow defining equations that fit into one of the following models: $A \stackrel{def}{=} P_A$, where $A \notin Cons(P_A)$, called *non-recursive equations*, or $A \stackrel{def}{=} \overrightarrow{\alpha}_1.A + \ldots + \overrightarrow{\alpha}_n.A + T_A$, called *recursive equations*, where $\overrightarrow{\alpha}_i$ denotes a sequence of actions $\alpha_1^i.\alpha_2^i \ldots \alpha_{m_i}^i$ and $A \notin Cons(T_A)$.

Definition 13. *We say that a process P is a* looping process *if $P = P_A$ for some constant A with a recursive defining equation or if $P = P_1 \mid P_2$ where P_1 or P_2 is a looping process. Otherwise, we say that P is a* non-looping process.

Definition 14. *We write $P \stackrel{\overrightarrow{\alpha}}{\Rightarrow} P'$ to express that the process P can perform the sequence of actions $\overrightarrow{\alpha}$ and after that behave as P'. We write $P \stackrel{\overrightarrow{\alpha}}{\Rightarrow} \mathbf{0}$ to express that the process P finishes after performing the sequence of actions $\overrightarrow{\alpha}$.*

Definition 15. *We call a sequence $\overrightarrow{\alpha}$ a* breaker *for a looping process P if $P = P_A$, $P \stackrel{\overrightarrow{\alpha}}{\Rightarrow} P'$ and $A \notin Cons(P')$ (or $P \stackrel{\overrightarrow{\alpha}}{\Rightarrow} \mathbf{0}$) or if $P = P_1 \mid P_2$, P_i ($i \in \{1,2\}$) is a looping process and $\overrightarrow{\alpha}$ is a breaker for P_i. These sequences are called breakers because after performing $\overrightarrow{\alpha}$, there is no sequence of actions $\overrightarrow{\alpha}'$ such that $P' \stackrel{\overrightarrow{\alpha}'}{\Rightarrow} P$, i. e., the loop around P is broken by $\overrightarrow{\alpha}$.*

Using the concept of a breaker, we can split a looping process P into two parts: the *looping part*, denoted by L_P, and the *tail part*, denoted by T_P. Informally, L_P describes one loop around P and T_P describes the behaviour of P when it stops looping. Let $Br(P)$ be the set of all breakers for P. Then,

$$T_P = \sum\{\overrightarrow{\alpha}.P' : \overrightarrow{\alpha} \in Br(P), P \stackrel{\overrightarrow{\alpha}}{\Rightarrow} P' \text{ and } \forall \overrightarrow{\alpha}' \subsetneq \overrightarrow{\alpha}, P \stackrel{\overrightarrow{\alpha}'}{\not\Rightarrow} P\} \text{ and}$$

$$L_P = \sum\{\overrightarrow{\alpha} : P \stackrel{\overrightarrow{\alpha}}{\Rightarrow} P \text{ and } \forall \overrightarrow{\alpha}' \subsetneq \overrightarrow{\alpha}, P \stackrel{\overrightarrow{\alpha}'}{\not\Rightarrow} P\} \ .$$

If $P = P_A = \overrightarrow{\alpha}_1.A + \ldots + \overrightarrow{\alpha}_n.A + T_A$, it is not difficult to see that $L_P = \overrightarrow{\alpha}_1 + \ldots + \overrightarrow{\alpha}_n$ and $T_P = T_A$. We also define the process L'_P, that describes one or more loops around P, as $L'_P = \sum\{\overrightarrow{\alpha}.Z_P : P \stackrel{\overrightarrow{\alpha}}{\Rightarrow} P \text{ and } \forall \overrightarrow{\alpha}' \subsetneq \overrightarrow{\alpha}, P \stackrel{\overrightarrow{\alpha}'}{\not\Rightarrow} P\} + L_P$, where Z_P is a new constant with defining equation $Z_P \stackrel{def}{=} L'_P$. It is important to notice that $Cons(L_P) \subsetneq Cons(P)$ and $Cons(T_P) \subsetneq Cons(P)$.

Definition 16. *We say that a process* can unfold a constant *if it has the form $\alpha.A$ or if it has the form $P_1 + P_2$ or $P_1 \mid P_2$, where P_1 or P_2 can unfold a constant.*

Theorem 4. *If P is a process that cannot unfold a constant and $P \stackrel{\alpha}{\rightarrow} P'$, then $||P'|| < ||P||$.*

Definition 17. *A* **frame** *for CCS-PDL is a tuple $\mathcal{F} = (W, R_\alpha, R_P)$ where:*

- *W is a non-empty set of states;*
- *R_α is a binary relation for each basic action α, including τ;*
- *R_P is a binary relation for each non-basic process P, inductively built as:*
  - *For* **looping processes***: $R_P = R^*_{L_P} \circ R_{T_P}$*
  - *For* **non-looping processes***:*
    - *$R_{\alpha.P}$, $R_{P_1+P_2}$ and $R_{P_1|P_2}$ as in definition 9;*
    - *$R_{\alpha.A} = R_\alpha \circ R_{P_A}$*

It should be noticed that the restriction on the set $Cons(P)$ and the restriction on the formation of the defining equations, both presented in the beginning of the section, together with the definition of the relation R_P for looping processes P based on R_{L_P} and R_{T_P} make it impossible for the relations to be able to unfold constants forever. This, together with theorem 4, guarantee that we can fully define the relation R_P in terms of the relations R_α, for any process P, applying the above rules a finite number of times and that we can build well-founded proofs by induction on the structure of a process P.

The notions of models and satisfaction are defined analogously to Definitions 10 and 11.

4.2 Proof Theory

The proof theory is similar to the one presented in section 3.2. We consider the following set of axioms and rules, where p, q and r are proposition symbols and φ and ψ are formulas.

- The axioms **(PL)** and **(K)** and the rules **(Sub)**, **(MP)** and **(Gen)**.
- Axioms for looping processes:
  (Rec) $\langle P\rangle p \leftrightarrow \langle T_P\rangle p \vee \langle L_P\rangle\langle P\rangle p$
  (FP) $(r \rightarrow ([T_P]\neg p \wedge [L_P]r)) \wedge [L'_P](r \rightarrow ([T_P]\neg p \wedge [L_P]r)) \rightarrow (r \rightarrow [P]\neg p)$
- Axioms for non-looping processes:
  (SCCS) The axioms **(Pr)**, **(NC)** and **(PC)**.
  (Cons) $\langle \alpha.A\rangle p \leftrightarrow \langle \alpha\rangle\langle P_A\rangle p$

The proof of soundness is analogous to the proof of soundness for PDL and CTL, as all of our axioms are very closely related to axioms for these logics.

Theorem 5 (Completeness). *Every consistent formula is satisfiable in a finite model that respects definition 17.*

Proof. See the appendix. □

5 Final Remarks and Future Work

In this work, we present a PDL-like logic in which the programs are CCS terms (CCS-PDL). We provide a simple Kripke semantics for it and also give an axiomatization for this logic. We prove the completeness of the axiomatic system and the finite model property for the logic using a Fischer-Ladner construction.

As a continuation of this work, it would be interesting to study the complexity of the satisfiability problem for this logic, possibly relating it to the satisfiability problem for standard PDL.

We would also like to investigate some extension of CCS-PDL to deal with the restriction operator and a PDL for π-Calculus programs [6]. It would be interesting to develop an automatic theorem prover for CCS-PDL. This would involve, among other things, efficient algorithmic methods to determine the processes L_P and T_P related to a looping process P and to deal with the expansion of parallel processes.

Acknowledgements. The first author was partially supported by a grant from CNPq. The second author was supported by a D.Sc. scholarship from CNPq.

References

1. Peleg, D.: Communication in Concurrent Dynamic Logic. Journal of Computer and System Sciences 35, 23–58 (1987)
2. Peleg, D.: Concurrent Dynamic Logic. Journal of the ACM 34, 450–479 (1987)
3. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
4. dos Santos, V.L.P.: Concorrência e Sincronização para Lógica Dinâmica de Processos. PhD thesis, Federal University of Rio de Janeiro (2005)
5. Stirling, C.: Modal and Temporal Properties of Processes. Springer, Heidelberg (2001)
6. Milner, R.: Communicating and Mobile Systems: the π-Calculus. Cambridge University Press, Cambridge (1999)

A Completeness Proof

Definition 18. *Let ϕ be a formula. We define the formula $\overline{\phi}$ as $\overline{\phi} = \psi$, if $\phi = \neg\psi$, or $\overline{\phi} = \neg\phi$, otherwise.*

Definition 19 (Fischer-Ladner Closure). *Let Γ be a set of formulas. The* Fischer-Ladner Closure *of Γ, notation $C(\Gamma)$, is the smallest set of formulas that contains Γ and satisfies the following conditions:*

- *$C(\Gamma)$ is closed under sub-formulas;*
- *if $\phi \in C(\Gamma)$, then $\overline{\phi} \in C(\Gamma)$;*
- *For looping processes:*
  - *If $\langle P\rangle\varphi \in C(\Gamma)$, then $\langle T_P\rangle\varphi \vee \langle L_P\rangle\langle P\rangle\varphi \in C(\Gamma)$.*
- *For non-looping processes:*
  - *If $\langle \alpha.P\rangle\varphi \in C(\Gamma)$, then $\langle\alpha\rangle\langle P\rangle\varphi \in C(\Gamma)$;*
  - *If $\langle \alpha.A\rangle\varphi \in C(\Gamma)$, then $\langle\alpha\rangle\langle P_A\rangle\varphi \in C(\Gamma)$;*
  - *If $\langle P_1 + P_2\rangle\varphi \in C(\Gamma)$, then $\langle P_1\rangle\varphi \vee \langle P_2\rangle\varphi \in C(\Gamma)$;*
  - *If $\langle P_1 \mid P_2\rangle\varphi \in C(\Gamma)$, then $\bigvee_{P_1 \xrightarrow{\alpha} P_1'}\langle\alpha\rangle\langle P_1' \mid P_2\rangle\varphi \vee \bigvee_{P_2 \xrightarrow{\alpha} P_2'}\langle\alpha\rangle\langle P_1 \mid P_2'\rangle\varphi \vee \bigvee_{\substack{P_1 \xrightarrow{\lambda} P_1' \\ P_2 \xrightarrow{\overline{\lambda}} P_2'}}\langle\tau\rangle\langle P_1' \mid P_2'\rangle\varphi \in C(\Gamma)$.*

It is not difficult to prove that if Γ is finite, then the closure $C(\Gamma)$ is also finite. We assume Γ to be finite from now on.

Definition 20. *Every formula ϕ that is derivable from the set of axioms and rules in section 4.2 is called a* theorem *and denoted as $\vdash \phi$. We say that a formula ϕ is* consistent *iff $\neg\phi$ is not a theorem, i.e., iff $\nvdash \neg\phi$ and* inconsistent *otherwise. A set of formulas $\Delta = \{\phi_1, \ldots, \phi_n\}$ is said to be consistent iff $\psi = \phi_1 \wedge \ldots \wedge \phi_n$ is consistent.*

Definition 21. *A set of formulas $\mathcal{A}$ is said to be an* atom *over Γ if it is a maximal consistent subset of $C(\Gamma)$. The set of all atoms over Γ is denoted by $At(\Gamma)$. We denote the conjunction of all the formulas in an atom $\mathcal{A}$ as $\bigwedge \mathcal{A}$.*

Lemma 1. *Every atom $\mathcal{A} \in At(\Gamma)$ has the following properties:*

1. *For every $\phi \in C(\Gamma)$, exactly one of ϕ and $\neg\phi$ belongs to $\mathcal{A}$.*
2. *For every $\phi \wedge \psi \in C(\Gamma)$, $\phi \wedge \psi \in \mathcal{A}$ iff $\phi \in \mathcal{A}$ and $\psi \in \mathcal{A}$.*

Proof. This follows immediately from the definition of atoms as maximal consistent subsets of $C(\Gamma)$. □

Lemma 2. *If $\Delta \subseteq C(\Gamma)$ and Δ is consistent then there exists an atom $\mathcal{A} \in At(\Gamma)$ such that $\Delta \subseteq \mathcal{A}$.*

Proof. We can construct the atom $\mathcal{A}$ as follows. First, we enumerate the elements of $C(\Gamma)$ as $\phi_1, \ldots, \phi_n$. We start the construction making $\mathcal{A}_0 = \Delta$. Then, for $0 \leq i < n$, we know that $\bigwedge \mathcal{A}_i \leftrightarrow (\bigwedge \mathcal{A}_i \wedge \phi_{i+1}) \vee (\bigwedge \mathcal{A}_i \wedge \overline{\phi_{i+1}})$ is a tautology and therefore either $\mathcal{A}_i \cup \{\phi_{i+1}\}$ or $\mathcal{A}_i \cup \{\overline{\phi_{i+1}}\}$ is consistent. We take $\mathcal{A}_{i+1}$ as the consistent extension. At the end, we make $\mathcal{A} = \mathcal{A}_n$. □

Corollary 1. *If $\varphi \in C(\Gamma)$ is a consistent formula, then there is an atom $\mathcal{A} \in At(\Gamma)$ such that $\varphi \in \mathcal{A}$.*

Definition 22 (Canonical model over Γ). *Let Γ be a finite set of formulas. The* canonical model *over Γ is the tuple $\mathcal{M}^\Gamma = (At(\Gamma), \{S_P\}, \mathbf{V})$ where, for all elements $p \in \Phi$, we have $\mathbf{V}(p) = \{\mathcal{A} \in At(\Gamma) \mid p \in \mathcal{A}\}$ and for all atoms $\mathcal{A}, \mathcal{B} \in At(\Gamma)$,*

$$\mathcal{A} S_P \mathcal{B} \quad \textit{iff} \quad \bigwedge \mathcal{A} \wedge \langle P \rangle \bigwedge \mathcal{B} \ \textit{is consistent} \ .$$

$\mathbf{V}$ is called the canonical valuation *and S_P the* canonical relations*, where P is a CCS process.*

Definition 23 (CCS-PDL model over Γ). *The* CCS-PDL model *over Γ is the tuple $\mathcal{N}^\Gamma = (At(\Gamma), \{R_P\}, \mathbf{V})$, where R_P is defined as $R_\alpha = S_\alpha$ for all basic processes α and according to definition 17 for all complex processes. $\mathbf{V}$ is the canonical valuation.*

If $\Gamma = \{\varphi\}$, we write $C(\varphi)$, $At(\varphi)$, $\mathcal{M}^\varphi$ and $\mathcal{N}^\varphi$ instead of $C(\{\varphi\})$, $At(\{\varphi\})$, $\mathcal{M}^{\{\varphi\}}$ and $\mathcal{N}^{\{\varphi\}}$.

Lemma 3 (Existence Lemma for Basic Processes). *Let $\mathcal{A}$ be an atom, and let α be a basic process. Then, for all formulas $\langle\alpha\rangle\phi \in C(\Gamma)$, $\langle\alpha\rangle\phi \in \mathcal{A}$ iff there is a $\mathcal{B} \in At(\Gamma)$ such that $\mathcal{A} R_\alpha \mathcal{B}$ and $\phi \in \mathcal{B}$.*

Proof. ($\Rightarrow$) Suppose $\langle\alpha\rangle\phi \in \mathcal{A}$. We can build an appropriate atom $\mathcal{B}$ by forcing choices. Enumerate the formulas in $C(\Gamma)$ as $\phi_1, \ldots, \phi_n$. Define $\mathcal{B}_0 = \{\phi\}$. Suppose, as an inductive hypothesis that $\mathcal{B}_m$ is defined such that $\bigwedge \mathcal{A} \wedge \langle\alpha\rangle \bigwedge \mathcal{B}_m$ is consistent, for $0 \leq m < n$. We have that

$$\vdash \langle\alpha\rangle \bigwedge \mathcal{B}_m \leftrightarrow \langle\alpha\rangle((\bigwedge \mathcal{B}_m \wedge \phi_{m+1}) \vee (\bigwedge \mathcal{B}_m \wedge \overline{\phi_{m+1}})) \ ,$$

thus

$$\vdash \langle\alpha\rangle \bigwedge \mathcal{B}_m \leftrightarrow (\langle\alpha\rangle(\bigwedge \mathcal{B}_m \wedge \phi_{m+1}) \vee \langle\alpha\rangle(\bigwedge \mathcal{B}_m \wedge \overline{\phi_{m+1}})) \ .$$

Therefore, either for $\mathcal{B}' = \mathcal{B}_m \cup \{\phi_{m+1}\}$ or for $\mathcal{B}' = \mathcal{B}_m \cup \{\overline{\phi_{m+1}}\}$, we have that $\bigwedge \mathcal{A} \wedge \langle\alpha\rangle \bigwedge \mathcal{B}'$ is consistent. We take $\mathcal{B}_{m+1}$ as the consistent extension. At the end, we make $\mathcal{B} = \mathcal{B}_n$. We have that $\phi \in \mathcal{B}$ and, as $\bigwedge \mathcal{A} \wedge \langle\alpha\rangle \bigwedge \mathcal{B}$ is consistent, $\mathcal{A}S_\alpha\mathcal{B}$, by definition 22, which implies that $\mathcal{A}R_\alpha\mathcal{B}$.

($\Leftarrow$): Suppose that there is an atom $\mathcal{B}$ such that $\phi \in \mathcal{B}$ and $\mathcal{A}R_\alpha\mathcal{B}$. Then $\mathcal{A}S_\alpha\mathcal{B}$ and $\bigwedge \mathcal{A} \wedge \langle\alpha\rangle \bigwedge \mathcal{B}$ is consistent by definition 22. As ϕ is one of the conjuncts of $\bigwedge \mathcal{B}$, $\bigwedge \mathcal{A} \wedge \langle\alpha\rangle\phi$ is also consistent. As $\langle\alpha\rangle\phi$ is in $C(\Gamma)$, it must also be in $\mathcal{A}$, since $\mathcal{A}$ is a maximal consistent subset of $C(\Gamma)$. □

Lemma 4. *For all looping processes P, $S_P \subseteq S'_P$, where $S'_P = S^*_{L_P} \circ S_{T_P}$.*

Proof. For an atom $\mathcal{B} \in At(\Gamma)$ and a relation S, we denote the set of atoms $\{\mathcal{A} \mid \mathcal{A}S\mathcal{B}\}$ as $\langle S\rangle\mathcal{B}$. Suppose there are two atoms $\mathcal{A}, \mathcal{B} \in At(\Gamma)$ such that $\mathcal{A} \in \langle S_P\rangle\mathcal{B}$, but $\mathcal{A} \notin \langle S'_P\rangle\mathcal{B}$. Let $V = \{\mathcal{C} \in At(\Gamma) \mid \mathcal{C} \in \langle S_P\rangle\mathcal{B}$ but $\mathcal{C} \notin \langle S'_P\rangle\mathcal{B}\} \cup \{\mathcal{C} \in At(\Gamma) \mid \mathcal{C} \notin \langle S_P\rangle\mathcal{B}\}$ and $\overline{V} = At(\Gamma) \setminus V = \{\mathcal{C} \in At(\Gamma) \mid \mathcal{C} \in \langle S_P\rangle\mathcal{B}$ and $\mathcal{C} \in \langle S'_P\rangle\mathcal{B}\}$. Thus, $\mathcal{A} \in V$. Let $r = \bigvee\{\bigwedge \mathcal{C} \mid \mathcal{C} \in V\}$. It is not difficult to see that $\neg r = \bigvee\{\bigwedge \mathcal{C} \mid \mathcal{C} \in \overline{V}\}$.

First, we have that $\vdash r \rightarrow [T_P]\neg \bigwedge \mathcal{B}$. Otherwise, $\neg(r \rightarrow [T_P]\neg \bigwedge \mathcal{B}) \equiv r \wedge \langle T_P\rangle \bigwedge \mathcal{B}$ is consistent. This means that there is $\mathcal{A}' \in V$ such that $\bigwedge \mathcal{A}' \wedge \langle T_P\rangle \bigwedge \mathcal{B}$ is consistent. On one hand, this implies, by **(Rec)**, that $\bigwedge \mathcal{A}' \wedge \langle P\rangle \bigwedge \mathcal{B}$ is consistent, which means that $\mathcal{A}' \in \langle S_P\rangle\mathcal{B}$. On the other hand, it implies that $\mathcal{A}'S_{T_P}\mathcal{B}$, which means that $\mathcal{A}' \in \langle S'_P\rangle\mathcal{B}$. These two conclusions contradict the fact that $\mathcal{A}' \in V$.

Second, we also have that $\vdash r \rightarrow [L_P]r$. Otherwise, $\neg(r \rightarrow [L_P]r) \equiv r \wedge \langle L_P\rangle\neg r$ is consistent. This means that there are $\mathcal{A}' \in V$ and $\mathcal{B}' \in \overline{V}$ such that $\bigwedge \mathcal{A}' \wedge \langle L_P\rangle \bigwedge \mathcal{B}'$ is consistent, which implies that $\mathcal{A}'S_{L_P}\mathcal{B}'$. Since $\mathcal{B}' \in \overline{V}$, $\mathcal{B}'S_P\mathcal{B}$ and $\mathcal{B}'S'_P\mathcal{B}$. On one hand, $\mathcal{A}'S_{L_P}\mathcal{B}'$ and $\mathcal{B}'S'_P\mathcal{B}$ imply that $\mathcal{A}'S'_P\mathcal{B}$ (*). On the other hand, $\mathcal{A}'S_{L_P}\mathcal{B}'$ and $\mathcal{B}'S_P\mathcal{B}$ imply that $\bigwedge \mathcal{A}' \wedge \langle L_P\rangle\langle P\rangle \bigwedge \mathcal{B}$ is consistent, which, by **(Rec)**, implies that $\bigwedge \mathcal{A}' \wedge \langle P\rangle \bigwedge \mathcal{B}$ is consistent, which means that $\mathcal{A}'S_P\mathcal{B}$ (**). The conclusions in (*) and (**) contradict the fact that $\mathcal{A}' \in V$.

Taking these two results together, we conclude that $\vdash r \rightarrow ([T_P]\neg \bigwedge \mathcal{B} \wedge [L_P]r)$. By **(Gen)**, **(PL)**, **(FP)** and **(MP)**, $\vdash r \rightarrow [P]\neg \bigwedge \mathcal{B}$. But, as $\mathcal{A} \in V$, $\vdash \bigwedge \mathcal{A} \rightarrow r$, which means that $\vdash \bigwedge \mathcal{A} \rightarrow [P]\neg \bigwedge \mathcal{B}$. This implies that $\bigwedge \mathcal{A} \wedge \langle P\rangle \bigwedge \mathcal{B}$ is inconsistent, contradicting the fact that $\mathcal{A}S_P\mathcal{B}$. Thus, there cannot be a pair of atoms $\mathcal{A}, \mathcal{B} \in At(\Gamma)$ such that $\mathcal{A} \in \langle S_P\rangle\mathcal{B}$, but $\mathcal{A} \notin \langle S'_P\rangle\mathcal{B}$. □

Lemma 5. *For all processes P, $S_P \subseteq R_P$.*

Proof. The proof is by induction on the structure of the process P.

- The base case is immediate, for we defined $R_\alpha = S_\alpha$ for all basic processes α.
- P is a non-looping process:

- Suppose $\mathcal{A}S_{\alpha.P}\mathcal{B}$, that is, $\bigwedge\mathcal{A}\wedge\langle\alpha.P\rangle\bigwedge\mathcal{B}$ is consistent. By **(Pr)**, $\bigwedge\mathcal{A}\wedge\langle\alpha\rangle\langle P\rangle\bigwedge\mathcal{B}$ is consistent as well. Using a "forcing choices" argument (as exemplified in lemma 3), we can construct an atom $\mathcal{C}$ such that $\bigwedge\mathcal{A}\wedge\langle\alpha\rangle\bigwedge\mathcal{C}$ and $\bigwedge\mathcal{C}\wedge\langle P\rangle\bigwedge\mathcal{B}$ are both consistent. But then, by the inductive hypothesis, $\mathcal{A}R_\alpha\mathcal{C}$ and $\mathcal{C}R_P\mathcal{B}$. It follows that $\mathcal{A}R_{\alpha.P}\mathcal{B}$ as required.
- Suppose $\mathcal{A}S_{\alpha.A}\mathcal{B}$, that is, $\bigwedge\mathcal{A}\wedge\langle\alpha.A\rangle\bigwedge\mathcal{B}$ is consistent. By **(Cons)**, $\bigwedge\mathcal{A}\wedge\langle\alpha\rangle\langle P_A\rangle\bigwedge\mathcal{B}$ is consistent as well. Using a "forcing choices" argument, we can construct an atom $\mathcal{C}$ such that $\bigwedge\mathcal{A}\wedge\langle\alpha\rangle\bigwedge\mathcal{C}$ and $\bigwedge\mathcal{C}\wedge\langle P_A\rangle\bigwedge\mathcal{B}$ are both consistent. But then, by the inductive hypothesis, $\mathcal{A}R_\alpha\mathcal{C}$ and $\mathcal{C}R_{P_A}\mathcal{B}$. It follows that $\mathcal{A}R_{\alpha.A}\mathcal{B}$ as required.
- Suppose $\mathcal{A}S_{P_1+P_2}\mathcal{B}$, that is, $\bigwedge\mathcal{A}\wedge\langle P_1+P_2\rangle\bigwedge\mathcal{B}$ is consistent. By **(NC)**, $\bigwedge\mathcal{A}\wedge\langle P_1\rangle\bigwedge\mathcal{B}$ is consistent or $\bigwedge\mathcal{A}\wedge\langle P_2\rangle\bigwedge\mathcal{B}$ is consistent. But then, by the inductive hypothesis, $\mathcal{A}R_{P_1}\mathcal{B}$ or $\mathcal{A}R_{P_2}\mathcal{B}$. It follows that $\mathcal{A}R_{P_1+P_2}\mathcal{B}$ as required.
- Suppose $\mathcal{A}S_{P_1|P_2}\mathcal{B}$, that is, $\bigwedge\mathcal{A}\wedge\langle P_1\mid P_2\rangle\bigwedge\mathcal{B}$ is consistent. By **(PC)**, $\bigwedge\mathcal{A}\wedge\langle\alpha\rangle\langle P'\rangle\bigwedge\mathcal{B}$ is consistent for some basic process α and some process P'. Using a "forcing choices" argument, we can construct an atom $\mathcal{C}$ such that $\bigwedge\mathcal{A}\wedge\langle\alpha\rangle\bigwedge\mathcal{C}$ and $\bigwedge\mathcal{C}\wedge\langle P'\rangle\bigwedge\mathcal{B}$ are both consistent. But then, by the inductive hypothesis, $\mathcal{A}R_\alpha\mathcal{C}$ and $\mathcal{C}R_{P'}\mathcal{B}$. It follows that $\mathcal{A}R_{\alpha.P'}\mathcal{B}$, which means that $\mathcal{A}R_{P_1|P_2}\mathcal{B}$ as required.

– Suppose $\mathcal{A}S_P\mathcal{B}$, where P is a looping process. By lemma 4, $S_P\subseteq S'_P$, where $S'_P=S^*_{L_P}\circ S_{T_P}$. By the induction hypothesis, $S_{L_P}\subseteq R_{L_P}$ and $S_{T_P}\subseteq R_{T_P}$. This implies that $S'_P\subseteq R_P$, which proves the result. □

Lemma 6 (Existence Lemma). *For all atoms $\mathcal{A}\in At(\Gamma)$ and all formulas $\langle P\rangle\phi\in C(\Gamma)$, $\langle P\rangle\phi\in\mathcal{A}$ iff there is $\mathcal{B}\in At(\Gamma)$ such that $\mathcal{A}R_P\mathcal{B}$ and $\phi\in\mathcal{B}$.*

Proof. ($\Rightarrow$) Suppose $\langle P\rangle\phi\in\mathcal{A}$. We can build an atom $\mathcal{B}$ such that $\phi\in\mathcal{B}$ and $\mathcal{A}S_P\mathcal{B}$ by "forcing choices". But, by lemma 5, $S_P\subseteq R_P$, thus $\mathcal{A}R_P\mathcal{B}$ as well.

($\Leftarrow$) We proceed by induction on the structure of P.

– The base case is just the Existence Lemma for basic processes.
– P is a non-looping process:
  - Suppose P has the form $\alpha.P'$, $\mathcal{A}R_{\alpha.P'}\mathcal{B}$ and $\phi\in\mathcal{B}$. Thus, there is an atom $\mathcal{C}$ such that $\mathcal{A}R_\alpha\mathcal{C}$ and $\mathcal{C}R_{P'}\mathcal{B}$. By the Fischer-Ladner closure conditions, $\langle P'\rangle\phi\in C(\Gamma)$, hence by the induction hypothesis, $\langle P\rangle\phi\in C$. Similarly, as $\langle\alpha\rangle\langle P'\rangle\phi\in C(\Gamma)$, $\langle\alpha\rangle\langle P'\rangle\phi\in\mathcal{A}$. Hence, by **(Pr)**, $\langle\alpha.P\rangle\phi\in\mathcal{A}$.
  - Suppose P has the form $\alpha.A$, $\mathcal{A}R_{\alpha.A}\mathcal{B}$ and $\phi\in\mathcal{B}$. Thus, there is an atom $\mathcal{C}$ such that $\mathcal{A}R_\alpha\mathcal{C}$, $\mathcal{C}R_{P_A}\mathcal{B}$ and $\phi\in\mathcal{B}$. By the Fischer-Ladner closure conditions, $\langle P_A\rangle\phi\in C(\Gamma)$, hence by the induction hypothesis, $\langle P_A\rangle\phi\in C$. Similarly, as $\langle\alpha\rangle\langle P_A\rangle\phi\in C(\Gamma)$, $\langle\alpha\rangle\langle P_A\rangle\phi\in\mathcal{A}$. Hence, by **(Cons)**, $\langle\alpha.A\rangle\phi\in\mathcal{A}$.
  - Suppose P has the form P_1+P_2, $\mathcal{A}R_{P_1+P_2}\mathcal{B}$ and $\phi\in\mathcal{B}$. Thus, $\mathcal{A}R_{P_1}\mathcal{B}$ or $\mathcal{A}R_{P_2}\mathcal{B}$. By the Fischer-Ladner closure conditions, $\langle P_1\rangle\phi,\langle P_2\rangle\phi\in C(\Gamma)$, hence by the inductive hypothesis, $\langle P_1\rangle\phi\in\mathcal{A}$ or $\langle P_2\rangle\phi\in\mathcal{A}$. Hence, by **(NC)**, $\langle P_1+P_2\rangle\phi\in\mathcal{A}$.

  - Suppose P has the form $P_1 \mid P_2$, $\mathcal{A}R_{P_1|P_2}\mathcal{B}$ and $\phi \in \mathcal{B}$. Thus, $\mathcal{A}R_{\alpha.P'}\mathcal{B}$ for some process α and some process P'. Then, there is an atom $\mathcal{C}$ such that $\mathcal{A}R_\alpha\mathcal{C}$ and $\mathcal{C}R_{P'}\mathcal{B}$. By the Fischer-Ladner closure conditions, $\langle\alpha.P'\rangle\phi, \langle\alpha\rangle\langle P'\rangle\phi, \langle P'\rangle\phi \in C(\Gamma)$, hence by the inductive hypothesis, $\langle P\rangle\phi \in C$ and $\langle\alpha\rangle\langle P'\rangle\phi \in \mathcal{A}$. Hence, by **(Pr)**, $\langle\alpha.P\rangle\phi \in \mathcal{A}$ and, by **(PC)**, $\langle P_1 \mid P_2\rangle\phi \in \mathcal{A}$.
- Suppose P is a looping process, $\mathcal{A}R_P\mathcal{B}$ and $\phi \in \mathcal{B}$. Then, there is a finite sequence of atoms $\mathcal{C}_0 \ldots \mathcal{C}_n$ such that $\mathcal{A} = \mathcal{C}_0 R_{L_P} \mathcal{C}_1 \ldots \mathcal{C}_{n-1} R_{L_P} \mathcal{C}_n R_{T_P} \mathcal{B}$. We prove by a sub-induction on n that $\langle P\rangle\phi \in \mathcal{C}_i$, for all i. The desired result for $\mathcal{A} = \mathcal{C}_0$ follows immediately.
  - Base case: $n = 0$. This means $\mathcal{A}R_{T_P}\mathcal{B}$. By the Fischer-Ladner closure conditions, $\langle T_P\rangle\phi \in C(\Gamma)$, hence by the inductive hypothesis, $\langle T_P\rangle\phi \in \mathcal{A}$. Hence, by **(Rec)**, $\langle P\rangle\phi \in \mathcal{A}$.
  - Inductive step: Suppose the result holds for $k < n$, and that $\mathcal{A} = \mathcal{C}_0 R_{L_P} \mathcal{C}_1 \ldots R_{L_P} \mathcal{C}_n R_{T_P} \mathcal{B}$. By the inductive hypothesis, $\langle P\rangle\phi \in \mathcal{C}_1$. Hence $\langle L_P\rangle\langle P\rangle\phi \in \mathcal{A}$, as $\langle L_P\rangle\langle P\rangle\phi \in C(\Gamma)$. By **(Rec)**, $\langle P\rangle\phi \in \mathcal{A}$. □

Lemma 7 (Truth Lemma). *Let $\mathcal{N}^\Gamma = (At(\Gamma), \{R_P\}, \mathbf{V})$ be the CCS-PDL model over Γ. For all atoms $\mathcal{A} \in At(\Gamma)$ and all formulas $\varphi \in C(\Gamma)$, $\mathcal{N}^\Gamma, \mathcal{A} \Vdash \varphi$ iff $\varphi \in \mathcal{A}$.*

Proof. The proof is by induction on the structure of the formula φ.

- ϕ is a proposition symbol: The proof follows directly from the definition of $\mathbf{V}$.
- $\phi = \neg\psi$ or $\phi = \psi_1 \wedge \psi_2$: The proof follows directly from lemma 1.
- $\phi = \langle P\rangle\psi$:
  ($\Rightarrow$) Suppose that $\mathcal{N}^\Gamma, \mathcal{A} \Vdash \langle P\rangle\psi$. Then, there exists $\mathcal{A}' \in \mathcal{N}^\Gamma$ such that $\mathcal{A}R_P\mathcal{A}'$ and $\mathcal{N}^\Gamma, \mathcal{A}' \Vdash \psi$. By the induction hypothesis, we know that $\psi \in \mathcal{A}'$ and, by the Existence Lemma, we have that $\langle P\rangle\psi \in \mathcal{A}$.
  ($\Leftarrow$) Suppose that $\langle P\rangle\psi \in \mathcal{A}$. Then, by the Existence Lemma, there is $\mathcal{A}' \in \mathcal{N}^\Gamma$ such that $\mathcal{A}R_P\mathcal{A}'$ and $\psi \in \mathcal{A}'$. By the induction hypothesis, $\mathcal{N}^\Gamma, \mathcal{A}' \Vdash \psi$, which implies $\mathcal{N}^\Gamma, \mathcal{A} \Vdash \langle P\rangle\psi$. □

Theorem 6 (Completeness). *Every consistent formula is satisfiable in a finite model that respects definition 17.*

Proof. Let φ be a consistent formula. Let $C(\varphi)$ be its closure under the conditions of definition 19. As φ is consistent, by corollary 1, there is an atom $\mathcal{A} \in At(\varphi)$ such that $\varphi \in \mathcal{A}$. Let $\mathcal{N}^\varphi$ be the CCS-PDL model over φ. Then, by the Truth Lemma (lemma 7), as $\varphi \in \mathcal{A}$, we conclude that $\mathcal{N}^\varphi, \mathcal{A} \Vdash \varphi$, which proves the theorem. □

Towards Ontology Evolution in Physics*

Alan Bundy and Michael Chan

School of Informatics,
University of Edinburgh,
Edinburgh, UK
{bundy,mchan}@inf.ed.ac.uk

Abstract. We investigate the problem of automatically repairing inconsistent ontologies. A repair is triggered when a contradiction is detected between the current theory and new experimental evidence. We are working in the domain of physics because it has good historical records of such contradictions and how they were resolved. We use these records to both develop and evaluate our techniques. To deal with problems of inferential search control and ambiguity in the atomic repair operations, we have developed *ontology repair plans*, which represent common patterns of repair. They first diagnose the inconsistency and then direct the resulting repair. Two such plans have been developed to repair ontologies that disagree over the value and the dependence of a function, respectively. We have implemented the repair plans in the GALILEO system and successfully evaluated GALILEO on a diverse range of examples from the history of physics.

1 Introduction

Most ontologies are built manually for a particular reasoning task. Successful reasoning depends on striking a compromise between the expressiveness of the representation and the efficiency of the reasoning process. If either the reasoning environment or the goals subsequently change, then the reasoning process is likely to fail because the ontology is no longer well suited to its task.

Many modern applications of automated reasoning need to work in a changing environment with changing goals. Their reasoning systems need to adapt to these changes automatically. In particular, their ontologies need to evolve automatically [1]. It is not enough to remove from or add to the beliefs of the ontology. It is necessary to change its underlying formal language. Our group has pioneered work in this new area of research. Our techniques involve diagnosis of faults in an existing ontology and then repairing these faults. They have previously been implemented and evaluated in the Ontology Repair System (ORS) [2].

We are now applying and developing our techniques in the domain of physics [3]. This is an excellent domain because many of its most seminal advances can be seen as ontology evolution, i.e., changing the way that physicists view the world. These changes are often triggered by a contradiction between existing

* The research reported in this paper was supported by EPSRC grant EP/E005713/1.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 98–110, 2008.

theory and experimental observation. These contradictions, their diagnosis and the resulting repairs have usually been well documented by historians of science, providing us with a rich vein of case studies for the development and evaluation of our techniques.

We will use the word 'ontology' generically, to refer to any logical theory, i.e., not restricted to ontologies in description logic, KIF or other logics in which ontologies have been traditionally formalised. The physics domain requires higher-order logic: both at the object-level, to describe things like planetary orbits and calculus, and at the meta-level, to describe the ontological repair operations. An ontology consists of two parts: the *signature*, which declares the functions and their types, and the *theory*, which is the set of theorems, defined recursively via the axioms and the rules of inference.

Our research will progress by composing together a number of diagnosis and repair operations into what we call *repair plans*, analysing a wide ranging development set of case studies, developing more repair plans and evaluating their performance on a test set of case studies. We will use this work as a basis to develop a theory of ontology evolution that we hope will also be applicable outwith the physics domain. We have already experimented with two repair plans, which we call "Where's my stuff?" (WMS) and "Inconstancy". In this paper we summarise our progress so far: describing our two repair plans, their implementation and their application to some seminal events in the history of physics.

2 Related Work

ORS evolved first-order ontologies by first diagnosing their faults via the execution failures of multi-agent plans, then using this diagnosis to guide repairs to these ontologies. These repairs were not just belief revisions, but changes to the underlying signatures, e.g., adding or removing function arguments, and splitting or conflating functions. These signature changes include but go beyond mere definitional extensions, i.e., adding definitions of new functions in terms of old ones. Some of the changes automated by both ORS and our current system extend the range of concepts that can be expressed.

There is also related work on repairing inconsistencies in OWL ontologies, for instance [6]. It focuses on strategies for removing inconsistent axioms and for identifying syntactical modelling errors in OWL ontologies to assist users to rewrite faulty axioms. Our focus, in contrast, is on repairing deeper conceptual errors in the underlying physical theory, rather than fixing errors in the use of the OWL operators. We are focusing on signature changes rather than removing or repairing axioms. We are also applying our techniques to higher-order rather than description logic ontologies. Nevertheless, we will be investigating this and other related work to identify opportunities for synergy.

More generally, our research differs from previous work on **ontology matching** by being focused on repairing a single ontology dynamically, automatically and without access to third party ontologies. It also differs from previous work on **belief revision** by being focused on signature changes rather than theory

changes. However, again we will try to identify opportunities for synergy with these parallel strands of work.

3 Ontology Repair Plans

Adding arguments to and splitting functions are examples of *refinement*, in which ontologies are enriched. Such ontology refinement presents the following tough challenges:

1. Refinement operations are only partially defined. For instance, when an additional argument is added to a function it is not always clear what value each of its instances should take, or indeed whether any candidate values are available. When a function is split into two, it is not always clear to which of the new functions each occurrence of the old one should be mapped.
2. There are combinatorial explosions in both the object-level inference within the evolving ontology and the meta-level inference required to diagnose and repair that ontology.
3. To evaluate ORS we wanted to compare its evolutionary behaviour to that of manually executed ontology modifications. For this we needed a series of versions of such ontologies together with a justification of the modifications between successive versions. This proved very difficult to obtain: ontology developers do not usually keep publically accessible development histories[1].

The work outlined in this paper addresses these problems in the following way:

- Problems 1 and 2 are being addressed by developing *repair plans*. A repair plan is analogous to a *proof plan* [5], a generic proof outline that can be used to guide the search for a proof. Repair plans are generic combinations of diagnosis and repair operations that guide the ontology evolution process. By grouping these meta-level operations they trade off completeness against a dramatic reduction in search. Appendices A and B describe the two repair plans we have developed to date. In addition, we will develop a theory of ontology evolution by isolating and generalising the atomic ontology repair operations arising in our case studies. For instance, since repairs need to be minimal in order to avoid unmotivated and unnecessary repairs, then a suitable concept of *minimality* needs to be defined and our repairs shown to be minimal with respect to it. Figure 1 outlines one approach we are currently exploring.
- Problem 3 is being addressed by working in the domain of physics. Some of the most seminal advances in the development of physics have required profound ontology evolution. Moreover, the evolutionary process in physics is

[1] A rare exception can be found in the CVS records at `http://sigmakee.cvs.sourceforge.net/sigmakee/KBs/`. Although, even here the annotations typically record *what* was changed but not much about *why*. We are also working on a project to re-evaluate ORS on some of these CVS records.

To define a concept of minimal repair we are experimenting with extending *conservative extension* to changes of signature, namely:

$$\phi \in Sig(O) \implies (\nu(O) \vdash \nu(\phi) \iff O \vdash \phi)$$

where $\phi \in Sig(O)$ means that ϕ is a formula in the signature of ontology O and $\nu(\phi)$ is the repaired ϕ in repaired ontology $\nu(O)$. In the repair plan in Appendix A on p107 both $\nu(O_t)$ and $\nu(O_s)$ are conservative in this extended sense. Their combination, of course, is not, since the purpose of the repair is to prevent a contradiction being derived in the repaired combination.

Fig. 1. Minimal Ontology Repair via Conservative Extensions

very well documented. Detailed accounts are available of: the problems with the prior ontology, e.g., a contradiction between theory and experiment; the new ontology; and an account of the reasoning which led to it.

The WMS ontology repair plan, described in detail in Appendix A, aims at resolving contradictions arising when the expected value returned by a function does not match the observed value. The expected value can be viewed as being deducible from an existing theory, whereas the observed value is obtained from some sensory data, which can be collected from empirical experiments. To break the inconsistency, the conflicting function in the theory is either split into two parts, *visible* and *invisible*, or becomes the visible part of a *total* function. The intuition behind this repair is that the discrepancy arose because the function was not being applied to the same *stuff* in the theory and the sensory ontologies - some of the stuff was invisible in one of the ontologies.

The Inconstancy ontology repair plan, described in Appendix B, is triggered when there is a conflict between the expected independence and the observed dependence of a function on some variable, i.e., the returned value of a function unexpectedly varies. To effect the repair, the variable causing the unexpected variation is first identified and a new, consistent definition for the conflicting function is then created. The new definition relies on a function relating the old definition to the varying condition. In our current implementation, this function is computed using curve fitting techniques.

4 Implementation

Our repair plans have been implemented in the *Guided Analysis of Logical Inconsistencies Leads to Evolved Ontologies* (GALILEO) system; we chose λProlog [4] as our implementation language because it provides a polymorphic, higher-order logic. This is well suited to the representation of the higher-order functions that occur in both the object-level domain of physics and the meta-level formulae of the repair plans. It provides higher-order unification to support matching of the meta-level triggers of the repair plans to the object-level triggering formulae in the case studies. It facilitates the representation of polymorphic functions such as

$-$, $<$ and $=$. It also provides depth-first search to facilitate inference at both the object- and meta-levels. These features combines to support rapid prototyping of the repair plans and their application to the development case studies.

To illustrate the GALILEO code, the main clause of the WMS plan is provided in Appendix C.

5 Applications to the Development Case Studies

We have applied GALILEO to the emulation of a small but diverse set of physics case studies. For instance, WMS has been applied to the discovery of latent heat, an apparent paradox about the lost energy of a bouncing ball, the invention of dark matter and the speculation of the additional planet Vulcan to explain an anomaly in the precession of the perihelion of Mercury. Inconstancy has been applied to the extension of Boyle's Law to the Gas Law and the adaption of Newton's theory of gravity (MOND) to explain an anomaly in the rate of expansion of the Universe. §5.1 and §5.2 outline two of the applications of WMS and §5.3 outlines an application of Inconstancy.

5.1 The Application of WMS to the Discovery of Latent Heat

Joseph Black discovered the concept of latent heat around 1750. Wiser and Carey [7] discuss a period when heat and temperature were conflated, which presented a conceptual barrier that Black had to overcome before he could formulate the concept of latent heat. This conflation creates a paradox: as water is frozen it is predicted to lose heat, but its heat, as measured by temperature, remains constant. Black had to split the concept of heat into energy and temperature.

The paradox faced by Black can be formalised as follows:

$$O_t \vdash Heat(H_2O, Start(Freeze)) = Heat(H_2O, Start(Freeze)) \quad (1)$$

$$O_s \vdash Heat(H_2O, Start(Freeze)) = Heat(H_2O, End(Freeze)) \quad (2)$$

$$O_t \vdash Heat(H_2O, Start(Freeze)) \neq Heat(H_2O, End(Freeze)) \quad (3)$$

where H_2O is the water being frozen, $Freeze$ is the time interval during which the freezing takes place, $Start$ returns the first moment of this period and End the last. (1) comes from the reflexive law of equality, (2) comes from the observed constant temperature during freezing and (3) is deduced from the then current physical theory that heat decreases strictly monotonically when objects are cooled.

These formulae match the repair plan trigger (7) in Appendix A with the following substitution:

$$\{Heat/stuff,\ \langle H_2O, Start(Freeze)\rangle/\boldsymbol{s},\ Heat(H_2O, Start(Freeze))/v_1,\ Heat(H_2O, End(Freeze))/v_2\}$$

To effect the repair we will define $\sigma_{vis} = \{Temp/stuff\}$ and $\sigma_{invis} = \{LHF/stuff\}$, respectively, in anticipation of their intended meanings, where LHF can be read

as the latent heat of fusion. These choices instantiate (8) in Appendix A to:

$$\forall o{:}obj, t{:}mom.\ LHF(o,t) ::= Heat(o,t) - Temp(o,t)$$

which is not quite what is required, but is along the right lines. Some further indirect observations of *LHF* are required to witness its behaviour under different states of o so that it can be further repaired, e.g., the removal of its t argument. The $Temp$ part of the new definition needs to be further refined so that its contribution of energy depends both on temperature and mass. These further refinements will be the subject of future ontology repair plans.

In the repaired ontologies, since $Heat(H_2O, Start(Freeze))$ is greater than $Heat(H_2O, End(Freeze))$, the repaired triggering formulae are transformed to:

$$\nu(O_t) \vdash Heat(H_2O, Start(Freeze)) = Heat(H_2O, Start(Freeze))$$
$$\nu(O_s) \vdash Temp(H_2O, Start(Freeze)) = Temp(H_2O, End(Freeze))$$

which breaks the derivation of the detected contradiction, as required.

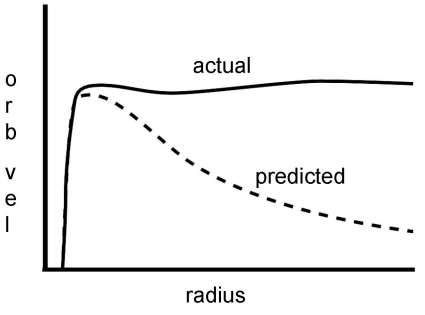
5.2 The Application of WMS to Dark Matter

The evidence for dark matter arises comes from various sources, for instance, from an anomaly in the orbital velocities of stars in spiral galaxies identified by Rubin [8]. Given the observed distribution of mass in these galaxies, we can use Newtonian Mechanics to predict that the orbital velocity of each star should be inversely proportional to the square root of its distance from the galactic centre (called its *radius*). However, observation of these stars show their orbital velocities to be roughly constant and independent of their radius. Figure 2 illustrates the predicted and actual graphs. In order to account for this discrepancy, it is hypothesised that galaxies also contain a halo of, so called, *dark matter*, which is invisible to our radiation detectors, such as telescopes, because it does not radiate, so can only be measured indirectly.

We can trigger the preconditions (7) in Appendix A with the following formulae:

$$O_t \vdash \lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_p \quad (4)$$
$$O_s \vdash \lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_a \quad (5)$$
$$O_t \vdash Graph_p \neq Graph_a \quad (6)$$

where $Orb\_Vel(s)$ is the orbital velocity of star s, $Rad(s)$ is the radius of s from the centre of its galaxy and *Spiral* is a particular spiral galaxy, represented as the set of stars it contains. Formula (4) shows the predicted graph, $Graph_p$: the orbital velocity decreases roughly inversely with the square root of the radius (see Figure 2). This graph is deduced by Newtonian Mechanics from the observed distribution of the visible stars in the spiral galaxy. Formula (5) shows the actual observed orbital velocity graph, $Graph_a$: it is almost a constant function over most of the values of s (see Figure 2). Note the use of λ abstraction in (4) and



This diagram is taken from `http://en.wikipedia.org/wiki/Galaxy_rotation_problem`. *The x-axis is the radii of the stars and the y-axis is their orbital velocities. The dotted line represents the predicted graph and the solid line is the actual graph that is observed.*

Fig. 2. Predicted *vs* Observed Stellar Orbital Velocities

(5) to create graph objects as unary functions. These two graphs are unequal (6), within the range of legitimate experimental variation.

These three formulae instantiate the trigger preconditions (7) with the following substitution:

$$\{\lambda s \in g.\ \langle Rad(s), Orb\_Vel(s)\rangle/stuff,\ \langle Spiral\rangle/\boldsymbol{s},\ Graph_p/v_1,\ Graph_a/v_2\}$$

Note that the repair plan works perfectly well with higher-order objects as the values v_1 and v_2, provided that polymorphic $-$ and $\neq$ can be defined as having meaning over this data-type: in this case a piecewise subtraction over the individual values for each star and a fuzzy, negated equality between graphs.

To effect the repair we will define $\sigma_{vis} = \{Spiral_{vis}/g\}$ and $\sigma_{invis} = \{Spiral_{invis}/g\}$, so the instantiation of definition (8) suggested by this triggering is:

$$\begin{aligned}&\lambda s \in Spiral_{invis}.\ \langle Rad(s), Orb\_Vel(s)\rangle\\ ::= &\lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle -\\ &\lambda s \in Spiral_{vis}.\ \langle Rad(s), Orb\_Vel(s)\rangle\end{aligned}$$

where $Spiral_{vis}$ is the visible part of the galaxy, that can be detected from its radiation, and $Spiral_{invis}$ is its dark matter part.

In the repaired ontologies, since $Graph_p < Graph_a$, the repaired triggering formulae are:

$$\begin{aligned}\nu(O_t) &\vdash \lambda s \in Spiral_{vis}.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_p\\ \nu(O_s) &\vdash \lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle = Graph_a\end{aligned}$$

which breaks the previous derivation of a contradiction, as required. Note that, unlike the latent heat case study, it is the repaired *theoretical* ontology that

formalises the visible stuff and it is the repaired *experimental* ontology that formalises the total stuff. This is because $Graph_p$ is based on the visible mass in the spiral and is, therefore, smaller than $Graph_a$.

5.3 The Application of Inconstancy to Modified Newtonian Mechanics

Another explanation of the anomaly in orbital velocities of stars in spiral galaxies is provided by MOdified Newtonian Dynamics (MOND), proposed by Moti Milgrom in 1981 as an alternative to the dark matter explanation. We have discussed in §5.2 that dark matter is an example of the WMS plan. MOND is an example of the Inconstancy plan. This is a good example of how the same observational discrepancies can trigger different repair plans. Essentially, MOND suggests that the gravitational constant is not a constant, but depends on the acceleration between the objects on which it is acting[2]. It is constant until the acceleration becomes very small and then it depends on this acceleration, which is the case for stars in spiral galaxies. So, the gravitational constant G can be repaired by giving it an additional argument to become $G(Acc(s))$, where $Acc(s)$ is the acceleration of a star s due to the gravitational attraction between the star and the galaxy in which it belongs. $Acc(s)$ is the variad and G is the inconstancy.

To satisfy the preconditions of the Inconstancy plan (10) and (10), we modify (4), (5), and (6) from §5.2. We want to have G instead of $\lambda s \in Spiral.\ \langle Rad(s), Orb\_Vel(s)\rangle$ on the left-hand-sides of (4) and (5). Similarly, we want to have the unrepaired representation of the gravitational constant on the right-hand-side of (4). We know from the law of universal gravitation that the square of the radius is inversely proportional to the acceleration of the orbiting star due to gravity, with the product of the gravitational constant and the mass of the galaxy being the constant of proportionality, i.e., $Rad(S_i)^2 = \frac{G \times M}{Acc(S_i)}$, where G, M, and $Acc(S_i)$ denote the gravitational constant, the mass of the galaxy, and the acceleration of the star w.r.t. the galaxy in which it belongs. So, we need to collect evidence for a variety of stars: S_i for $1 \leq i \leq n$, where $Acc(S_i)$ varies from large, i.e., S_i is near the centre of the galaxy, to small, i.e., S_i is on the periphery of the spiral galaxy. The graph shown in Figure 2, therefore, provides us with sufficient information to relate orbital velocities with accelerations of stars.

The trigger formulae for the Inconstancy plan will then be:

$$\begin{aligned} O_s(Acc(S_1) = A_1) \vdash G = M2OV^{-1}(OV(S_1), Mass(S_1),\\ \lambda s \in Spiral \setminus \{S_1\}.\ (Posn(s), Mass(s))) \ (= G_1)\\ \vdots \quad \vdots\\ O_s(Acc(S_n) = A_n) \vdash G = M2OV^{-1}(OV(S_n), Mass(S_n), \end{aligned}$$

[2] It can also be presented as breaking of the equivalence of inertial and gravitational mass, but the varying gravity story fits our purposes better.

$$\lambda s \in Spiral \setminus \{S_n\}.\ (Posn(s), Mass(s)))\ (= G_n)$$
$$O_t \vdash G ::= 6.67 \times 10^{-11}$$
$$\exists i \neq j.\ O_t \vdash G_i \neq G_j$$

where $M2OV^{-1}$ is the inverse of $M2OV$, which takes the value of G, the mass of a star s $Mass(s)$ and the mass distribution of all the remaining stars in the spiral galaxy based on their positions $Posn(s)$, and calculates the orbital velocity of s. $M2OV^{-1}$, therefore, takes the observed orbital velocity of a star s $OV(s)$, the mass of s and the mass distribution of all the remaining stars in the galaxy and calculates what value of G would account for the observed orbital velocity of s.

The formulae above triggers the Inconstancy plan with the following substitution:

$$\{G/stuff,\ \langle\rangle/\boldsymbol{s},\ \langle\rangle/\boldsymbol{x},\ 6.67 \times 10^{-11}/c,\ Acc/V,\ \langle S_i\rangle/\boldsymbol{b_i},\ G_1/c_1,\ G_n/c_n\}$$

Since G is a constant, both $\boldsymbol{s}$ and $\boldsymbol{x}$ are simply empty vectors.

Following the instructions for repair in Figure B, the variad is given to the inconstancy by:

$$\nu(G) ::= \lambda s.F(6.67 \times 10^{-11}, Acc(s))$$

and the repaired triggering formulae are therefore:

$$\begin{aligned}
\nu(O_s(Acc(S_1) = A_1)) \vdash\ & \nu(G)(S_1) = M2OV^{-1}(OV(S_1), Mass(S_1),\\
& \lambda s \in Spiral \setminus \{S_1\}.\ (Posn(s), Mass(s)))\\
& (= G_1)\\
\vdots\quad & \vdots\\
\nu(O_s(Acc(S_n) = A_n)) \vdash\ & \nu(G)(S_n) = M2OV^{-1}(OV(S_n), Mass(S_n),\\
& \lambda s \in Spiral \setminus \{S_n\}.\ (Posn(s), Mass(s)))\\
& (= G_n)\\
\nu(O_t) \vdash\ & \nu(G) ::= \lambda s.F(6.67 \times 10^{-11}, Acc(s))
\end{aligned}$$

which breaks the derivation of the detected contradiction, as required.

The function F can be determined by finding the best-fit curve for the whole dataset, in which each data point represents an observed G_i made under a particular condition $Acc(S_i) = A_i$. F is a reasonable approximation only if a fairly large number of observations of G_i for a wide range of accelerations of stars $Acc(S_i)$ are analysed. If F is a correct and complete approximation of $\nu(G)$, then $F(6.67 \times 10^{-11}, Acc(s))$ returns the unrepaired value 6.67×10^{-11} if a star s has an acceleration much greater than $1.2 \times 10^{-10} ms^{-2}$ (close to the centre of the galaxy). If s has an acceleration that is much less than $1.2 \times 10^{-10} ms^{-2}$ (near the periphery of the galaxy), the value returned will be greater than 6.67×10^{-11} and proportional to $Acc(s)^2 \times Rad(s)^2$, where $Rad(s)$ is the radius of the star's orbit.

6 Conclusion

Our proposed research programme is still in its early stages, although initial progress is promising. We have identified two of the five to ten general-purpose repair plans we seek, implemented them and applied them to the development set of case studies. Although small, this development set is satisfyingly diverse. Our current initial ontologies are *ad hoc*. We plan to develop more principled ones. Our prototype implementation requires the triggering formulae to be in just the right format, whereas later work will explore how to derive formulae meeting this format. This will raise difficult issues of search control. We need to evaluate our existing and future repair plans on a wider test set. We will require further investigation into the history of physics to identify both additional plans and both development and test case studies. Our theory of ontology evolution is in its infancy, although our extended notion of conservative extension is a promising start.

References

1. Bundy, A., McNeill, F.: Representation as a fluent: An AI challenge for the next half century. IEEE Intelligent Systems 21(3), 85–87 (2006)
2. McNeill, F., Bundy, A.: Dynamic, automatic, first-order ontology repair by diagnosis of failed plan execution. IJSWIS, Special issue on ontology matching 3(3), 1–35 (2007)
3. Bundy, A.: Where's my stuff? An ontology repair plan. In: Ahrendt, W., Baumgartner, P., de Nivelle, H. (eds.) Proceedings of the Workshop on Disproving - Non-Theorems, Non-Validity, Non-Provability, Bremen, Germany, July 2007, pp. 2–12 (2007), http://www.cs.chalmers.se/~ahrendt/CADE07-ws-disproving/
4. Miller, D., Nadathur, G.: An overview of λProlog. In: Bowen, R. (ed.) Proceedings of the Fifth International Logic Programming Conference/ Fifth Symposium on Logic Programming. MIT Press, Cambridge (1988)
5. Bundy, A.: A science of reasoning. In: Lassez, J.L., Plotkin, G. (eds.) Computational Logic: Essays in Honor of Alan Robinson, pp. 178–198. MIT Press, Cambridge (1991)
6. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.: Repairing Unsatisfiable Concepts in OWL Ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 170–184. Springer, Heidelberg (2006)
7. Wiser, M., Carey, S.: When heat and temperature were one. In: Stevens, A., Gentner, D. (eds.) Mental Models, pp. 267–297. Erlbaum, Mahwah (1983)
8. Rubin, V.C., Thonnard, N., Ford, W.K.J.: Rotational properties of 21 SC galaxies with a large range of luminosities and radii, from NGC 4605 /R = 4kpc/ to UGC 2885 /R = 122 kpc/. Astrophysical Journal 238, 471 (1980)
9. Langley, P., Zytkow, J., Simon, H., Bradshaw, G.: Mechanisms for qualitative and quantitative discovery. In: Michalski, R.S. (ed.) Proceedings of the International Machine Learning Workshop, University of Illinois, June 1983, pp. 121–132 (1983)

A The "Where's My Stuff?" Ontology Repair Plan

Suppose we have an ontology O_t representing the current state of a physical theory and an ontology O_s representing some sensory information arising from

an experiment. Suppose these two ontologies disagree over the value of some function *stuff*[3] when it is applied to a vector of arguments $\boldsymbol{s}$ of type $\boldsymbol{\tau}$. $\mathit{stuff}(\boldsymbol{s})$ might, for instance, be the heat content of a block of ice or the orbit of a planet.

Trigger: If $\mathit{stuff}(\boldsymbol{s})$ has two different values in O_t and O_s then the following formula will be triggered, identifying a potential contradiction between theory and experiment.

$$O_t \vdash \mathit{stuff}(\boldsymbol{s}) = v_1, O_s \vdash \mathit{stuff}(\boldsymbol{s}) = v_2, O_t \vdash v_1 \neq v_2 \tag{7}$$

where $O \vdash \phi$ means that formula ϕ is a theorem of ontology O. Below we deal with the case where $v_1 > v_2$. The other case is symmetric, with the roles of O_t and O_s reversed.

Split Stuff: The repair is to split *stuff* into three new functions: visible stuff, invisible stuff and total stuff, recycling the original name for total stuff. Then we create a definition of invisible stuff in terms of total and visible stuff.

$$\forall \boldsymbol{s}{:}\boldsymbol{\tau}.\ \mathit{stuff}\,\sigma_{invis}(\boldsymbol{s}) ::= \mathit{stuff}(\boldsymbol{s}) - \mathit{stuff}\,\sigma_{vis}(\boldsymbol{s}) \tag{8}$$

When *stuff* is a constant then σ_{vis}[4] just replaces it with new constant standing for the visible stuff; when *stuff* is compound the replacement is more complex, but still automatable. Similar remarks hold for σ_{invis}.

Create New Axioms: Let $\nu(O_t)$ and $\nu(O_s)$ be the repaired ontologies. We calculate the axioms of the new ontologies in terms of those of the old as follows:

$$\begin{aligned} Ax(\nu(O_t)) ::= {} & \{\forall \boldsymbol{s}{:}\boldsymbol{\tau}.\ \mathit{stuff}\,\sigma_{invis}(\boldsymbol{s}) ::= \mathit{stuff}(\boldsymbol{s}) - \mathit{stuff}\,\sigma_{vis}(\boldsymbol{s})\} \cup \\ & Ax(O_t) \\ Ax(\nu(O_s)) ::= {} & \{\phi\{\mathit{stuff}/\mathit{stuff}\,\sigma_{vis}\} \mid \phi \in Ax(O_s)\} \end{aligned}$$

i.e., the axioms of $\nu(O_t)$ are the same as for O_t except for the addition of the new definition; the axioms of $\nu(O_s)$ are the same as for O_s except for the renaming of the original stuff to the visible stuff.

Note that the contradiction has now disappeared but the theorems of the two ontologies are preserved up to renaming and the logical consequences arising from adding the new *stuff* definition (8). Note also, that $=$, $>$ and $-$ have to be polymorphic, i.e., apply to a variety of types.

Having hypothesised the existence of some hitherto invisible (i.e., not detectable by current instruments) stuff, then a natural next question is to try to develop an instrument that *can* detect it, even if indirectly.

[3] *stuff* is a polymorphic, higher-order variable ranging over functions in physics.

[4] σ_{vis} and σ_{invis} are *replacements*. These resemble higher-order substitutions, except that constants, as well as variables, may be replaced with terms.

B The Inconstancy Ontology Repair Plan

Suppose that different sensory ontologies give distinct values for function $\mathit{stuff}(\boldsymbol{s})$ in different circumstances. Suppose function $V(\boldsymbol{s}, \boldsymbol{b})$, where $\boldsymbol{b}$ contains variables distinguishing among these circumstances, returns distinct values in each of these circumstances, but is *not* one of the parameters in $\boldsymbol{s}$, i.e., $\mathit{stuff}(\boldsymbol{s})$ does not depend on $V(\boldsymbol{s}, \boldsymbol{b})$. We will call $\mathit{stuff}(\boldsymbol{s})$ the *inconstancy* and $V(\boldsymbol{s}, \boldsymbol{b})$ the *variad*. The Inconstancy repair plan establishes a relationship between the variad $V(\boldsymbol{b})$ and the inconstancy $\mathit{stuff}(\boldsymbol{s})$. The inconstancy might, for instance, be the gravitational constant G and the variad might be the acceleration of an orbiting star due to the gravity, which is suggested by MOdified Newtonian Dynamics (MOND).

Trigger: If $\mathit{stuff}(\boldsymbol{s})$ is measured to take different values in different circumstances, then the following trigger formulae will be matched.

$$\begin{aligned} O_s(V(\boldsymbol{s}, \boldsymbol{b_1}) = v_1) &\vdash \mathit{stuff}(\boldsymbol{s}) = c_1 \\ &\vdots \quad \vdots \qquad (9) \\ O_s(V(\boldsymbol{s}, \boldsymbol{b_n}) = v_n) &\vdash \mathit{stuff}(\boldsymbol{s}) = c_n, \\ O_t &\vdash \mathit{stuff}(\boldsymbol{x}) ::= c(\boldsymbol{x}), \exists i \neq j. O_t \vdash c_i \neq c_j \qquad (10) \end{aligned}$$

where $\boldsymbol{x}$ can be instantiated by $\boldsymbol{s}$, $O_s(V(\boldsymbol{s}, \boldsymbol{b_i}) = v_i)$ is the sensory ontology containing observations made under the condition that $V(\boldsymbol{s}, \boldsymbol{b_i}) = v_i$ and $V(\boldsymbol{s}, \boldsymbol{b})$ is not an existing argument of $\mathit{stuff}(\boldsymbol{s})$, i.e., $V(\boldsymbol{s}, \boldsymbol{b}) \notin \boldsymbol{s}$.

Add Variad: The repair is to change the signature of all the ontologies to relate the inconstancy, $\mathit{stuff}(\boldsymbol{x})$, to the variad, $V(\boldsymbol{x}, \boldsymbol{y})$:

$$\nu(\mathit{stuff}) ::= \lambda \boldsymbol{y}, \boldsymbol{x}.\ F(c(\boldsymbol{x}), V(\boldsymbol{x}, \boldsymbol{y})) \qquad (11)$$

where F is a new function, whose value we will seek to determine by curve fitting against the data from the sensory ontologies.

Create New Axioms: We calculate the axioms of the new ontologies in terms of those of the old as follows:

$$\begin{aligned} Ax(\nu(O_s(V(\boldsymbol{s}, \boldsymbol{b_i}) = v_i))) ::= \{&\phi\{\mathit{stuff}/\nu(\mathit{stuff})(\boldsymbol{b_i})\} \mid \\ &\phi \in Ax(O_s(V(\boldsymbol{s}, \boldsymbol{b_i}) = v_i))\} \\ Ax(\nu(O_t)) ::= \{&\phi\{\mathit{stuff}/\nu(\mathit{stuff})(\boldsymbol{y})\} \mid \\ &\phi \in Ax(O_t) \setminus \{\mathit{stuff}(\boldsymbol{x}) ::= c(\boldsymbol{x})\}\} \cup \\ &\{\nu(\mathit{stuff}) ::= \lambda \boldsymbol{y}, \boldsymbol{x}.\ F(c(\boldsymbol{x}), V(\boldsymbol{x}, \boldsymbol{y}))\} \end{aligned}$$

i.e., the axioms of $\nu(O_t)$ and the $\nu(O_s(V(\boldsymbol{s}, \boldsymbol{b_i}) = v_i))$ are the same as for O_t and $O_s(V(\boldsymbol{s}, \boldsymbol{b_i}) = v_i)$ except for the replacement of the old *stuff* with $\nu(\mathit{stuff})$ and the replacement of the definition of $\mathit{stuff}(\boldsymbol{x})$ by the definition of $\nu(\mathit{stuff}(\boldsymbol{x}))$ in $\nu(O_t)$.

To discover the meaning of the function F, we follow the traditional of Langley's BACON program [9] by using curve fitting. The $O_s(V(\boldsymbol{s}, \boldsymbol{b_i}) = v_i)$ ontologies provide a useful collection of equations: $F(c(\boldsymbol{s}), V(\boldsymbol{s}, \boldsymbol{b_i})) = c_i$ for $i = 1, \ldots, n$. Curve fitting techniques, e.g., regression analysis, are applied to these equations to approximate a definition of F. This hypothesis can then be tested by creating additional observations $O_s(V(\boldsymbol{s}, \boldsymbol{b_j}) = v_j)$, for new values of $V(\boldsymbol{s}, \boldsymbol{b_j})$, and confirming or refuting the hypothesis.

C Example Code

Here is the main clause[5] of the WMS plan.

```
repair O1 O2 NA1 NA2 :-
    % Repair triggered. Find stuff, args and parity
    wms_trigger O1 O2 S L P,
    % Pick replaced stuff from S or L
    choose S L Tot,
    % Calculate total, visible and invisible stuff
    newstuff S L Tot STot SVis SInvis,
    % Get original axioms
    axioms O1 A1,
    % of both ontologies
    axioms O2 A2,
    % Flip to find opposite parity
    flip P FP,
    % Change both sets of axioms
    change P  O1 A1 STot SVis SInvis NA1,
    change FP O2 A2 STot SVis SInvis NA2.
```

`O1` and `O2` are the input initial theoretical and experimental ontologies (but which is which depends on the parity `P`). `NA1` and `NA2` are their output repaired axioms. `wms_trigger` checks that the triggering formula (7) is matched and returns the instantiations of *stuff*, its list of arguments and a parity according to whether $v_1 > v_2$ or $v_1 < v_2$. `choose` picks a candidate `Tot` to be replaced in σ_{vis} and σ_{invis}, and `newstuff` uses these replacements to calculate the new total, visible and invisible *stuff*. The old axioms are then found by `axioms` and repaired into the new axioms by `change`.

[5] Confusingly, λProlog uses the convention that words representing variables start with upper-case letters and constants with lower-case, which is the inverse of the standard mathematical convention we have used for our mathematical formula.

Nominal Matching and Alpha-Equivalence*
(Extended Abstract)

Christophe Calvès and Maribel Fernández

King's College London, Department of Computer Science,
Strand, London WC2R 2LS, UK
Christophe.Calves@kcl.ac.uk, Maribel.Fernandez@kcl.ac.uk

Abstract. Nominal techniques were introduced to represent in a simple and natural way systems that involve binders. The syntax includes an abstraction operator and a primitive notion of name swapping. Nominal matching is matching modulo α-equality, and has applications in programming languages and theorem proving, amongst others. In this paper we describe efficient algorithms to check the validity of equations involving binders, and also to solve matching problems modulo α-equivalence, using the nominal approach.

Keywords: Binders, α-equivalence, matching, nominal terms.

1 Introduction

The notion of a binder is ubiquitous in computer science. Programs, logic formulas, and process calculi, are some examples of systems that involve binding. Program transformations and optimisations, for instance, are defined as operations on programs, and therefore work uniformly on α-equivalence classes. To formally define a transformation rule acting on programs, we need to be able to distinguish between free and bound variables, and between meta-variables of the transformation rule and variables of the object language. We also need to be able to test for α-equivalence, and we need a notion of matching that takes into account α-equivalence.

Nominal techniques were introduced to represent in a simple and natural way systems that include binders [7,10,11]. The nominal approach to the representation of systems with binders is characterised by the distinction, at the syntactical level, between *atoms* (or object-level variables), which can be abstracted (we use the notation $[a]t$, where a is an atom and t is a term), and *meta-variables* (or just variables), which behave like first-order variables but may be decorated with atom permutations. Permutations are generated using swappings (e.g. $(a\ b)\cdot t$ means swap a and b everywhere in t). For instance, $(a\ b)\cdot\lambda[a]a = \lambda[b]b$, and $(a\ b)\cdot\lambda[a]X = \lambda[b](a\ b)\cdot X$ (we will introduce the notation formally in the next section). As shown in this example, permutations suspend on variables. The idea is that when a substitution is applied to X in $(a\ b)\cdot X$, the permutation will be

* This work has been partially funded by an EPSRC grant (EP/D501016/1 "CANS").

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 111–122, 2008.

applied to the term that instantiates X. Permutations of atoms are one of the main ingredients in the definition of α-equivalence for nominal terms.

Nominal terms [12] can be seen as trees, built from function symbols, tuples and abstraction term-constructors; atoms and variables are leaves. We can define by induction a *freshness relation* $a\#t$ (read "the atom a is fresh for the term t") which roughly corresponds to the notion of a not occurring unabstracted in t. Using freshness and swappings we can inductively define a notion of α-equivalence of terms. Nominal unification is the problem of deciding whether two nominal terms can be made α-equivalent by instantiating their variables. It is a generalisation of the unification problem for first-order terms [1], and has the same applications in rewriting [5], logic programming [3], etc. Urban, Pitts and Gabbay [12] showed that nominal unification is decidable, and gave an algorithm to find the most general solution to a nominal unification problem, if one exists.

In this paper we study a simpler version of the problem —nominal matching— that has applications in functional programming, rewriting and meta programming amongst others. In a matching problem $s \approx_\alpha t$, t is *ground* (i.e., it has no variables), or, more generally, it has variables that cannot be instantiated.[1] When the term t is ground we say that the matching problem is *ground*. The left-hand side of a matching problem $s \approx_\alpha t$ is called a *pattern*, and may have variables occurring multiple times. When each variable occurs at most once in patterns we say that the matching problem is *linear*. We present an efficient algorithm that can be used to solve both linear and non-linear matching problems modulo α, as well as ground and non-ground problems. An algorithm to test α-equivalence of nominal terms (ground or non-ground) can be easily derived.

The complexity of the algorithms depends on the kind of problem to be solved; it is given in the table below:

Case	Alpha-equivalence	Matching
Ground	linear	linear
Non-ground and linear	log-linear	log-linear
Non-ground and non-linear	log-linear	quadratic

We have implemented the algorithms using OCAML [9], the implementation is available from: `www.dcs.kcl.ac.uk/staff/maribel/CANS`. We give sample benchmarks in the Appendix (Section 6), for more details see the website above.

In functional programming applications, matching problems are ground and in this case our algorithm is linear in time, as indicated in the first line of the table above. To our knowledge, this is the only available nominal matching algorithm with this complexity.

We are currently deploying the algorithms in a rewriting tool that can be used to specify equational theories including binders in the nominal style (see [6, 4]), and to evaluate functions working on data structures that include binding. In future, we hope to be able to extend the implementation techniques discussed

[1] These variables may have suspended permutations.

in this paper to solve nominal unification problems. The complexity of nominal unification is still an open problem.

2 Background

Let Σ be a denumerable set of **function symbols** $f, g, \ldots$; $\mathcal{X}$ be a denumerable set of **variables** $X, Y, \ldots$ (representing meta-level variables); and $\mathcal{A}$ be a denumerable set of **atoms** $a, b, \ldots$ (representing object-level variables). We assume that Σ, $\mathcal{X}$ and $\mathcal{A}$ are pairwise disjoint. A **swapping** is a pair of (not necessarily distinct) atoms, written $(a\ b)$. **Permutations** π are lists of swappings, generated by the grammar: $\pi ::= \mathsf{Id} \mid (a\ b) \circ \pi$. We call Id the **identity permutation** and write π^{-1} for the permutation obtained by reversing the list of swappings in π. We denote by $\pi \circ \pi'$ the permutation containing all the swappings in π followed by those in π'. A pair $\pi{\cdot}X$ of a permutation π and a variable X is called a **suspension**.

Nominal terms, or just **terms** for short, over $\Sigma, \mathcal{X}, \mathcal{A}$ are generated by the grammar: $s, t ::= a \mid \pi{\cdot}X \mid (s_1, \ldots, s_n) \mid [a]s \mid f\ t$.

A term is **ground** if it has no variables; $V(t)$ denotes the set of elements of $\mathcal{X}$ that occur in t. We refer the reader to [12, 5] for more details and examples of nominal terms.

We can apply permutations and substitutions on terms, denoted $\pi{\cdot}t$ and $t[X \mapsto s]$ respectively. Permutations act top-down and accumulate on variables whereas substitutions act on variables. More precisely, $\pi{\cdot}t$ is defined by induction: $\mathsf{Id}{\cdot}t = t$ and $((a\ b) \circ \pi){\cdot}t = (a\ b){\cdot}(\pi{\cdot}t)$, where

$$(a\ b){\cdot}a = b \qquad (a\ b){\cdot}b = a \qquad (a\ b){\cdot}c = c \text{ if } c \notin \{a, b\}$$
$$(a\ b){\cdot}(\pi{\cdot}X) = ((a\ b) \circ \pi){\cdot}X \qquad (a\ b){\cdot}(f\ t) = f(a\ b){\cdot}t$$
$$(a\ b){\cdot}[n]t = [(a\ b){\cdot}n](a\ b){\cdot}t \qquad (a\ b){\cdot}(t_1, \ldots, t_n) = ((a\ b){\cdot}t_1, \ldots, (a\ b){\cdot}t_n)$$

In the sequel we abbreviate $\mathsf{Id}{\cdot}X$ as X when there is no ambiguity.

A **substitution** is generated by the grammar: $\sigma ::= \mathsf{Id} \mid [X \mapsto s]\sigma$. We write substitutions postfix and write $\circ$ for composition of substitutions: $t(\sigma \circ \sigma') = (t\sigma)\sigma'$. We define the instantiation of a term t by a substitution σ by induction: $t\ \mathsf{Id} = t$ and $t[X \mapsto s]\sigma = (t[X \mapsto s])\sigma$ where

$$a[X \mapsto s] = a \qquad (t_1, \ldots, t_n)[X \mapsto s] = (t_1[X \mapsto s], \ldots, t_n[X \mapsto s])$$
$$([a]t)[X \mapsto s] = [a]t[X \mapsto s] \qquad (ft)[X \mapsto s] = f(t[X \mapsto s])$$
$$(\pi{\cdot}X)[X \mapsto s] = \pi{\cdot}s \qquad (\pi{\cdot}Y)[X \mapsto s] = \pi{\cdot}Y$$

Constraints have the form: $a\#t$ or $s \approx_\alpha t$, where $\#$ is the **freshness** relation between atoms and terms, and $\approx_\alpha$ denotes **alpha-equality**. A set Pr of constraints is called a **problem**. Intuitively, $a\#t$ means that if a occurs in t then it must do so under an abstractor $[a]$-. For example, $a\#b$, and $a\#[a]a$ but not $a\#a$. We sometimes write $a, b\#s$ instead of $a\#s, b\#s$, or write $A\#s$, where A is a set of atoms, to mean that all atoms in A are fresh for s.

The following set of simplification rules from [12], acting on problems, can be used to *check* the validity of α-equality constraints (below $ds(\pi, \pi')$ is an abbreviation for $\{n \mid \pi{\cdot}n \neq \pi'{\cdot}n\}$).

$$
\begin{array}{rl}
a\#b, Pr \Longrightarrow Pr & \\
a\#fs, Pr \Longrightarrow a\#s, Pr & \\
a\#(s_1, \ldots, s_n), Pr \Longrightarrow a\#s_1, \ldots, a\#s_n, Pr & \\
a\#[b]s, Pr \Longrightarrow a\#s, Pr & \\
a\#[a]s, Pr \Longrightarrow Pr & \\
a\#\pi\cdot X, Pr \Longrightarrow \pi^{-1}\cdot a\#X, Pr & \pi \neq \mathsf{Id}
\end{array}
$$

$$
\begin{array}{rl}
a \approx_\alpha a, Pr \Longrightarrow Pr & \\
(l_1, \ldots, l_n) \approx_\alpha (s_1, \ldots, s_n), Pr \Longrightarrow l_1 \approx_\alpha s_1, \ldots, l_n \approx_\alpha s_n, Pr & \\
fl \approx_\alpha fs, Pr \Longrightarrow l \approx_\alpha s, Pr & \\
[a]l \approx_\alpha [a]s, Pr \Longrightarrow l \approx_\alpha s, Pr & \\
[a]l \approx_\alpha [b]s, Pr \Longrightarrow l \approx_\alpha (a\ b)\cdot s, a\#s, Pr & \\
\pi\cdot X \approx_\alpha \pi'\cdot X, Pr \Longrightarrow ds(\pi, \pi')\#X, Pr &
\end{array}
$$

Given a problem Pr, we apply the rules until we get an irreducible problem, i.e. a **normal form**. If only a set Δ of constraints of the form $a\#X$ are left, then the original problem is **valid** in the context Δ (i.e., $\Delta \vdash Pr$), otherwise it is **not valid**. Thus, a problem such as $X \approx_\alpha a$ is not valid, since it is irreducible. However, X can be made equal to a by *instantiation* (i.e., applying a substitution); we say that this constraint can be **solved**. If we impose the restriction that in a constraint $s \approx_\alpha t$ the variables in t cannot be instantiated and the variables in left-hand sides are disjoint from the variables in right-hand sides, then we obtain a nominal **matching** problem. If we require s to be linear (i.e., each variable occurs at most once in s), we obtain a **linear** nominal matching problem.

A most general **solution** to a nominal matching problem Pr is a pair (Δ, σ) of a freshness context and a substitution, obtained from the simplification rules above, enriched with an **instantiating** rule labelled with substitutions:

$$\pi\cdot X \approx_\alpha u, Pr \overset{X \mapsto \pi^{-1}\cdot u}{\Longrightarrow} Pr[X \mapsto \pi^{-1}\cdot u]$$

Note that there is no need to do an occur-check because left-hand side variables are distinct from right-hand side variables in a matching problem.

3 The Algorithm

The transformation rules given in Section 2 create permutations. Polynomial implementations of the nominal unification algorithm [2] rely on the use of **lazy permutations**: permutations are only pushed down a term when this is needed to apply a transformation rule. We will use this idea, but, since lazy permutations may grow (they accumulate), in order to obtain an efficient algorithm we will devise a mechanism to compose the swappings eagerly. The key idea is to work with a single *current* permutation, represented by an **environment**.

Definition 1. *Let s and t be terms, π be a permutation and A be a finite set of atoms. An **environment** ξ is a pair (π, A). We denote by ξ_π the permutation (resp. ξ_A the set of atoms) of an environment. We write $s \approx_\alpha \xi\Diamond t$ to represent $s \approx_\alpha \xi_\pi\cdot t$, $\xi_A \# t$, and call $s \approx_\alpha \xi\Diamond t$ an **environment constraint**.*

Definition 2. *An **environment problem** Pr is either $\bot$ or has the form $s_1 \approx_\alpha \xi_1 \Diamond t_1, \ldots, s_n \approx_\alpha \xi_n \Diamond t_n$, where $s_i \approx_\alpha \xi_i \Diamond t_i$ $(1 \leq i \leq n)$ are environment constraints. We will sometimes abbreviate it as $(s_i \approx_\alpha \xi_i \Diamond t_i)_1^n$.*

*The problems defined in Section 2 will be called **standard** to distinguish them from environment problems (standard problems have no environments). The **standard form** of an environment problem is obtained by applying as many times as possible the rule: $s \approx_\alpha \xi \Diamond t \Longrightarrow s \approx_\alpha \xi_\pi \cdot t,\ \xi_A \# t$. We denote by $[\![Pr]\!]$ the standard form of an environment problem Pr.*

This rule is terminating because it consumes a $\Diamond$ each time, without creating any. There is no critical pair so the system is locally confluent and because it terminates it is confluent [8]. Therefore the standard form of an environment problem exists and is unique, justifying the notation $[\![Pr]\!]$.

The **solutions** of an environment problem are the solutions of its standard form (see Section 2). A problem $\bot$ has no solutions. Two environment problems are **equivalent** if their standard forms are equivalent, i.e., have the same solutions.

Standard problems are translated into environment problems in linear time: $s \approx_\alpha t$ is encoded as $s \approx_\alpha \xi \Diamond t$ where $\xi = (\mathsf{Id}, \emptyset)$ and $A \# t$ is encoded as $t \approx_\alpha \xi \Diamond t$ where $\xi = (\mathsf{Id}, A)$. In the sequel we restrict our attention to checking α-equivalence constraints and solving matching problems. In the latter case, in environment constraints $s \approx_\alpha \xi \Diamond t$, the term t will not be instantiated and variables in s and t are disjoint. If right-hand sides t are ground terms, we will say that the problem is *ground*, and *non-ground* otherwise.

3.1 Core Algorithm

The algorithms to check α-equivalence constraints and to solve matching problems will be built in a modular way. The core module is common to both algorithms; only the final phase will be specific to matching or α-equivalence. There are four phases in the core algorithm. We denote by $\overline{Pr}^{\,c}$ the result of applying the core algorithm on Pr.

Phase 1. The input is an environment problem $Pr = (s_i \approx_\alpha \xi_i \Diamond t_i)_i^n$, that we reduce using the following transformation rules, where a, b could be the same atom and in the last rule $\xi' = ((a\ \xi_\pi \cdot b) \circ \xi_\pi,\ (\xi_A \cup \{\xi_\pi^{-1} \cdot a\}) \setminus \{b\})$.

$$
\begin{array}{lll}
Pr, & a & \approx_\alpha \xi \Diamond t \Longrightarrow \begin{cases} Pr & \text{if } a = \xi_\pi \cdot t \text{ and } t \notin \xi_A \\ \bot & \text{otherwise} \end{cases} \\
Pr, & (s_1, \ldots, s_n) & \approx_\alpha \xi \Diamond t \Longrightarrow \begin{cases} Pr,\ (s_i \approx_\alpha \xi \Diamond u_i)_1^n & \text{if } t = (u_1, \ldots, u_n) \\ \bot & \text{otherwise} \end{cases} \\
Pr, & f\ s & \approx_\alpha \xi \Diamond t \Longrightarrow \begin{cases} Pr,\ s \approx_\alpha \xi \Diamond u & \text{if } t = f\ u \\ \bot & \text{otherwise} \end{cases} \\
Pr, & [a]s & \approx_\alpha \xi \Diamond t \Longrightarrow \begin{cases} Pr,\ s \approx_\alpha \xi' \Diamond u & \text{if } t = [b]u \\ \bot & \text{otherwise} \end{cases}
\end{array}
$$

The environment problems that are irreducible for the rules above will be called **phase 1 normal forms** or **ph1nf** for short.

Proposition 1 (Phase 1 Normal Forms). *The normal forms for phase 1 rules are either* $\bot$ *or* $(\pi_i{\cdot}X_i \approx_\alpha \xi_i \Diamond s_i)_1^n$ *where* s_i *are nominal terms.*

Phase 2. This phase takes as input an environment problem in ph1nf, and moves the permutations to the right-hand side. More precisely, given a problem in ph1nf, we apply the rule:

$$\pi{\cdot}X \approx_\alpha \xi \Diamond t \Longrightarrow X \approx_\alpha (\pi^{-1}{\cdot}\xi)\Diamond t \qquad (\pi \neq \mathsf{Id})$$

where $\pi^{-1}{\cdot}\xi = (\pi^{-1} \circ \xi_\pi,\ \xi_A)$. Note that π^{-1} applies only to ξ_π here, because $\pi{\cdot}X \approx_\alpha \xi \Diamond t$ represents $\pi{\cdot}X \approx_\alpha \xi_\pi{\cdot}t,\ \xi_A \# t$.

If the problem is irreducible (i.e., it is a normal form for the rule above), we say it is a **phase 2 normal form**, or **ph2nf** for short.

Proposition 2 (Phase 2 Normal Forms). *Given a ph1nf problem, it has a unique normal form for the rule above, and it is either* $\bot$ *or a problem of the form* $(X_i \approx_\alpha \xi_i \Diamond t_i)_1^n$*, where the terms* t_i *are standard nominal terms.*

Phase 3. In the phases 1 and 2 we deal with $\approx_\alpha$ constraints. Phase 3 takes a ph2nf and simplifies freshness constraints, by propagating environments over terms. Since the input is a problem in ph2nf, each constraint has the form $X \approx_\alpha \xi \Diamond t$. We reduce it with the following rewrite rules, which propagate ξ over t and deal with problems containing $\bot$ (denoted $Pr[\bot]$):

$$\begin{array}{rcl}
\xi \Diamond a & \Longrightarrow & \begin{cases} \xi_\pi{\cdot}a & a \notin \xi_A \\ \bot & a \in \xi_A \end{cases} \\
\xi \Diamond f\ t & \Longrightarrow & f\ (\xi \Diamond t) \\
\xi \Diamond (t_1, \ldots, t_j) & \Longrightarrow & (\xi \Diamond t_i)_1^j \\
\xi \Diamond\ [a]s & \Longrightarrow & [\xi_\pi{\cdot}a]((\xi \setminus \{a\}) \Diamond s) \\
\xi \Diamond (\pi{\cdot}X) & \Longrightarrow & (\xi \circ \pi) \Diamond X \\
Pr[\bot] & \Longrightarrow & \bot
\end{array}$$

where $\xi \setminus \{a\} = (\xi_\pi, \xi_A \setminus \{a\})$ and $\xi \circ \pi = ((\xi_\pi \circ \pi), \pi^{-1}(\xi_A))$.

These rules move environments inside terms, so formally we need to extend the definition of nominal term, to allow us to attach an environment at any position inside the term. We omit the definition of terms with suspended environments, and give just the grammar for the normal forms, which may have environments suspended only on variable leaves:

Definition 3. *The language of normal environment terms is defined by:*

$$T_\xi = a \mid f\ T_\xi \mid (T_\xi, \ldots, T_\xi) \mid\ [a]T_\xi \mid \xi \Diamond X$$

Proposition 3 (Phase 3 Normal Forms - ph3nf). *The normal forms for this phase are either* $\bot$ *or* $(X_i \approx_\alpha t_i)_1^n$ *where* $t_i \in T_\xi$.

Phase 4. This phase computes the standard form of a ph3nf:

$$X \approx_\alpha C[\xi \Diamond X'] \Longrightarrow X \approx_\alpha C[\xi_\pi \cdot X'] \ , \ \xi_A \# X'$$

Proposition 4 (Phase 4 Normal Forms - ph4nf). *If we normalise a ph3nf using the rule above, we obtain either* $\bot$ *or* $(X_i \approx_\alpha u_i)_{i \in I}, (A_j \# X_j)_{j \in J}$ *where* u_i *are nominal terms and* I, J *may be empty.*

The core algorithm terminates, and preserves the set of solutions. Since all the reduction rules, except the rule dealing with $\bot$, are local (i.e. only modify one constraint), the result of applying the core algorithm to a set of constraints is the union of the results obtained for each individual constraint (assuming $\bot, Pr \equiv \bot$) .

3.2 Checking the Validity of α-Equivalence Constraints

To check that a set Pr of α-equivalence constraints is valid, we first run the core algorithm on Pr and then reduce the result $\overline{Pr}^{\,c}$ by the following rule:

$$(\alpha) \quad Pr \ , \ X \approx_\alpha t \Longrightarrow \begin{cases} Pr \ , \ supp(\pi) \# X & \text{if } t = \pi \cdot X \\ \bot & \text{otherwise} \end{cases}$$

where $supp(\pi)$ is the **support** of π: $supp(\pi) = \{a \mid \pi \cdot a \neq a\}$.

Since this rule is terminating (each application consumes one $\approx_\alpha$-constraint) and there are no critical pairs, it is confluent [8], therefore normal forms exist and are unique. $\overline{Pr}^{\,\approx_\alpha}$ denotes the normal form of $\overline{Pr}^{\,c}$ by the rule above.

Proposition 5 (Normal Forms $\overline{Pr}^{\,\approx_\alpha}$). $\overline{Pr}^{\,\approx_\alpha}$ *is either* $\bot$ *or* $(A_i \# X_i)_1^n$.

Proposition 6 (Correctness). *If* $\overline{Pr}^{\,\approx_\alpha}$ *is* $\bot$ *then* Pr *is not valid. If* $\overline{Pr}^{\,\approx_\alpha}$ *is* $(A_i \# X_i)_1^n$ *then* $\overline{Pr}^{\,\approx_\alpha} \vdash Pr$.

If the left-hand sides of $\approx_\alpha$-constraints in Pr are ground, then $\overline{Pr}^{\,c} = \overline{Pr}^{\,\approx_\alpha}$; rule (α) is not necessary in this case.

3.3 Solving Matching Problems

To solve a matching problem Pr, we first run the core algorithm on Pr and then if the problem is non-linear we normalise the result $\overline{Pr}^{\,c}$ by the following rule:

$$(?\approx) \quad Pr, \ X \approx_\alpha s \ , \ X \approx_\alpha t \Longrightarrow \begin{cases} Pr \ , \ X \approx_\alpha s \ , \ \overline{s \approx_\alpha t}^{\,\approx_\alpha} & \text{if } \overline{s \approx_\alpha t}^{\,\approx_\alpha} \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

This rule is terminating: each reduction consumes at least one $\approx_\alpha$-constraint, and the algorithm computing $\overline{Pr}^{\,\approx_\alpha}$ is also terminating. $\overline{Pr}^{\,?\approx}$ denotes a normal form of $\overline{Pr}^{\,c}$ by the rule $(?\approx)$.

Proposition 7 (Normal Forms $\overline{Pr}^{\,?\approx}$). *If we normalise $\overline{Pr}^{\,c}$ using the rule above, we obtain either $\bot$ or $(X_i \approx_\alpha s_i)_1^n$, $(A_i \# X_i)_1^m$ where s_i are standard terms, all X_i in the equations $(X_i \approx_\alpha s_i)_1^n$ are different variables and $\forall i, j : X_i \notin V(s_j)$.*

A problem of the form $(X_i \approx_\alpha s_i)_1^n$ where all X_i are distinct variables and $X_i \notin \mathrm{V}(s_j)$ is the coding of an idempotent substitution σ. $(A_i \# X_i)_1^n$ is a freshness context Δ. Thus, the result of the algorithm is either $\bot$ or a pair (σ, Δ) of a substitution and a freshness context.

Proposition 8 (Correctness). *$\overline{Pr}^{\,?\approx}$ is a most general solution of the matching problem Pr.*

4 Implementation

Coding the problem. Terms are represented as trees. We code $\xi \Diamond t$ by attaching ξ to the root of t. Environment constraints are represented as pairs of trees, and freshness constraints as a pair of a set of atoms and a variable. Problems are represented as lists of constraints.

Avoiding environment creation in the core algorithm. Instead of running each phase in turn, we combine them to have a **local reduction** strategy: we fully reduce one constraint into ph4nf before reducing other constraints.

Each rule in the algorithm involves at most one environment, obtained either by copying or modifying another one. Instead of copying environments (in the case of tuples), we will share them. Updates in the *current* environment will, because of sharing, affect all the constraints where this environment is used. However, thanks to our local reduction strategy, none of these constraints will be reduced until the current constraint is in ph4nf (and then it will not use any environment). At this point, by reversing the environment to its original state, we can safely reduce the other constraints. Therefore, we keep track of the operations we made in the environment, fully reduce the current constraint, and then reverse the operations before reducing another constraint.

Permutations and sets. We code atoms as integers, and permutations (resp. sets) as *mutable arrays* or as *functional maps* of atoms (resp. booleans) indexed by atoms such that the value at the index a is the image of a by the permutation (resp. the boolean indicating whether a is in the set or not).

On one hand, mutable arrays have the advantage that they can be accessed and updated in constant time, but are destructive so we may need to copy them sometimes (an operation that is linear in time and space in their size). On the other hand, an access/update on functional maps is logarithmic in time, but since updates are not destructive we do not need to copy them.

We will use either mutable arrays or functional maps depending on the kind of problem to be solved:

Case	Alpha-equivalence	Matching
Ground	mutable arrays	mutable arrays
Non-ground and linear	functional maps	functional maps
Non-ground and non-linear	functional maps	mutable arrays

Note that when the problem is ground, phase 4 is not needed in the core algorithm, and therefore we never need to display permutations or freshness constraints. Since in this case we only need to access and update the environment, arrays are more efficient. With linear, non-ground problems, we need phase 4, and the cost is quadratic using arrays, but log-linear using functional maps. We will discuss the non-linear case in Section 5.

Since we often need the inverse and the support of a permutation, when we create a permutation we compute at the same time its inverse and its support and keep them in the same tuple. This can be done in linear time with arrays and in log-linear time with maps.

Implementing the algorithms. The implementation of the core algorithm is essentially a traversal of the data structure representing the input problem Pr_0, propagating the environment using the techniques above. The result is a list $\underline{\overline{Pr_0}}^{\,c}$ of constraints in ph4nf. The α-equivalence algorithm then takes each $\approx_\alpha$-constraint in the list $\underline{\overline{Pr_0}}^{\,c}$ and reduces it with (α). The matching algorithm applies the rule $(?\approx)$: it traverses the list to take for each variable X the constraint $X \approx_\alpha s$ with minimal s (we define the size of s below), and store $S[X] = s$ in an array S indexed by variables. Then the algorithm traverses the list again applying the rule *Rl-Check-Subst:*

$$Pr\ ,\ X \approx_\alpha t \Longrightarrow \begin{cases} Pr\ ,\ \overline{S[X] \approx_\alpha t}^{\,\approx_\alpha} & \text{if } \overline{S[X] \approx_\alpha t}^{\,\approx_\alpha} \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

and $\overline{S[X] \approx_\alpha t}^{\,\approx_\alpha}$ is computed using arrays.

5 Complexity

Atoms are coded as integers, as explained above. Let M_{A_0} be the maximum atom in A_0 (the set of atoms occurring in the input problem Pr_0). Let $|t|_{\mathrm{n}}$ be the number of nodes in the tree representing t. Finally, let $\mathrm{MV}(t)$ be the multiset of the occurrences of variables in t.

Core algorithm. Below we analyse the cost of the algorithm.

Definition 4. *The size of the problem $s \approx_\alpha \xi \Diamond t$, written $|s \approx_\alpha \xi \Diamond t|$, is defined as $|s \approx_\alpha \xi \Diamond t| = |s| + |\xi| + |t|$ where $|\xi| = 2 \times |\xi_\pi| + |\xi_A|$, $|\xi_\pi|$ (resp. $|\xi_A|$) is the size of the array/map representing it, and $|t|$ is defined by: $|a| = |X| = 1$, $|f\ t| = 1+|t|$, $|(t_1, \ldots, t_j)| = 1+|t_1|+\ldots|t_j|$, $|\ [a]s| = 1+|s|$ and $|\pi \cdot X| = 1+|\pi|$.*

Proposition 9. *The core algorithm is linear in the size of the initial problem Pr_0 in the ground case, using mutable arrays. In the non-ground case, it is log-linear using functional maps and $\vartheta(|s \approx_\alpha t| + |M_{A_0}| \times |t|_n)$ using mutable arrays.*

The idea of the proof is that the core algorithm is essentially a traversal of the data structure representing the problem. Phases 1 to 3 are trivially linear with arrays and log-linear with functional maps. Phase 4 is done in $\vartheta(|\mathrm{M}_{A_0}|)$ with functional maps, and $\vartheta(|\mathrm{M}_{A_0}| \times |t|_n)$ with arrays.

Alpha-equivalence. To check the validity of an $\approx_\alpha$-constraint, after running the core algorithm we have to normalise the problem using the rule (α), as described in Section 3.2. If the right-hand sides of $\approx_\alpha$-constraints are ground, the core algorithm is sufficient and it is linear. Otherwise, each application of the rule (α) requires to know the support of a permutation, which we do because supports are always created with permutations and maintained when they are updated. Thanks to the use of functional maps, the support is copied in constant time when the permutation is copied, therefore the algorithm is also log-linear in the size of the problem in the non-ground case.

Matching. The algorithm to solve matching constraints consists of the core algorithm, followed by a normalisation phase (using rule ${}_?\!\approx$, see Section 3.3) that deals with variables occurring multiple times in the pattern. In the case of linear matching this is not needed – the core algorithm is sufficient.

In Section 4 we discussed the implementation of the rule ${}_?\!\approx$ using an array S indexed by variables and the rule *Rl-Check-Subst*. The construction of S requires the traversal of subterms of the term s and every term in the output of the core algorithm. This is done in time proportional to the size of the output of the core algorithm. At worst, the size is $|\mathrm{M}_{A_0}| \times \mathrm{MV}(t) + |s \approx_\alpha t|$ because phase 4 can add a suspended permutation and freshness constraints on every variable occurring in t. Therefore the output can be quadratic in the size of the input.

Then *Rl-Check-Subst* will compute $\overline{S[X_i] \approx_\alpha t_i}^{\approx_\alpha}$ for each constraint $X_i \approx_\alpha t_i$ in the result of the core algorithm. Phase 1 to 3 are linear in its size and phase 4 has a complexity $\vartheta(|\mathrm{M}_{A_0}| \times \mathrm{MV}(t_i))$, hence at worst quadratic in time in the size of the input problem. The worst case complexity arises when phase 4 suspends permutations on all variables. On the other hand, if the input problem already has in each variable a permutation of size $|\mathrm{M}_{A_0}|$ (i.e. variables are 'saturated' with permutations), then, since permutations cannot grow, the α-equivalence and matching algorithms are linear even using arrays. Note that the representation of a matching problem or an α-equivalence problem using higher-order abstract syntax does saturate the variables (they have to be applied to the set of atoms they can capture). The table below summarises the results:

Case	Alpha-equivalence	Matching
Ground	linear	linear
Non-ground and linear	log-linear	log-linear
Non-ground and non-linear	log-linear	quadratic

6 Benchmarks

The algorithms described above to check α-equivalence and to solve ground matching problems have been implemented in OCAML [9], using arrays. In Figure 1, we show benchmarks generated by building random solvable ground problems (i.e., problems that do not lead to $\bot$) and measuring the time taken by the α-equivalence and matching algorithms to give the result (marked as $\diamond$ and $+$ in the graph)[2].

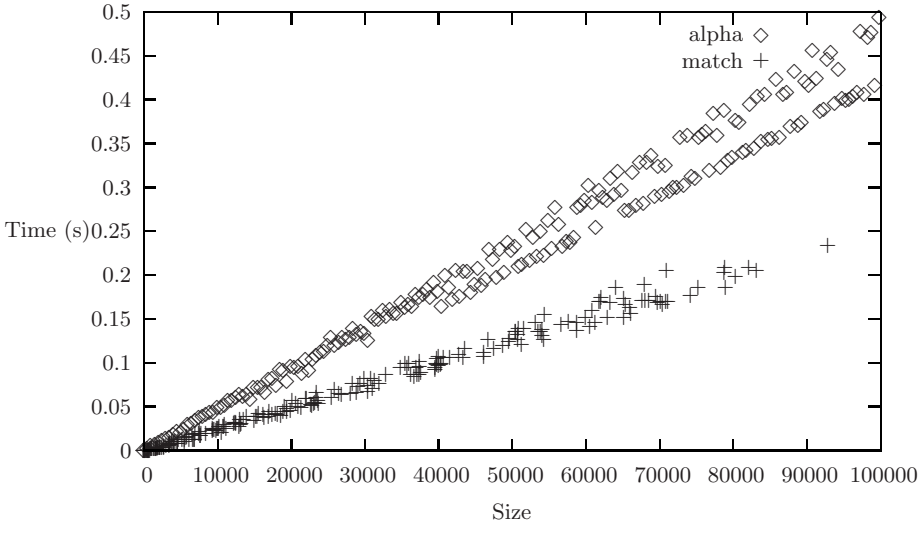


Fig. 1. Benchmarks

7 Conclusions

We described an algorithm to solve matching problems modulo α-equivalence which is linear time and space when the right-hand side terms are ground. Matching modulo α-equivalence has numerous applications, in particular, in the design of functional programming languages that provide constructs for declaring and manipulating abstract syntax trees involving names and binders, and as a basis for the implementation of nominal rewriting tools.

Acknowledgements. We thank James Cheney, Jamie Gabbay, Andrew Pitts, François Pottier and Christian Urban for useful discussions on the topics of this paper.

[2] The program is available from: `www.dcs.kcl.ac.uk/staff/maribel/CANS`

References

1. Baader, F., Snyder, W.: Unification Theory. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, ch. 8, vol. I, pp. 445–532. Elsevier Science, Amsterdam (2001)
2. Calvès, C., Fernández, M.: Implementing nominal unification. In: Proceedings of TERMGRAPH 2006, 3rd International Workshop on Term Graph Rewriting, ETAPS 2006, Vienna. Electronic Notes in Computer Science, Elsevier, Amsterdam (2006)
3. Cheney, J.: α-Prolog: an interpreter for a logic programming language based on nominal logic, `http://homepages.inf.ed.ac.uk/jcheney/programs/aprolog/`
4. Clouston, R.A., Pitts, A.M.: Nominal Equational Logic. In: Cardelli, L., Fiore, M., Winskel, G. (eds.) Computation, Meaning and Logic. Articles dedicated to Gordon Plotkin. Electronic Notes in Theoretical Computer Science, vol. 1496, Elsevier, Amsterdam (2007)
5. Fernández, M., Gabbay, M.J.: Nominal Rewriting. Information and Computation 205, 917–965 (2007)
6. Gabbay, M.J., Mathijssen, A.: Nominal Algebra. In: Proceedings of the 18th Nordic Workshop on Programming Theory (NWPT 2006) (2006)
7. Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. Formal Aspects of Computing 13, 341–363 (2001)
8. Newman, M.H.A.: On theories with a combinatorial definition of equivalence. Annals of Mathematics 43(2), 223–243 (1942)
9. OCAML, `http://caml.inria.fr/`
10. Pitts, A.M.: Nominal logic, a first order theory of names and binding. Information and Computation 186, 165–193 (2003)
11. Shinwell, M.R., Pitts, A.M., Gabbay, M.J.: FreshML: Programming with binders made simple. In: Eighth ACM SIGPLAN International Conference on Functional Programming (ICFP 2003), Sweden, pp. 263–274. ACM Press, New York (2003)
12. Urban, C., Pitts, A.M., Gabbay, M.J.: Nominal unification. Theoretical Computer Science 323, 473–497 (2004)

Interval Additive Generators of Interval T-Norms

G.P. Dimuro[1], B.C. Bedregal[2], R.H.S. Reiser[1], and R.H.N. Santiago[2]

[1] Programa de Pós-Graduação em Informática, Universidade Católica de Pelotas
Rua Felix da Cunha 412, 96010-000 Pelotas, Brazil
[2] Depto de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte
Campus Universitário s/n, 59072-970 Natal, Brazil
{liz,reiser}@ucpel.tche.br,{bedregal,regivan}@dimap.ufrn.br

Abstract. The aim of this paper is to introduce the notion of interval additive generators of interval t-norms as interval representations of additive generators of t-norms, considering both the correctness and the optimality criteria, in order to provide a more systematic methodology for the selection of interval t-norms in the various applications. We prove that interval additive generators satisfy the main properties of punctual additive generators discussed in the literature.

1 Introduction

Fuzzy set theory, which was introduced by Zadeh [1], is the oldest and most widely reported component of present-day soft computing, which deals with the design of flexible information processing systems [2], with applications in control systems [3], decision making [4], expert systems [5], pattern recognition [2, 6], etc.

Triangular norms (t-norms) on $([0,1],\leq)$ introduced by Schweizer [7] play an important role in fuzzy set theory (see, e.g., the works in [8, 9, 10, 11, 12, 13, 14, 15], where the importance of t-norms was extensively discussed). The use of t-norms and t-conorms can be considered vital for more flexible and reliable fuzzy logic controllers [16]. From t-norms, it is possible to derive several fuzzy implication functions (e.g, S-implications, R-implications, QL-implications, D-implications), which are important not only because they are used in representing "If ... then" rules in fuzzy systems, but also because their use in performing inferences in approximate reasoning and fuzzy control [17].

Generators on $([0,1],\leq)$ are very useful for the construction of t-norms, as transformations on the additive semigroup $([0,+\infty],+)$ and the multiplicative semigroup $([0,1],\cdot)$, and play an important role in the representation of continuous Archimedean t-norms (see, e.g., the results presented in [12, 18, 19, 11, 16, 20, 21, 22, 23]). Moreover, some properties of a t-norm that have a generator can be related to properties of its generator. The study of t-norms (and t-conorms) in terms of their additive generators can lead to a fresh view of t-operators and a more systematic methodology for their selection for the various applications, as properly pointed out by Leventides and Bounas [16].

On the other hand, Interval Mathematics [24, 25] is a mathematical theory that aims at the representation of uncertain input data and parameters, and the automatic and rigorous controlling of the errors that arise in numerical computations.

The integration of Fuzzy Theory and Interval Mathematics leads to the interval-valued fuzzy set theory, which has been studied from different viewpoints (see, e.g., [26, 27, 28,

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 123–135, 2008.

29, 30, 31, 32, 33, 34, 35, 36]). Lodwick [37] points out four ways to integrate fuzzy and interval approaches. One of them uses membership functions with intervals values, in order to model the uncertainty in the process of determining exact membership degrees with the usual fuzzy membership functions, i.e., interval memberships degrees are used to represent the uncertainty and the difficulty of an expert in determining which is the fairest membership degree for an element with respect to a linguistic term. Then, the radius of the interval indicates a maximal bound for the error [33, 38, 39], providing an estimation of that uncertainty. Interval degrees also can summarize the opinion of several experts with respect to the exact membership degree for an element with respect to a linguistic term. In this case, the left and right interval endpoints are, respectively, the least and the greatest degrees provided by a group of experts [28, 33, 40, 41].

In previous work [42,43,44,45,46], we investigated interval t-norms, t-conorms, and several interval fuzzy implications, adopting the approach introduced in [47,48], where interval extensions of some fuzzy connectives were constructed as their interval representations [49], which considers both correctness (accuracy) and optimality aspects as required in [50]. In this paper, we go towards the study of interval t-norms in terms of their interval additive generators, investigating their most important related properties and providing interval t-norms a more systematic view of their applicability. The paper is organized as follows. Section 2 presents the main concepts of interval representations of real functions. Additive generators and t-norms are discussed in Sect. 3. The definition and related properties of an interval t-norm are presented in Sect. 4. Section 5 introduces the interval additive generators of interval t-norms and the main related results. Section 6 is the Conclusion, with some final remarks on related work.

2 Interval Representations

Consider $a, b \in [-\infty, +\infty]$, with $a \leq b$, and the family $\mathbb{I}_{[a,b]} = \{[x, y] \mid a \leq x \leq y \leq b\}$. The concepts and results in the following were adapted from [49] by considering an interval $[a, b] \subseteq [-\infty, +\infty]$ and the family $\mathbb{I}_{[a,b]}$ in the place of $\mathbb{R}$ and $\mathbb{IR}$, respectively.

The set of intervals $\mathbb{I}_{[a,b]}$ has two projections $\pi_1, \pi_2 : \mathbb{I}_{[a,b]} \to [a, b]$, defined by $\pi_1([x_1, x_2]) = x_1$ and $\pi_2([x_1, x_2]) = x_2$, respectively, for any $[x_1, x_2] \in \mathbb{I}_{[a,b]}$. For each $X \in \mathbb{I}_{[a,b]}$, the projections $\pi_1(X)$ and $\pi_2(X)$ are also denoted by $\underline{X}$ and $\overline{X}$, respectively. Analogously, each interval function $F : \mathbb{I}^n_{[a,b]} \to \mathbb{I}_{[c,d]}$ has two projections, denoted by $\underline{F}, \overline{F} : [a, b]^n \to [c, d]$, defined, respectively, by:

$$\underline{F}(x_1, \ldots, x_n) = \pi_1(F([x_1, x_1], \ldots, [x_n, x_n])) \tag{1}$$

$$\overline{F}(x_1, \ldots, x_n) = \pi_2(F([x_1, x_1], \ldots, [x_n, x_n])) \tag{2}$$

The addition of two intervals $X = [x_1, x_2], Y = [y_1, y_2] \in \mathbb{I}_{[a,+\infty]}$, with $a \geq 0$, is defined by $X + Y = [x_1 + y_1, x_2 + y_2]$.

Several natural partial orders may be defined on $\mathbb{I}_{[a,b]}$ [51]. The most used orders in the context of interval mathematics, and considered in this work, are the following:

1. *Product order*: $X \leq Y$ if and only if $\underline{X} \leq \underline{Y}$ and $\overline{X} \leq \overline{Y}$.
2. *Inclusion order*: $X \subseteq Y$ if and only if $\underline{X} \geq \underline{Y}$ and $\overline{X} \leq \overline{Y}$.

In the theory of continuous domains, associated to both partial orders, there is also the *way below* relation [52, 53], which, for the case of the product order, is defined by:

$$[x_1, x_2] \ll [y_1, y_2] \text{ if and only if } x_1 < y_1 \text{ and } x_2 < y_2.$$

Remark 1. If $X \ll Y \leq Z$ then $x \ll Z$. If $X \ll Y$ then $X < Y$ (that is, $X \leq Y$ but $X \neq Y$). However, the converse may not hold (e.g, $[1,3] < [2,3]$, but $[1,3] \not\ll [2,3]$).

Consider $F : \mathbb{I}_{[a,b]} \to \mathbb{I}_{[c,d]}$. F is $\ll$-increasing if, for each $X, Y \in \mathbb{I}_{[a,b]}$ such that $X \ll Y$, it holds that $F(X) \ll F(Y)$. Analogously, F is $\ll$-decreasing if, for each $X, Y \in \mathbb{I}_{[a,b]}$ such that $X \ll Y$, it holds that $F(X) \gg F(Y)$, that is, $F(Y) \ll F(X)$.

Let X and Y be interval representations of a real number α, that is, $\alpha \in X$ and $\alpha \in Y$. X is said to be a better representation of α than Y whenever $X \subseteq Y$.

Definition 1. *A function $F : \mathbb{I}^n_{[a,b]} \longrightarrow \mathbb{I}_{[a,b]}$ is an* interval representation *of a function $f : [a,b]^n \longrightarrow [a,b]$ if, for each $\boldsymbol{X} \in \mathbb{I}^n_{[a,b]}$ and $\boldsymbol{x} \in \boldsymbol{X}$, $f(\boldsymbol{x}) \in F(\boldsymbol{X})$.* [1]

An interval function $F : \mathbb{I}^n_{[a,b]} \longrightarrow \mathbb{I}_{[c,d]}$ is a *better interval representation* of $f : [a,b]^n \longrightarrow [c,d]$ than $G : \mathbb{I}^n_{[a,b]} \longrightarrow \mathbb{I}_{[c,d]}$, denoted by $G \sqsubseteq F$, if, for each $\boldsymbol{X} \in \mathbb{I}^n_{[a,b]}$, $F(\boldsymbol{X}) \subseteq G(\boldsymbol{X})$.

Definition 2. *The* best interval representation *of a real function $f : [a,b]^n \longrightarrow [a,b]$, is the interval function $\widehat{f} : \mathbb{I}^n_{[a,b]} \longrightarrow \mathbb{I}_{[a,b]}$ defined by*

$$\widehat{f}(\boldsymbol{X}) = [\inf\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{X}\}, \sup\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{X}\}]. \tag{3}$$

The interval function $\widehat{f}$ is well defined and for any other interval representation F of f, $F \sqsubseteq \widehat{f}$. $\widehat{f}$ returns a narrower interval than any other interval representation of f. Thus, $\widehat{f}$ has the *optimality property* of interval algorithms mentioned by Hickey et al. [50], when it is seen as an algorithm to compute a real function f.

Remark 2. The best interval representation of a function $f : [a,b]^n \longrightarrow [a,b]$ coincides with the range of f in X, that is, for each $X \in \mathbb{I}^n_{[a,b]}$, $\widehat{f}(\boldsymbol{X}) = \{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{X}\} = f(\boldsymbol{X})$, if and only if f is continuous in the usual sense [49].

The range of a function $f : A \longrightarrow B$ is denoted by $Ran(f)$.

Proposition 1. *Consider $F : \mathbb{I}_{[a,b]} \to \mathbb{I}_{[c,d]}$. If F is $\ll$-increasing (decreasing) then $\underline{F}$ and $\overline{F}$ are strictly increasing (decreasing).*

Proof. It is straightforward.

Let $f, g : [a,b]^n \longrightarrow [a,b]$ be functions such that $f \leq g$ (i.e., for each $\boldsymbol{x} \in [a,b]^n$, $f(\boldsymbol{x}) \leq g(\boldsymbol{x})$). If f and g are both strictly increasing then define $\mathbb{I}_{[f,g]} : \mathbb{I}^n_{[a,b]} \longrightarrow \mathbb{I}_{[a,b]}$ by $\mathbb{I}_{[f,g]}(\boldsymbol{X}) = [f(\underline{\boldsymbol{X}}), g(\overline{\boldsymbol{X}})]$. If f and g are both strictly decreasing then define $\mathbb{I}_{[f,g]} : \mathbb{I}^n_{[a,b]} \longrightarrow \mathbb{I}_{[a,b]}$ by $\mathbb{I}_{[f,g]}(\boldsymbol{X}) = [f(\overline{\boldsymbol{X}}), g(\underline{\boldsymbol{X}})]$. Clearly, in both cases, it holds that $\mathbb{I}_{[f,f]} = \widehat{f}$. Moreover, for each function $h : [a,b]^n \longrightarrow [a,b]$ such that $f \leq h \leq g$, it holds that $\mathbb{I}_{[f,g]}$ (in both cases) is an interval representation of h.

[1] Other authors, e.g., [54, 55, 56], also consider this definition but with other name and purpose.

Lemma 1. *Consider $F : \mathbb{I}_{[a,b]} \to \mathbb{I}_{[c,d]}$. If F is $\ll$-increasing (decreasing) and inclusion monotonic then F is increasing (decreasing) w.r.t. the product order.*

Proof. Consider $X, Y \in \mathbb{I}_{[a,b]}$. If $X \leq Y$, then we have four possibilities: (1) $X = Y$, (2) $X \ll Y$, (3) $\underline{X} = \underline{Y}$ and $\overline{X} < \overline{Y}$, or (4) $\underline{X} < \underline{Y}$ and $\overline{X} = \overline{Y}$. The two first cases are trivial, whereas the third and fourth are analogous. So, we only prove the third case. Notice that, in this case, $X \subseteq Y$ and, therefore, $F(X) \subseteq F(Y)$. Thus, it follows that $\pi_1(F(X)) \leq \pi_1(F(Y))$. On the other hand, $X \ll [\overline{Y}, \overline{Y}] \subseteq Y$, and then it holds that $F(X) \ll F([\overline{Y}, \overline{Y}]) \subseteq F(Y)$. Therefore, we have that $\pi_2(F(X)) < \pi_2(F([\overline{Y}, \overline{Y}]) \leq \pi_2(F(Y))$ and, hence, we conclude that $F(X) \leq F(Y)$.

Proposition 2. *Consider $F : \mathbb{I}_{[a,b]} \to \mathbb{I}_{[c,d]}$. If F is $\ll$-increasing (decreasing) and inclusion monotonic then $F = \mathbb{I}_{[\underline{F}, \overline{F}]}$.*

Proof. Consider $X \in \mathbb{I}_{[a,b]}$. Since $[\underline{X}, \underline{X}] \leq X$, $[\underline{X}, \underline{X}] \subseteq X$ and F is increasing and inclusion monotonic, then it holds that $F([\underline{X}, \underline{X}]) \leq F(X)$ and $F([\underline{X}, \underline{X}]) \subseteq F(X)$. Thus, we have that $\pi_1(F([\underline{X}, \underline{X}])) \leq \pi_1(F(X))$ and $\pi_1(F(X)) \leq \pi_1(F([\underline{X}, \underline{X}]))$, and, therefore, $\pi_1(F(X)) = \pi_1(F([\underline{X}, \underline{X}]))$. Following an analogous reasoning, it is possible to prove that $\pi_2(F(X)) = \pi_2(F([\overline{X}, \overline{X}]))$. We conclude that $F = \mathbb{I}_{[\underline{F}, \overline{F}]}$. The $\ll$-decreasing case is analogous but considering $[\overline{X}, \overline{X}]$ instead of $[\underline{X}, \underline{X}]$.

Proposition 3. *Consider $F : \mathbb{I}_{[a,b]} \to \mathbb{I}_{[c,d]}$. If $F = \mathbb{I}_{[f,g]}$, for some (strictly) increasing (decreasing) functions $f, g : [a, b] \to [c, d]$, then $\underline{F} = f$ and $\overline{F} = g$.*

Proof. For $x \in [a, b]$, one has that $\underline{F}(x) = \pi_1(F([x, x])) = \pi_1(\mathbb{I}_{[f,g]}([x, x])) = \pi_1([f(x), g(x)]) = f(x)$. The case of $\overline{F}$ is analogous.

3 Additive Generators and T-Norms

The definitions and properties included in this section were extracted from [11, 12] [14, 15].

Consider the real unit interval $U = [0, 1]$. A t-norm is an increasing function $T : U^2 \to U$ that is commutative, associative and has 1 as neutral element. Typical examples of t-norms are: (1) *Gödel* or *minimum*: $G(x, y) = \min(x, y)$; (2) *Product*: $P(x, y) = x \cdot y$; (3) *Łukasiewicz*: $L(x, y) = \max(x + y - 1, 0)$. Notice that, $L \leq P$, that is, for each $x, y \in U$, it holds that $L(x, y) \leq P(x, y)$.

Definition 3. *Let $f : [a, b] \to [c, d]$ be a monotonic function. The function $f^{(-1)} : [c, d] \to [a, b]$ defined by*

$$f^{(-1)}(y) = \begin{cases} \sup\{x \in [a, b] \mid f(x) < y\} & \text{if } f(a) < f(b), \\ \sup\{x \in [a, b] \mid f(x) > y\} & \text{if } f(a) > f(b), \\ a & \text{if } f(a) = f(b) \end{cases} \tag{4}$$

is called the pseudo-inverse of f.

Notice that when f is a strictly increasing function then so is $f^{(-1)}$ and

$$f^{(-1)}(y) = \sup\{x \in [a,b] \mid f(x) < y\}. \tag{5}$$

Analogously, when f is a strictly decreasing function then so is $f^{(-1)}$ and

$$f^{(-1)}(y) = \sup\{x \in [a,b] \mid f(x) > y\}. \tag{6}$$

Thus, if f is strictly increasing or decreasing then so is $f^{(-1)}$.

Example 1. The functions $p, l : U \to [0,\infty]$, defined, respectively, by: $p(x) = -\ln x$ if $x \neq 0$, and $p(x) = \infty$ if $x = 0$, and $l(x) = 1 - x$, are strictly decreasing and $p(1) = l(1) = 0$. In this case, for each $y \in [0,\infty]$, it holds that $p^{(-1)}(y) = e^{-y}$, and $l^{(-1)}(y) = 1 - y$ if $y \in [0,1]$, and $l^{(-1)}(y) = 0$ if $y > 1$.

Theorem 1. *Let $f : U \to [0,\infty]$ be a strictly decreasing function with $f(1) = 0$ such that for each $(x,y) \in U^2$*

$$f(x) + f(y) \in Ran(f) \cup [f(0),\infty]. \tag{7}$$

Then, the function $T_f : U^2 \to U$, defined by

$$T_f(x,y) = f^{(-1)}(f(x) + f(y)), \tag{8}$$

is a t-norm. In this case, the function f is called an additive generator *of the t-norm T_f.*

Example 2. Consider $x, y \in U$. We have that $T_l(x,y) = l^{(-1)}(l(x)+l(y)) = l^{(-1)}((1-x) + (1-y))$. Thus, if $x + y < 1$ then $T_l(x,y) = 0$, and if $x + y \geq 1$ then $T_l(x,y) = x+y-1$. Thus, it holds that $T_l = L$. Analogously, we have that $T_p(x,y) = p^{(-1)}(p(x)+p(y)) = e^{(\ln x + \ln y)} = e^{(\ln x \cdot y)} = x \cdot y$. It follows that $T_p = P$.

Proposition 4. *Let $f, g : U \to [0,\infty]$ be strictly decreasing functions with $f(1) = g(1) = 0$. Then, it holds that if $f \leq g$ then $T_f \leq T_g$.*

Proof. If $f \leq g$, then, for each $x \in U$, $f(x) \leq g(x)$ and $f^{(-1)}(x) \leq g^{(-1)}(x)$. Thus, for each $x, y \in U$, it follows that

$$\begin{aligned} T_f(x,y) &= f^{(-1)}(f(x) + f(y)) \\ &\leq f^{(-1)}(g(x) + g(y)) \\ &\leq g^{(-1)}(g(x) + g(y)) \\ &= T_g(x,y). \end{aligned}$$

4 Interval T-Norms

Based on the main contribution of the interval generalization proposed in [47], an interval triangular norm may be considered as an interval representation of a t-norm. This generalization fits with the fuzzy principle, which means that the interval membership degree may be thought as an approximation of the exact degree [33,40,41,28,37]. In the following, the extension of the t-norm notion for $\mathbb{I}_U$ introduced in [47,48] is considered.

Definition 4. *A function* $\mathbb{T} : \mathbb{I}_U^2 \to \mathbb{I}_U$ *is an* interval t-norm *if it is commutative, associative, monotonic with respect to the product and inclusion order and* $[1,1]$ *is the neutral element.*

Example 3. Consider $\mathbb{P}, \mathbb{L}, \mathbb{LP} : \mathbb{I}_U^2 \to \mathbb{I}_U$, defined by $\mathbb{P}(X,Y) = [\underline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}]$,

$$\mathbb{L}(X,Y) = \begin{cases} [\underline{X} + \underline{Y} - 1, \overline{X} + \overline{Y} - 1] & \text{if } \underline{X} + \underline{Y} \geq 1 \\ [0,0] & \text{if } \overline{X} + \overline{Y} \leq 1 \\ [0, \overline{X} + \overline{Y} - 1] & \text{otherwise,} \end{cases}$$

$$\mathbb{LP}(X,Y) = \begin{cases} [\underline{X} + \underline{Y} - 1, \overline{X} \cdot \overline{Y}] & \text{if } \underline{X} + \underline{Y} \geq 1 \\ [0, \overline{X} \cdot \overline{Y}] & \text{otherwise.} \end{cases}$$

Notice that $\mathbb{P} = \widehat{P}$, $\mathbb{L} = \widehat{L}$ and $\mathbb{LP} = \mathbb{I}_{[L,P]}$, and, therefore, they are interval t-norms.

Proposition 5. *A function* $\mathbb{T} : \mathbb{I}_U^2 \to \mathbb{I}_U$ *is an interval t-norm if and only if there exist t-norms* T_1 *and* T_2 *such that* $T_1 \leq T_2$ *and* $\mathbb{T} = \mathbb{I}_{[T_1,T_2]}$.

Proof. See [48].

Notice that, in this case, $\mathbb{T}$ is an interval representation of any t-norm T such that $T_1 \leq T \leq T_2$, $\underline{\mathbb{T}} = T_1$ and $\overline{\mathbb{T}} = T_2$. Moreover, if $T_1 = T_2$ then we have that $\mathbb{T} = \widehat{T_1}$.

5 Interval Additive Generators of Interval T-Norms

Definition 5. *Let* $[a,b], [c,d] \in \mathbb{I}_{[-\infty,\infty]}$ *and* $F : \mathbb{I}_{[a,b]} \to \mathbb{I}_{[c,d]}$ *be a monotonic function with respect to the product order, such that* $F([a,a]) \ll F([b,b])$, $F([a,a]) \gg F([b,b])$ *or* $F([a,a]) = F([b,b])$. *The function defined by*

$$F^{(-1)}(Y) = \begin{cases} \sup\{X \in \mathbb{I}_{[a,b]} \mid F(X) \ll Y\} & \text{if } F([a,a]) \ll F([b,b]), \\ \sup\{X \in \mathbb{I}_{[a,b]} \mid F(X) \gg Y\} & \text{if } F([a,a]) \gg F([b,b]), \\ [a,a] & \text{if } F([a,a]) = F([b,b]), \end{cases} \tag{9}$$

is called the pseudo-inverse of F.

Analogously to the punctual case, when F is $\ll$-increasing then $F^{(-1)}$ also is and

$$F^{(-1)}(Y) = \sup\{X \in \mathbb{I}_{[a,b]} \mid F(X) \ll Y\}. \tag{10}$$

Analogously, when F is $\ll$-decreasing then $F^{(-1)}$ also is and

$$F^{(-1)}(Y) = \sup\{X \in \mathbb{I}_{[a,b]} \mid F(X) \gg Y\}. \tag{11}$$

Example 4. Consider $F_l, F_p : \mathbb{I}_U \to \mathbb{I}_{[0,\infty]}$, defined, respectively, by $F_l(X) = [1,1] - X$, and $F_p(X) = -[\ln \underline{X}, \ln \overline{X}]$. These functions are both $\ll$-decreasing, and $F_l([1,1]) = F_p([1,1]) = [0,0]$. Clearly, it holds that $F_l = \widehat{l}$, $F_p = \widehat{p}$. Notice that $F_l^{(-1)}, F_p^{(-1)} : \mathbb{I}_{[0,\infty]} \to \mathbb{I}_U$ can be expressed, respectively, by $F_l^{(-1)}(Y) = [1,1] - Y$ and $F_p^{(-1)}(Y) = [e^{-\overline{Y}}, e^{-\underline{Y}}]$.

Lemma 2. *Let $f, g : [a,b] \to [c,d]$ be strictly increasing (decreasing) functions, where $[a,b],[c,d] \in \mathbb{I}_{[-\infty,\infty]}$ and $f \leq g$. Then $\mathbb{I}_{[f,g]}$ is an $\ll$-increasing (decreasing) interval function.*

Proof. Consider $X, Y \in \mathbb{I}_{[a,b]}$. If $X \ll Y$ then $\underline{X} < \underline{Y}$ and $\overline{X} < \overline{Y}$. So, since f and g are strictly increasing, then $f(\underline{X}) < f(\underline{Y})$ and $g(\overline{X}) < g(\overline{Y})$. Therefore, it follows that $\mathbb{I}_{[f,g]}(X) \ll \mathbb{I}_{[f,g]}(Y)$. The case where f and g are strictly decreasing is analogous.

Theorem 2. *Let $f, g : [a,b] \to [c,d]$ be strictly increasing (decreasing) functions where $[a,b],[c,d] \in \mathbb{I}_{[-\infty,\infty]}$ and $f \leq g$. Then it holds that $\mathbb{I}_{[f^{(-1)},g^{(-1)}]} = \mathbb{I}^{(-1)}_{[f,g]}$.*

Proof. If f and g are both strictly increasing, then $f^{(-1)}$ and $g^{(-1)}$ also are strictly increasing and $f^{(-1)} \leq g^{(-1)}$. Then, it follows that $\mathbb{I}_{[f^{(-1)},g^{(-1)}]}$ is well defined and, by Lemma 2, it is $\ll$-increasing. Therefore, we have that:

$$\begin{aligned}
\mathbb{I}_{[f^{(-1)},g^{(-1)}]}(Y) &= [f^{(-1)}(\underline{Y}), g^{(-1)}(\overline{Y})] \\
&= [\sup\{x \in [a,b] \mid f(x) < \underline{Y}\}, \sup\{x \in [a,b] \mid g(x) < \overline{Y}\}] \\
&= \sup\{[x_1,x_2] \in \mathbb{I}_{[a,b]} \mid f(x_1) < \underline{Y} \text{ and } g(x_2) < \overline{Y}\} \\
&= \sup\{[x_1,x_2] \in \mathbb{I}_{[a,b]} \mid \mathbb{I}_{[f,g]}([x_1,x_2]) \ll Y\} = \mathbb{I}^{(-1)}_{[f,g]}(Y).
\end{aligned}$$

Analogously, if f and g are both strictly decreasing, then $f^{(-1)}$ and $g^{(-1)}$ also are strictly decreasing, and $f^{(-1)} \leq g^{(-1)}$. Then, $\mathbb{I}_{[f^{(-1)},g^{(-1)}]}$ is well defined and, by Lemma 2, it follows that it is $\ll$-decreasing. Therefore, we have that:

$$\begin{aligned}
\mathbb{I}_{[f^{(-1)},g^{(-1)}]}(Y) &= [f^{(-1)}(\overline{Y}), g^{(-1)}(\underline{Y})] \\
&= [\sup\{x \in [a,b] \mid f(x) > \overline{Y}\}, \sup\{x \in [a,b] \mid g(x) > \underline{Y}\}] \\
&= \sup\{[x_1,x_2] \in \mathbb{I}_{[a,b]} \mid f(x_2) > \overline{Y} \text{ and } g(x_1) > \underline{Y}\} \\
&= \sup\{[x_1,x_2] \in \mathbb{I}_{[a,b]} \mid \mathbb{I}_{[f,g]}([x_1,x_2]) \gg Y\} = \mathbb{I}^{(-1)}_{[f,g]}(Y).
\end{aligned}$$

Corollary 1. *Let $f : [a,b] \to [c,d]$ be a strictly increasing (decreasing) function, with $[a,b],[c,d] \in \mathbb{I}_{[-\infty,\infty]}$. Then $\widehat{f}$ is $\ll$-increasing (decreasing) and $\widehat{f}^{(-1)} = \widehat{f^{(-1)}}$.*

Proof. It follows from Lemma 2, Theorem 2 and the fact that $\widehat{f} = \mathbb{I}_{[f,f]}$.

Proposition 6. *Let $F : \mathbb{I}_{[a,b]} \to \mathbb{I}_{[c,d]}$ be a $\ll$-increasing (decreasing) function, with $[a,b],[c,d] \in \mathbb{I}_{[-\infty,\infty]}$. Then, $\underline{F^{(-1)}} = \underline{F}^{(-1)}$ and $\overline{F^{(-1)}} = \overline{F}^{(-1)}$, where $\underline{F}, \overline{F} : [a,b] \to [c,d]$ and $\underline{F^{(-1)}}, \overline{F^{(-1)}} : [c,d] \to [a,b]$ are the projections functions defined in equations (1) and (2).*

Proof. Suppose that F is $\ll$-increasing. Then, $F^{(-1)}$ is also $\ll$-increasing, and, by Prop. 1, $\underline{F}$ and $\underline{F}^{(-1)}$ are $\ll$-increasing too. By equations (1), (2) and (10), we have that $\underline{F^{(-1)}}(y) = \pi_1(\sup\{X \in \mathbb{I}_{[a,b]} \mid F(X) \gg [y,y]\})$. Thus, by Remark 1 and because $X \leq [\pi_2(X), \pi_2(X)]$ and F is $\ll$-increasing, then $\sup\{X \in \mathbb{I}_{[a,b]} \mid F(X) \gg [y,y]\} = \sup\{[x,x] \in \mathbb{I}_{[a,b]} \mid F([x,x]) \gg [y,y]\}$. It follows that $\underline{F^{(-1)}}(y) = \pi_1(\sup\{[x,x] \in \mathbb{I}_{[a,b]} \mid F([x,x]) > [y,y]\}) = \sup\{\pi_1([x,x]) \in \mathbb{I}_{[a,b]} \mid \pi_1(F([x,x])) > \pi_1([y,y])\} = \sup\{x \in [a,b] \mid \underline{F}(x) > y\} = \underline{F}^{(-1)}(y)$. The other cases are analogous.

Theorem 3. *Let $F : \mathbb{I}_U \to \mathbb{I}_{[0,\infty]}$ be a $\ll$-decreasing function, with $F([1,1]) = [0,0]$, such that $F(X) + F(Y) \in Ran(F) \cup \{Z \in \mathbb{I}_{[\infty,\infty]} \mid F([0,0]) \leq Z\}$, for all $(X,Y) \in \mathbb{I}_U^2$. The function $\mathbb{T}_F : \mathbb{I}_U^2 \to \mathbb{I}_U$, given by*

$$\mathbb{T}_F(X,Y) = F^{(-1)}(F(X) + F(Y)), \tag{12}$$

where $F^{(-1)}$ is the pseudo-inverse of F, is an interval t-norm. The interval function F is called the interval additive generator *of the interval t-norm $\mathbb{T}_F$.*

Proof. By Prop. 5, there exist t-norms $\underline{T_F}$ and $\overline{T_F}$ such that $\mathbb{T}_F(X,Y) = [\underline{T_F}(\pi_1(X),\pi_1(Y)), \overline{T_F}(\pi_2(X),\pi_2(Y))]$. Thus, the following properties hold:

Commutativity: From Eq. (12), it follows that $\mathbb{T}_F(X,Y) = \mathbb{T}_F(Y,X)$.

Associativity: It follows that:

$$\begin{aligned}
&\mathbb{T}_F(X,\mathbb{T}_F(Y,Z)) = \mathbb{T}_F(X,[\underline{T_F}(\pi_1(Y),\pi_1(Z)),\overline{T_F}(\pi_2(Y),\pi_2(Z))])\\
&= [\underline{T_F}(\pi_1(X),\pi_1([\underline{T_F}(\pi_1(Y),\pi_1(Z)),\overline{T_F}(\pi_2(Y),\pi_2(Z))])),\\
&\quad \overline{T_F}(\pi_2(X),\pi_2([\underline{T_F}(\pi_1(Y),\pi_1(Z)),\overline{T_F}(\pi_2(Y),\pi_2(Z))]))]\\
&= [\underline{T_F}(\pi_1(X),\underline{T_F}(\pi_1(Y),\pi_1(Z))),\overline{T_F}(\pi_2(X),\overline{T_F}(\pi_2(Y),\pi_2(Z)))]\\
&= [\underline{T_F}(\underline{T_F}(\pi_1(X),\pi_1(Y)),\pi_1(Z)),\overline{T_F}(\overline{T_F}(\pi_2(X),\pi_2(Y)),\pi_2(Z))]\\
&= \mathbb{T}([\underline{T_F}(\pi_1(X),\pi_1(Y)),\overline{T_F}(\pi_2(X),\pi_2(Y))],Z) = \mathbb{T}(\mathbb{T}(X,Y),Z)
\end{aligned}$$

Monotonicity: If $Y \leq Z$ then, since F is decreasing, it follows that $F(X) + F(Y) \geq F(X) + F(Z)$ and then $F^{(-1)}$ is also decreasing. Then, we have that $\mathbb{T}_F(X,Y) = F^{(-1)}(F(X) + F(Y)) \leq F^{(-1)}(F(X) + F(Z)) = \mathbb{T}_F(X,Z)$.

$\subseteq$-Monotonicity: If $Y \subseteq Z$ then $\pi_1(Z) \leq \pi_1(Y)$ and $\pi_2(y) \leq \pi_2(Z)$. So, by the monotonicity of t-norms $\underline{T_F}$ and $\overline{T_F}$, it follows that

$$\begin{aligned}
\mathbb{T}_F(X,Y) &= [\underline{T_F}(\pi_1(X),\pi_1(Y)),\overline{T_F}(\pi_2(X),\pi_2(Y))]\\
&\subseteq [\underline{T_F}(\pi_1(X),\pi_1(Z)),\overline{T_F}(\pi_2(X),\pi_2(Z))] = \mathbb{T}_F(X,Z)
\end{aligned}$$

Example 5. Consider $\gg$-decreasing functions F_p and F_l of Ex. 4. Then we have that $\mathbb{T}_{F_p}(X,Y) = X \cdot Y$ and $\mathbb{T}_{F_l}(X,Y) = [\max(\underline{X} + \underline{Y} - 1, 0), \max(\overline{X} + \overline{Y} - 1)] = \sup\{X + Y - [1,1], [0,0]\}$. It is immediate that $\mathbb{T}_{F_l} = \widehat{T_l} = \widehat{L}$ and $\mathbb{T}_{F_p} = \widehat{T_p} = \widehat{G}$.

Theorem 4. *A strictly decreasing function $f : U \to [0,\infty]$ is an additive generator of a t-norm T_f if and only if $\widehat{f}$ is an interval additive generator of the interval t-norm $\widehat{T_f}$, that is, $\widehat{T_f} = \mathbb{T}_{\widehat{f}}$.*

Proof. ($\Rightarrow$) Consider $X, Y \in \mathbb{I}_U$. If $X \ll Y$, then, because f is strictly decreasing, $f(\overline{Y}) < f(\underline{Y}) < f(\underline{X})$ and $f(\overline{Y}) < f(\overline{X}) < f(\underline{X})$. Then, we have that $\widehat{f}(X) = [f(\overline{X}), f(\underline{X})] \ll [f(\overline{Y}), f(\underline{Y})] = \widehat{f}(Y)$. Thus, we conclude that $\widehat{f}$ is $\ll$-decreasing. By Cor. 1, it holds that $\widehat{f}^{(-1)}(Y) = [f^{(-1)}(\overline{Y}), f^{(-1)}(\underline{Y})]$. So, for each $X, Y \in \mathbb{I}_U$:

$$\begin{aligned}
&\widehat{T_f}(X,Y) = [T_f(\underline{X},\underline{Y}), T_f(\overline{X},\overline{Y})] = [f^{(-1)}(f(\underline{X}) + f(\underline{Y})), f^{(-1)}(f(\overline{X}) + f(\overline{Y}))]\\
&= \widehat{f}^{(-1)}([f(\overline{X}) + f(\overline{Y}), f(\underline{X}) + f(\underline{Y})]) = \widehat{f}^{(-1)}([f(\overline{X}), f(\underline{X})] + [f(\overline{Y}), f(\underline{Y})])\\
&= \widehat{f}^{(-1)}(\widehat{f}(X) + \widehat{f}(Y)) = \mathbb{T}_{\widehat{f}}(X,Y).
\end{aligned}$$

Thus, by Theorem 3, $\widehat{T_f}$ is an interval t-norm with $\widehat{f}$ as the interval additive generator. ($\Leftarrow$) It is immediate that, for each $x, y \in [a, b]$, $\widehat{T_f}([x, x], [y, y]) = [T_f(x, y), T_f(x, y)]$.

Proposition 7. *Let $f, g : U \to [0, \infty]$ be strictly decreasing functions such that $f(1) = g(1) = 0$. If $f \leq g$ then it holds that $\mathbb{T}_{\mathbb{I}_{[f,g]}} \subseteq \mathbb{I}_{[T_f, T_g]}$.*

Proof. Consider $X, Y \in \mathbb{I}_U$. Since clearly $\mathbb{I}_{[f,g]}([1, 1]) = [0, 0]$, and by Lemma 2 $\mathbb{I}_{[f,g]}$ is $\ll$-decreasing, then it follows that

$$
\begin{aligned}
&\mathbb{T}_{\mathbb{I}_{[f,g]}}(X, Y) = \mathbb{I}^{(-1)}_{[f,g]}(\mathbb{I}_{[f,g]}(X) + \mathbb{I}_{[f,g]}(Y)) \\
&= \mathbb{I}^{(-1)}_{[f,g]}([f(\overline{X}), g(\underline{X})] + [f(\overline{Y}), g(\underline{Y})]) = \mathbb{I}^{(-1)}_{[f,g]}([f(\overline{X}) + f(\overline{Y}), g(\underline{X}) + g(\underline{Y})]) \\
&= \sup\{[x_1, x_2] \in \mathbb{I}_U \mid \mathbb{I}_{[f,g]}([x_1, x_2]) \gg [f(\overline{X}) + f(\overline{Y}), g(\underline{X}) + g(\underline{Y})]\} \\
&= \sup\{[x_1, x_2] \in \mathbb{I}_U \mid [f(x_2), g(x_1)] \gg [f(\overline{X}) + f(\overline{Y}), g(\underline{X}) + g(\underline{Y})]\} \\
&= [\sup\{x \in U \mid f(x) > f(\overline{X}) + f(\overline{Y})\}, \sup\{x \in U \mid g(x) > g(\underline{X}) + g(\underline{Y})\}] \\
&\subseteq [\sup\{x \in U \mid f(x) > f(\underline{X}) + f(\underline{Y})\}, \sup\{x \in U \mid g(x) > g(\overline{X}) + g(\overline{Y})\}] \\
&= [f^{(-1)}(f(\underline{X}) + f(\underline{Y})), g^{(-1)}(g(\overline{X}) + g(\overline{Y}))] = [T_f(\underline{X}, \underline{Y}), T_g(\overline{X}, \overline{Y})] \\
&= \mathbb{I}_{[T_f, T_g]}(X, Y).
\end{aligned}
$$

Corollary 2. *Let $\mathbb{T}$ be an interval t-norm and $F : \mathbb{I}_U \to \mathbb{I}_{[0,\infty]}$ be an interval additive generator of $\mathbb{T}$. If F is inclusion monotonic, then $\mathbb{I}_{[T_{\underline{F}}, T_{\overline{F}}]} \subseteq \mathbb{T}$.*

Proof. By Prop. 1, $\underline{F}$ and $\overline{F}$ are strictly decreasing. Since $\underline{F}(1) = \overline{F}(1) = 0$ and $\underline{F} \leq \overline{F}$, then by Prop. 7, we have that $\mathbb{T}_{\mathbb{I}_{[\underline{F}, \overline{F}]}} \subseteq \mathbb{I}_{[T_{\underline{F}}, T_{\overline{F}}]}$. Thus, since, by Prop. 2, $F = \mathbb{I}_{[\underline{F}, \overline{F}]}$, then it follows that $\mathbb{I}_{[T_{\underline{F}}, T_{\overline{F}}]} = \mathbb{T}_F$. Since F is an interval additive generator of $\mathbb{T}$, then it holds that $\mathbb{T}_F = \mathbb{T}$. Therefore, we conclude that $\mathbb{I}_{[T_{\underline{F}}, T_{\overline{F}}]} \subseteq \mathbb{T}$.

Theorem 5. *Let $F : \mathbb{I}_U \to \mathbb{I}_{[0,\infty]}$ be an interval additive generator of an interval t-norm $\mathbb{T}$. If F is inclusion monotonic and F represents a strictly decreasing function $f : U \to [0, \infty]$ such that $f(1) = 0$, then $\mathbb{T}$ represents the t-norm T_f.*

Proof. Consider $X, Y \in \mathbb{I}_U$. For each $x \in X$, since F represents f, then $f(x) \in F(X)$ and so $\pi_1(F(X)) \leq f(x) \leq \pi_2(F(X))$. By Lemma 1, F is decreasing, and then $F([\overline{X}, \overline{X}]) \leq F(X) \leq F([\underline{X}, \underline{X}])$. Thus, it follows that $\underline{F}(\overline{X}) \leq f(x) \leq \overline{F}(\underline{X})$. Analogously, if $y \in Y$, then it holds that $\underline{F}(\overline{Y}) \leq f(y) \leq \overline{F}(\underline{Y})$. Thus, it follows that $\underline{F}(\overline{X}) + \underline{F}(\overline{Y}) \leq f(x) + f(y) \leq \overline{F}(\underline{X}) + \overline{F}(\underline{Y})$. Hence, we have that $\underline{F}^{(-1)}(\overline{F}(\underline{X}) + \overline{F}(\underline{Y})) \leq f^{(-1)}(f(x) + f(y)) \leq \overline{F}^{(-1)}(\underline{F}(\overline{X}) + \underline{F}(\overline{Y}))$. It follows that

$$
\begin{aligned}
T_f(x, y) &\in \mathbb{I}_{[\underline{F}^{(-1)}, \overline{F}^{(-1)}]}([\underline{F}(\overline{X}) + \underline{F}(\overline{Y}), \overline{F}(\underline{X}) + \overline{F}(\underline{Y})]) \\
&= \mathbb{I}^{(-1)}_{[\underline{F}, \overline{F}]}([\underline{F}(\overline{X}) + \underline{F}(\overline{Y}), \overline{F}(\underline{X}) + \overline{F}(\underline{Y})]) && \text{(by Theorem 2)} \\
&= F^{(-1)}([\underline{F}(\overline{X}), \overline{F}(\underline{X})] + ([\underline{F}(\overline{Y}), \overline{F}(\underline{Y})]) && \text{(by Prop. 2)} \\
&= F^{(-1)}(\mathbb{I}_{[\underline{F}, \overline{F}]}(X) + \mathbb{I}_{[\underline{F}, \overline{F}]}(Y)) = F^{(-1)}(F(X) + F(Y)) && \text{(by Prop. 2)} \\
&= \mathbb{T}_F(X, Y) && \text{(by Eq. 12).}
\end{aligned}
$$

6 Conclusion and Final Remarks

The main contribution of this paper is the introduction of interval additive generators of interval t-norms as *interval representations* of punctual additive generators of punctual t-norms, considering both the correctness and the optimality criteria, in an approach that was already considered in our previous work [42, 43, 45, 46, 47, 48]. We also show that interval additive generators satisfy the main properties of punctual additive generators discussed in the literature, and, therefore, they may play an important role for practical applications, by providing more flexibility in the selection of interval t-norms.

On the other hand, related work [57] presented additive and multiplicative generators in interval-valued fuzzy set theory, with a study about the relationship between generators and negators and between generators and t-norms. However, it adopts a different approach from ours, in a way that we explain in the following. The additive and multiplicative generators are defined on the lattice $\mathcal{L}^I$ [2], which is the underlying lattice of interval-valued fuzzy set theory, and a special class of generators on $\mathcal{L}^I$ were investigated, showing that they can be represented by generators on $([0,1],\cdot)$. We notice that a different definition for the interval addition is used, which is not the usual one [24, 25]. Also, the approach is restricted to strictly decreasing functions. Moreover, although they consider the idea of representation of interval t-norms as the one that can be obtained from punctual t-norms, no correctness criterium is applied.

The approach we have adopted seems to have led us to more natural and simple definitions than the ones presented in [57]. Moreover, since we have considered additive generators that are either strictly increasing or strictly decreasing, the results presented in this paper can be easily extended to obtain additive generators of interval t-conorms.

On going work is the study of interval additive generators that are inclusion monotonic, which is a nice property that only makes sense for the interval approach, since there is no counterpart for punctual functions. Further work consists of the investigation of the relationship between the right-continuity property of interval additive generators and the generated interval t-norms. We also intend to analyze the relationships between interval additive generators and the classes of (i) Archimedean interval t-norms and (ii) interval automorphisms. Another future work is the extension of the concept for multiplicative generators, and also to develop an approach for interval t-conorms.

Acknowledgments. This work has been partially supported by CNPq. We are are very grateful to the referees for their valuable suggestions that help us to improve the paper.

References

1. Zadeh, L.A.: Fuzzy sets. Information and Control, 338–353 (1965)
2. Mitra, S., Pal, S.K.: Fuzzy sets in pattern recognition and machine intelligence. Fuzzy Sets and Systems 156, 381–386 (2005)
3. Chen, G., Pham, T.T.: Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems. CRC Press, Boca Raton (2001)

[2] $\mathcal{L}^I = (L^I, \leq_{L^I})$, where $L^I = \{[x_1, x_2] \mid (x_1, x_2) \in [0,1]^2 \text{ and } x_1 \leq x_2\}$ and $\leq_{L^I}$ is the product order (see Sect. 2), is a complete lattice. [57]

4. Carlsson, C., Fuller, R.: Fuzzy Reasoning in Decision Making and Optimization. Physicva-Verlag, Springer, Heidelberg (2002)
5. Siler, W., Buckley, J.J.: Fuzzy Expert Systems and Fuzzy Reasoning. John Wiley, NY (2004)
6. Bedregal, B.C., Costa, A.C.R., Dimuro, G.P.: Fuzzy rule-based hand gesture recognition. In: Bramer, M. (ed.) Artificial Intelligence in Theory And Practice. IFIP Series, vol. 271, pp. 285–294. Springer, Boston (2006)
7. Schweizer, B., Sklar, A.: Probabilistic Metric Spaces. North-Holland, NY (1983)
8. Fodor, J., Roubens, M.: Fuzzy Preference Modelling and Multicriteria Decision Support. Kluwer Academic Publisher, Dordrecht (1994)
9. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer Academic Publishers, Boston (1998)
10. Klement, E., Mesiar, R., Pap, E.: Triangular norms. position paper I: basic analytical and algebraic properties. Fuzzy Sets and Systems 143(1), 5–26 (2004)
11. Klement, E., Mesiar, R., Pap, E.: Triangular norms. position paper II: general constructions and parameterized families. Fuzzy Sets and Systems 145(3), 411–438 (2004)
12. Klement, E., Mesiar, R., Pap, E.: Triangular Norms. Kluwer, Dordrecht (2000)
13. Klement, E., Navara, M.: A survey on different triangular norm-based fuzzy logics. Fuzzy Sets and Systems 101, 241–251 (1999)
14. Klement, E.P., Mesiar, R., Pap, E.: Triangular norms: Basic notions and properties. In: Klement, E.P., Mesiar, R. (eds.) Logical, Algebraic, Analytic, and Probabilistic Aspects of Triangular Norms, pp. 17–60. Elsevier, Amsterdam (2005)
15. Mesiarová, A.: Generators of triangular norms. In: Klement, E.P., Mesiar, R. (eds.) Logical, Algebraic, Analytic, and Probabilistic Aspects of Triangular Norms, pp. 95–111. Elsevier, Amsterdam (2005)
16. Leventides, J., Bounas, A.: An approach to the selection of fuzzy connectives in terms of their additive generators. Fuzzy Sets and Systems 126, 219–224 (2002)
17. Mas, M., Monserrat, M., Torrens, J.: Two types of implications derived from uninorms. Fuzzy Sets and Systems 158(3), 2612–2626 (2007)
18. Faucett, W.M.: Compact semigroups irreducibly connected between two idempotents. Proc. American Mathematics Society 6, 741–747 (1955)
19. Klement, E.P., Mesiar, R., Pap, E.: Quasi- and pseudo-inverses of monotone functions, and the construction of t-norms. Fuzzy Sets and Systems 104, 3–13 (1999)
20. Ling, C.H.: Representation of associative functions. Publ. Math. Debrecen 12, 189–212 (1965)
21. Mesiarová, A.: H-transformation of t-norms. Information Sciences 176, 1531–1545 (2006)
22. Mostert, P.S., Shields, A.L.: On the structure of semigroups on a compact manifold with boundary. Ann. Math. 65, 117–143 (1957)
23. Viceník, P.: Additive generators of associative functions. Fuzzy Sets and Systems 153, 137–160 (2005)
24. Alefeld, G., Herzberger, J.: Introduction to Interval Computations. Academic Press, New York (1983)
25. Moore, R.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)
26. Cornelis, G., Deschrijver, G., Kerre, E.: Implication in intuitionistic fuzzy and interval-valued fuzzy set theory: construction, classification, application. Int. Journal Approximate Reason 35, 55–95 (2004)
27. Deschrijver, G., Kerre, E.E.: Implicators based on binary aggregation operators in interval-valued fuzzy set theory. Fuzzy Sets and Systems 153(2), 229–248 (2005)
28. Dubois, D., Prade, H.: Interval-valued fuzzy sets, possibility theory and imprecise probability. In: Proc. Intl. Conf. Fuzzy Logic and Technology, Barcelona, pp. 314–319 (2005)
29. Gehrke, M., Walker, C., Walker, E.: Some comments on interval valued fuzzy sets. Intl. Journal of Intelligent Systems 11, 751–759 (1996)

30. Gorzalczany, M.B.: A method of inference in approximate reasoning based on interval-valued fuzzy sets. Fuzzy Sets and Systems 21(1), 1–17 (1987)
31. Grattan-Guiness, I.: Fuzzy membership mapped onto interval and many-valued quantities. Z. Math. Logik. Grundladen Math. 22, 149–160 (1975)
32. Moore, R., Lodwick, W.: Interval analysis and fuzzy set theory. Fuzzy Sets and Systems 135(1), 5–9 (2003)
33. Nguyen, H., Kreinovich, V., Zuo, Q.: Interval-valued degrees of belief: applications of interval computations to expert systems and intelligent control. Int. Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems 5(3), 317–358 (1997)
34. Sambuc, R.: Fonctions ϕ-floues. Application l'aide au diagnostic en pathologie thyroidienne. PhD thesis, Univ. Marseille, Marseille (1975)
35. Turksen, I.: Interval valued fuzzy sets based on normal forms. Fuzzy Sets and Systems 20, 191–210 (1986)
36. Gasse, B.V., Cornelis, G., Deschrijver, G., Kerre, E.: On the properties of a generalized class of t-norms in interval-valued fuzzy logics. New Math. and Natural Comput. 2, 29–42 (2006)
37. Lodwick, W.A.: Preface. Reliable Computing 10(4), 247–248 (2004)
38. Moore, R.E.: Interval Arithmetic and Automatic Error Analysis in Digital Computing. PhD thesis, Stanford University, Stanford (1962)
39. Kearfort, R.B., Kreinovich, V. (eds.): Applications of Interval Computations. Kluwer Academic Publishers, Boston (1996)
40. Walley, P.: Statistical Reasoning with Imprecise Probabilities. Chapman&Hall, London (1991)
41. Walley, P.: Measures of uncertainty. Artificial Intelligence 83, 1–58 (1996)
42. Bedregal, B.C., Santiago, R.H.N., Reiser, R.H.S., Dimuro, G.P.: The best interval representation of fuzzy Simplications and automorphisms. In: Proc. of the IEEE Intl. Conf. on Fuzzy Systems, Londres, pp. 3220–3230. IEEE, Los Alamitos (2007)
43. Bedregal, B.C., Santiago, R.H.N., Reiser, R.H.S., Dimuro, G.P.: Analyzing properties of fuzzy implications obtained via the interval constructor. In: IEEE Post-Proceedings of SCAN 2006: Revised Setected Papers of 12th GAMM - IMACS Intl. Symp. Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg, paper no. 13. IEEE, Los Alamitos (2007)
44. Bedregal, B.C., Santiago, R.H.N., Reiser, R.H.S., Dimuro, G.P.: Properties of fuzzy implications obtained via the interval constructor. TEMA 8(1), 33–42 (2007), `http://www.sbmac.org.br/tema`
45. Bedregal, B.C., Santiago, R.H.N., Dimuro, G.P., Reiser, R.H.S.: Interval valued R-implications and automorphisms. In: Pre-Proceedings of the 2nd Work. on Logical and Semantic Frameworks, with Applications, Ouro Preto, pp. 82–97 (2007)
46. Reiser, R.H.S., Dimuro, G.P., Bedregal, B., Santiago, R.: Interval valued QL-implications. In: Leivant, D., de Queiroz, R. (eds.) WoLLIC 2007. LNCS, vol. 4576, pp. 307–321. Springer, Heidelberg (2007)
47. Bedregal, B., Takahashi, A.: The best interval representation of t-norms and automorphisms. Fuzzy Sets and Systems 157(24), 3220–3230 (2006)
48. Bedregal, B., Takahashi, A.: Interval valued versions of t-conorms, fuzzy negations and fuzzy implications. In: IEEE Proc. Intl. Conf. on Fuzzy Systems, Vancouver, Los Alamitos, pp. 9553–9559 (2006)
49. Santiago, R., Bedregal, B., Acióly, B.: Formal aspects of correctness and optimality in interval computations. Formal Aspects of Computing 18(2), 231–243 (2006)
50. Hickey, T., Ju, Q., Emdem, M.: Interval arithmetic: from principles to implementation. Journal of the ACM 48(5), 1038–1068 (2001)
51. Callejas-Bedregal, R., Bedregal, B.C.: Intervals as a domain constructor. TEMA 2(1), 43–52 (2001), `http://www.sbmac.org.br/tema`

52. Abramsky, S., Jung, A.: Domain theory. In: Abramsky, S., Gabbay, D., Maimbaum, T. (eds.) Handbook of Logic in Computer Science, vol. 3. Oxford Svience Publications (1994)
53. Gierz, G., Hoffman, K., Keimel, K., Lawson, J., Scott, D.: Continuous Lattices and Domains. Cambridge University Press, Cambridge (2003)
54. Kearfott, R.B.: Rigorous Global Search: Continuous problems. Kluwer, Dordrecht (1996)
55. Caprani, O., Madsen, K., Stauning, O.: Existence test for asynchronous interval iteration. Reliable Computing 3(3), 269–275 (1997)
56. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analisys: With examples in parameter and state estimation, robust control and robotic. Springer, Heidelberg (2001)
57. Deschrijver, G.: Additive and multiplicative generators in interval-valued fuzzy set theory. IEEE Transactions on Fuzzy Systems 15(2), 1063–6706 (2007)

Propositional Dynamic Logic as a Logic of Belief Revision

Jan van Eijck and Yanjing Wang

Center for Mathematics and Computer Science (CWI)
Kruislaan 413 1098 SJ Amsterdam, The Netherlands
{jve,y.wang}@cwi.nl

Abstract. This paper shows how propositional dynamic logic (PDL) can be interpreted as a logic for multi-agent belief revision. For that we revise and extend the logic of communication and change (LCC) of [9]. Like LCC, our logic uses PDL as a base epistemic language. Unlike LCC, we start out from agent plausibilities, add their converses, and build knowledge and belief operators from these with the PDL constructs. We extend the update mechanism of LCC to an update mechanism that handles belief change as relation substitution, and we show that the update part of this logic is more expressive than either that of LCC or that of doxastic/epistemic PDL with a belief change modality. It is shown that the properties of knowledge and belief are preserved under any update, and that the logic is complete.

Keywords: PDL, epistemic dynamic logic, belief revision, knowledge update.

1 Introduction

Proposals for treating belief revision in the style of dynamic epistemic logic (see Gerbrandy [15], van Ditmarsch [12], van Benthem [6,10], and Baltag, Moss and coworkers [3,1,2], or the textbook treatment in [13]) were made in [8] and [7], where it is suggested that belief revision should be treated as relation substitution. This is different from the standard action product update from [3], and it suggests that the proper relation between these two update styles should be investigated.

We propose a new version of action product update that integrates belief revision as relation substitution with belief update by means of action product. We show that this allows to express updates that cannot be expressed with action product only or with relation substitution only.

We graft this new update mechanism on a base logic that can express knowledge, safe belief, conditional belief, and plain belief, and we show that the proper relations between these concepts are preserved under any update. The completeness of our logic is also provided.

Our main source of inspiration is the logic of communication and change (LCC) from [9]. This system has the flaw that updates with non-S5 action models may

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 136–148, 2008.

destroy knowledge or belief. If one interprets the basic relations as knowledge relations, then updating with a lie will destroy the S5 character of knowledge; similarly, if one interprets the basic relations as belief, the relational properties of belief can be destroyed by malicious updates. Our redesign does not impose any relational conditions on the basic relations, so this problem is avoided. Our completeness proof is an adaptation from the completeness proof for LCC. The treatment of conditional belief derives from [11]. Our work can be seen as a proposal for integrating belief revision by means of relation substitution, as proposed in [7] with belief and knowledge update in the style of [3].

2 PDL as a Belief Revision Logic

A preference model $\mathbf{M}$ for set of agents Ag and set of basic propositions $Prop$ is a tuple (W, P, V) where W is a non-empty set of worlds, P is a function that maps each agent a to a relation P_a (the preference relation for a, with wP_aw' meaning that w' is at least as good as w), and V is a map from W to $\mathcal{P}(Prop)$ (a map that assigns to each world a $Prop$-valuation). A distinctive preference model is a pair consisting of a preference model and a set of distinctive states in that model. The intuitive idea is that the actual world is constrained to be among the distinctive worlds. This information is typically not available to the agents, for an agent's knowledge about what is actual and what is not is encoded in her P_a relation (see below).

There are no conditions at all on the P_a. Appropriate conditions will be imposed by constructing the operators for belief and knowledge by means of PDL operations.

We fix a PDL style language for talking about preference (or: plausibility). Assume p ranges over $Prop$ and a over Ag.

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid [\pi]\phi$$
$$\pi ::= a \mid a^{\smile} \mid ?\phi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^*$$

We use $PROG$ for the set of program expressions (expressions of the form π) of this language.

This is to be interpreted in the usual PDL manner, with $[\![\pi]\!]^{\mathbf{M}}$ giving the relation that interprets relational expression π in $\mathbf{M} = (W, P, V)$. $[\pi]\phi$ is true in world w of $\mathbf{M}$ if for all v with $(w, v) \in [\![\pi]\!]^{\mathbf{M}}$ it holds that ϕ is true in v. We adopt the usual abbreviations of $\bot$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$ and $\langle\pi\rangle\phi$.

The following additional abbreviations allow us to express knowledge, safe belief, conditional belief and plain belief:

Knowledge. $\sim_a$ abbreviates $(a \cup a^{\smile})^*$.
Safe belief. $\geq_a$ abbreviates a^*.
Conditional belief. $[\rightarrow_a^{\phi}]\psi$ abbreviates $\langle\sim_a\rangle\phi \rightarrow \langle\sim_a\rangle(\phi \wedge [\geq_a](\phi \rightarrow \psi))$.
Plain belief. $[\rightarrow_a]\phi$ abbreviates $[\rightarrow_a^{\top}]\phi$. (note: it follows that $[\rightarrow_a]\phi$ is equivalent to $\langle\sim_a\rangle[\geq_a]\phi$).

We will occasionally use $\leq_a$ for the converse of $\geq_a$.

Safe belief is belief that persists under revision with true information (see Stalnaker [20]). The definition of $[\rightarrow_a^\phi]\psi$ (conditional belief for a, with condition ϕ) is from Boutillier [11] This definition, also used in [5], states that conditional to ϕ, a believes in ψ if either there are no accessible ϕ worlds, or there is an accessible ϕ world in which the belief in $\phi \rightarrow \psi$ is safe. The definition of $[\rightarrow_a^\phi]\psi$ matches the well-known accessibility relations $\rightarrow_a^P$ for each subset P of the domain, given by:

$$\rightarrow_a^P := \{(x, y) \mid x \sim_a y \wedge y \in \mathrm{MIN}_{\leq_a} P\},$$

where $\mathrm{MIN}_{\leq_a} P$, the set of minimal elements of P under $\leq_a$, is defined as

$$\{s \in P : \forall s' \in P(s' \leq_a s \Rightarrow s \leq_a s')\}.$$

This logic is axiomatised by the standard PDL rules and axioms ([19,18]) plus axioms that define the meanings of the converses $a^\smile$ of basic relations a. The PDL rules and axioms are:

Modus ponens	and axioms for propositional logic
Modal generalisation	From $\vdash \phi$ infer $\vdash [\pi]\phi$

Normality	$\vdash [\pi](\phi \rightarrow \psi) \rightarrow ([\pi]\phi \rightarrow [\pi]\psi)$
Test	$\vdash [?\phi]\psi \leftrightarrow (\phi \rightarrow \psi)$
Sequence	$\vdash [\pi_1; \pi_2]\phi \leftrightarrow [\pi_1][\pi_2]\phi$
Choice	$\vdash [\pi_1 \cup \pi_2]\phi \leftrightarrow ([\pi_1]\phi \wedge [\pi_2]\phi)$
Mix	$\vdash [\pi^*]\phi \leftrightarrow (\phi \wedge [\pi][\pi^*]\phi)$
Induction	$\vdash (\phi \wedge [\pi^*](\phi \rightarrow [\pi]\phi)) \rightarrow [\pi^*]\phi$

The relation between the basic programs a and $a^\smile$ is expressed by the standard modal axioms for converse:

$$\vdash \phi \rightarrow [a]\langle a^\smile\rangle\phi \qquad \vdash \phi \rightarrow [a^\smile]\langle a\rangle\phi$$

Any preference relation P_a can be turned into a pre-order by taking its reflexive transitive closure ${P_a}^*$. So our abbreviation introduces the $\geq_a$ as names for these pre-orders. The knowledge abbreviation introduces the $\sim_a$ as names for the equivalences given by $(P_a \cup {P_a}^\smile)^*$. If the P_a are well-founded, $\mathrm{MIN}_{\leq_a} P$ will be non-empty for non-empty P. Wellfoundedness of P_a is the requirement that there is no infinite sequence of different $w_1, w_2, \ldots$ with $\ldots P_a w_2 P_a w_1$. Fortunately, we do not have to worry about this relational property, for the canonical model construction for PDL yields finite models, and each relation on a finite model is well-founded.

This yields a very expressive complete and decidable PDL logic for belief revision, to which we can add mechanisms for belief update and for belief change.

Theorem 1. *The above system of belief revision PDL is complete for preference models. Since the canonical model construction for PDL yields finite models, it is also decidable.*

Knowledge is S5 (equivalence), safe belief is S4 (reflexive and transitive), plain belief is KD45 (serial, transitive and euclidean). Note that the following is valid:

$$\langle\sim_a\rangle[\geq_a]\phi \to [\sim_a]\langle\geq_a\rangle\langle\sim_a\rangle[\geq_a]\phi$$

This shows that plain belief is euclidean.

3 Action Model Update

We give the definition of action models $\mathbf{A}$ and of the update product operation $\otimes$ from Baltag, Moss, Solecki [3]. An action model is like a preference model for Ag, with the difference that the worlds are now called *actions* or *events*, and that the valuation has been replaced by a map **pre** that assigns to each event e a formula of the language called the *precondition* of e. From now on we call the preference models *static models*.

Updating a static model $\mathbf{M} = (W, P, V)$ with an action model $\mathbf{A} = (E, \mathbf{P}, \mathbf{pre})$ succeeds if the set

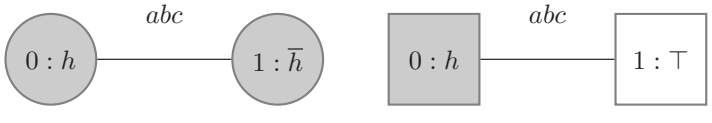
$$\{(w, e) \mid w \in W, e \in E, \mathbf{M}, w \models \mathbf{pre}(e)\}$$

is non-empty. The update result is a new static model $\mathbf{M} \otimes \mathbf{A} = (W', P', V')$ with

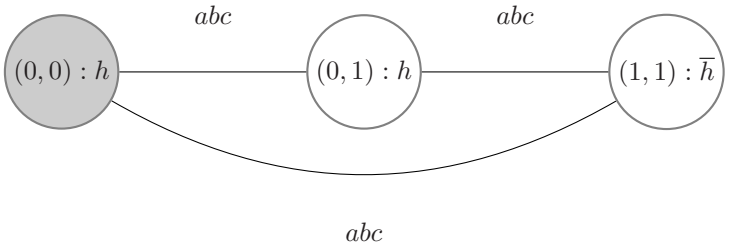
- $W' = \{(w, e) \mid w \in W, e \in E, \mathbf{M}, w \models \mathbf{pre}(e)\}$,
- P'_a is given by $\{(w, e), (v, f)) \mid (w, v) \in P_a, (e, f) \in \mathbf{P}_a\}$,
- $V'(w, e) = V(w)$.

If the static model has a set of distinctive states W_0 and the action model a set of distinctive events E_0, then the distinctive worlds of $\mathbf{M} \otimes A$ are the (w, e) with $w \in W_0$ and $e \in E_0$.

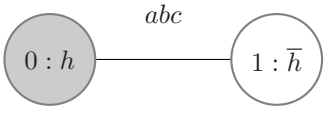
Below is an example pair of a static model with an update action. The static model, on the left, pictures the result of a hidden coin toss, with three onlookers, Alice, Bob and Carol. The model has two distinctive worlds, marked in grey; h in a world means that the valuation makes h true, $\overline{h}$ in a world means that the valuation makes h false in that world. The P_a relations for the agents are assumed to be equivalences; reflexive loops for a, b, c at each world are omitted from the picture.



The action model represents a secret test whether the result of the toss is h. The distinctive event of the update is marked grey. The P_i relations are drawn, for three agents a, b, c. The result of the update is shown here:



This result can be reduced to the bisimilar model below:



The result of the update is that the distinction mark on the $\overline{h}$ world has disappeared, without any of a, b, c being aware of the change.

4 Adding Factual Change and Belief Change

Factual change was already added to update models in LCC. We will now also add belief change. Let an action model with both changes be a quintuple.

$$A = (E, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$$

where $E, \mathbf{P}, \mathbf{pre}$ are as before, **Sub** is a function that assigns a propositional binding to each $e \in E$, and **SUB** is a function that assigns a relational binding to each $e \in E$. A propositional substitution is a map from proposition letters to formulas, represented by a finite set of bindings.

$$\{p_1 \mapsto \phi_1, \ldots, p_n \mapsto \phi_n\}$$

where the p_k are all different, and where no ϕ_k is equal to p_k. It is assumed that each p that does not occur in a left-hand side of a binding is mapped to itself.

Similarly, a relational substitution is a map from agents to program expressions, represented by a finite set.

$$\{a_1 \mapsto \pi_1, \ldots, a_n \mapsto \pi_n\}$$

where the a_j are agents, all different, and where the π_j are program expressions from the PDL language. It is assumed that each a that does not occur in the left-hand side of a binding is mapped to a. Use ϵ for the identity propositional or relational substitution.

Definition 1 (Update execution). *The update execution of static model* $\mathbf{M} = (W, P, V)$ *with action model* $\mathbf{A} = (E, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$ *is a tuple:* $\mathbf{M} \circledast \mathbf{A} = (W', P', V')$ *where:*

- $W' = \{(w, e) \mid \mathbf{M}, w \vDash \mathbf{pre}(e)\}$.
- P'_a *is given by*

$$\{((w_1, e_1), (w_2, e_2)) \mid \textit{there is a } \mathbf{SUB}(e_1)(a) \textit{ path from } (w_1, e_1) \textit{ to } (w_2, e_2) \textit{ in } \mathbf{M} \otimes \mathbf{A}\}.$$

- $V'(p) = \{(w, e) \in W' \mid \mathbf{M}, w \vDash \mathbf{Sub}(e)(p)\}$.

Note: the definition of P'_a refers to paths in the old style update product.

Consider the suggestive upgrade $\sharp_a\phi$ discussed in Van Benthem and Liu [8] as a relation changer (*uniform* relational substitution):

$$\sharp_a\phi =_{\text{def}} \ ?\phi; a; ?\phi \ \cup ?\neg\phi; a; ?\neg\phi \ \cup ?\neg\phi; a; ?\phi.$$

This models a kind of belief change where preference links from ϕ worlds to $\neg\phi$ worlds for agent a get deleted. It can be modelled as the following example of public belief change.

Example 1 (Public Belief Change). Action model

$$G = (\{e\}, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$$

where:

- For all the $i \in Ag$, $\mathbf{P}_i = \{(e, e)\}$.
- $\mathbf{pre}(e) = \top$.
- $\mathbf{Sub}(e) = \epsilon$.
- $\mathbf{SUB}(e) = \{a \mapsto \ \sharp_a\phi, b \mapsto \ \sharp_b\phi\}$.

Note that our action model and its update execution implement the *point-wise* relation substitutions which is more powerful than merely upgrading the relations *uniformly* everywhere in the model, as the following example shows:

Example 2 (Non-public Belief Change). Action model

$$G' = (\{e_0, e_1\}, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$$

where:

- For all $i \in Ag$, if $i \neq b$ then $\mathbf{P}_i = \{(e_0, e_0), (e_1, e_1)\}$, $\mathbf{P}_b = \{(e_0, e_0), (e_1, e_1), (e_0, e_1), (e_1, e_0)\}$
- $\mathbf{pre}(e_0) = \mathbf{pre}(e_1) = \top$.
- $\mathbf{Sub}(e_0) = \mathbf{Sub}(e_1) = \epsilon$.
- $\mathbf{SUB}(e_0) = \{a \mapsto \ \sharp_a\phi\}$, $\mathbf{SUB}(e_1) = \epsilon$.

Assume e_0 is the actual event.

This changes the belief of a while b remains unaware of the change.

Let PDL$^+$ be the result of adding modalities of the form $[\mathbf{A}, e]\phi$ to PDL, with the following interpretation clause:

$$\mathbf{M}, w \models [\mathbf{A}, e]\phi \ \text{ iff } \ \mathbf{M}, w \models \mathbf{pre}(e) \text{ implies } \mathbf{M} \circledast \mathbf{A}, (w, e) \models \phi.$$

Theorem 2 (Soundness and Completeness for PDL$^+$). $\vDash \phi$ *iff* $\vdash \phi$.

Proof. Completeness can be proved by a patch of the LCC completeness proof in [9] where the action modalities are pushed through program modalities by program transformations. See the first Appendix.

5 Expressivity of Action Update with Changes

Although PDL$^+$ reduces to PDL, just like LCC, the new action update mechanism (the model transformation part) is more *expressive* than classic product update and product update with factual changes, as we will show in this section. Call a function on epistemic models that is invariant for bisimulation a model transformer. Then each update can be viewed as a model transformer, and a set of model transformers corresponds to an update mechanism. If U is an update mechanism, let $\mathrm{Tr}(U)$ be its set of model transformers.

Definition 2. *Update mechanism U_1 is* less expressive *than update mechanism U_2 if $Tr(U_1) \subset Tr(U_2)$.*

First note that the classical product update (with factual changes) has the *eliminative* nature for relational changing: according to the definition, the relations in the updated model must come from relations in the static model. For example, it is not possible, by product update, to introduce a relational link for agent a to a static model where the a relation was empty. However, we can easily do this with an uniform relation substitution $a \mapsto ?\top$. Thus we have:

Proposition 1. *Relational substitution can express updates that cannot be expressed with action product update(with factual changes) alone, so relational substitution is not less expressive than action product update.*

On the other hand, relational substitution alone cannot add worlds into a static model, while the classical product update mechanism can copy sets of worlds. Therefore it is not hard to see:

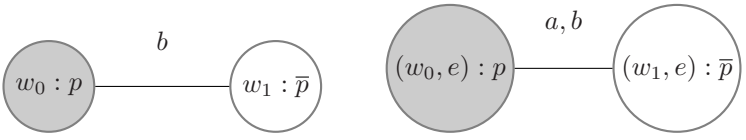
Proposition 2. *Action product update can express updates that cannot be expressed with relational substitution alone, so action product update is not less expressive than relational substitution.*

Our action update with both relational and factual changes combines the power of product update and propositional/relational substitutions. Thus according to Propositions 1 and 2, it is more expressive than relational eliminative product update with factual changes in LCC, and more expressive than propositional/relation changing substitution *simpliciter*. Moreover, we can prove a even stronger result for the case of $S5$ updates.

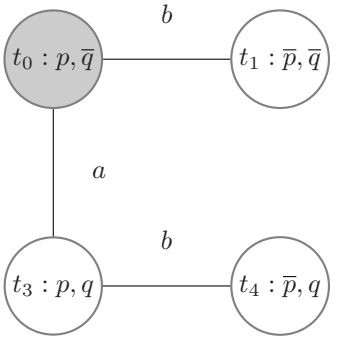
Theorem 3. *In the class of $S5$ model transformers, action product update with factual changes is less expressive than action update with both factual and relational changes.*

Proof. Let $\mathbf{A}$ be the action model $(\{e\}, \mathbf{P}, \mathbf{pre}, \mathbf{Sub}, \mathbf{SUB})$ where $\mathbf{P}_a = \{(e, e)\} = \mathbf{P}_b; \mathbf{pre}(e) = \top; \mathbf{Sub}(e) = \epsilon; \mathbf{SUB}(e) = \{a \mapsto b\}$. It is easy to see that this action model will change the relation a as b uniformly while keeping the updated model being still $S5$, if the static model is indeed $S5$. We now show that it does not have a corresponding action model in LCC style (only factual changes) which can give the bisimilar updated result for every static model.

First consider the following static $S5$ model $\mathbf{M}$ (left) and its updated model $\mathbf{M} \circledast \mathbf{A}$(right) (reflexive loops are omitted):



For a contradiction, suppose there is a LCC action model $\mathbf{A}'$ with distinctive event e' such that $M_1 \circledast \mathbf{A}, (w_0, e) \underline{\leftrightarrow} \mathbf{M} \otimes \mathbf{A}', (w_0, e')$. Then according to the definition of bisimulation, there must be an $a-$link from (w_0, e') to a $\overline{p}$ world (s, e'') in $\mathbf{M} \otimes \mathbf{A}'$. According to the definition of $\otimes$, $(w_0, s) \in p_a$ in $\mathbf{M}$ and $(e', e'') \in \mathbf{P}_a$ in $\mathbf{A}'$. Thus $s = w_0$ and $\mathbf{M}, w_0 \vDash pre(e') \wedge pre(e'')$. Let us consider the following $S5$ model $\mathbf{M}'$ which consists of two copies of $\mathbf{M}$ with an $a-$link in between:



where q does not show up in $pre(e')$ and $pre(e'')$. Thus it is not hard to see that $pre(e') \wedge pre(e')$ holds on t_0 and t_3. Then $\mathbf{M}' \otimes \mathbf{A}', (t_0, e')$ must has an a link from a $\overline{q}$ world (t_0, e') to a q world (t_3, e''), while in $\mathbf{M}' \circledast \mathbf{A}, (t_0, e)$ there is no such link. Thus $\mathbf{M}' \circledast \mathbf{A}, (t_0, e)$ and $\mathbf{M}' \otimes \mathbf{A}', (t_0, e')$ are not bisimilar. Contradiction.

An example illustrating the use of the new belief revision update mechanism is worked out in the second Appendix. This example also shows the difference in expressive power for the achievement of common knowledge between knowledge update and belief revision.

6 Future Work

Several update mechanisms for dynamic epistemic logic have been proposed in the literature. A very expressive one is the action-priority upgrade proposed in [4,5]. Comparing the expressiveness of our update with factual and relation change with that of their mechanism is future work.

The new update mechanism proposed above is grafted on a doxastic/epistemic logic that does not impose any conditions on the basic preference relations. Thus, any update will result in a proper epistemic model. This situation changes as

soon as one imposes further conditions. E.g., if the basic preferences are assumed to be locally connected, then one should restrict the class of update models to those that preserve this constraint. For each reasonable constraint, there is a corresponding class of model transformers that preserve this constraint. Finding syntactic characterizations of these classes is future work.

We are interested in **model checking** with doxastic/epistemic PDL and updates/upgrades in the new style, and we are currently investigating its complexity. We intend to use the logic, and the new update/upgrade mechanism, in the next incarnation of the epistemic model checker DEMO [14].

Acknowledgments. We have profited from discussions with Johan van Benthem, Hans van Ditmarsch and Barteld Kooi, and we thank two anonymous referees for their comments. The second author is supported by the Dutch Organisation for Scientic Research (NWO) under research grant no. 612.000.528 (VEMPS).

References

1. Baltag, A.: A logic for suspicious players: epistemic action and belief-updates in games. Bulletin of Economic Research 54(1), 1–45 (2002)
2. Baltag, A., Moss, L.S.: Logics for epistemic programs. Synthese 139(2), 165–224 (2004)
3. Baltag, A., Moss, L.S., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. In: Bilboa, I. (ed.) Proceedings of TARK 1998, pp. 43–56 (1998)
4. Baltag, A., Smets, S.: Conditional doxastic models: A qualitative approach to dynamic belief revision. Electronic Notes in Theoretical Computer Science (ENTCS) 165, 5–21 (2006)
5. Baltag, A., Smets, S.: A qualitative theory of dynamic interactive belief revision. In: Bonanno, G., van der Hoek, W., Wooldridge, M. (eds.) Texts in Logic and Games. Amsterdam University Press (to appear, 2008)
6. van Benthem, J.: Language, logic, and communication. In: van Benthem, J., Dekker, P., van Eijck, J., de Rijke, M., Venema, Y. (eds.) Logic in Action, pp. 7–25. ILLC (2001)
7. van Benthem, J.: Dynamic logic for belief revision. Journal of Applied Non-Classical Logics 2, 129–155 (2007)
8. van Benthem, J., Liu, F.: Dynamic logic of preference upgrade. Journal of Applied Non-Classical Logics 14(2) (2004)
9. van Benthem, J., van Eijck, J., Kooi, B.: Logics of communication and change. Information and Computation 204(11), 1620–1662 (2006)
10. van Benthem, J.: One is a lonely number: on the logic of communication. Technical Report PP-2002-27, ILLC, Amsterdam (2002)
11. Boutilier, C.: Toward a logic of qualitative decision theory. In: Doyle, J., Sandewall, E., Torasso, P. (eds.) Proceedings of the 4th International Conference on Principle of Knowledge Representation and Reasoning (KR 1994), pp. 75–86. Morgan Kaufmann, San Francisco (1994)
12. van Ditmarsch, H.: Knowledge Games. PhD thesis, ILLC, Amsterdam (2000)

13. van Ditmarsch, H.P., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Synthese Library, vol. 337. Springer, Heidelberg (2006)
14. van Eijck, J.: DEMO — a demo of epistemic modelling. In: van Benthem, J., Gabbay, D., Löwe, B. (eds.) Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop, number 1 in Texts in Logic and Games, pp. 305–363. Amsterdam University Press (2007)
15. Gerbrandy, J.: Bisimulations on planet Kripke. PhD thesis, ILLC (1999)
16. Gray, J.: Notes on database operating systems. In: Bayer, R., Graham, R.M., Seegmuller, G. (eds.) Operating Systems: an advanced course. LNCS, vol. 66, Springer, Berlin (1978)
17. Halpern, J.Y., Moses, Y.O.: Knowledge and common knowledge in a distributed environment. In: Proceedings 3rd ACVM Symposium on Distributed Computing, pp. 50–68 (1984)
18. Kozen, D., Parikh, R.: An elementary proof of the completeness of PDL. Theoretical Computer Science 14, 113–118 (1981)
19. Segerberg, K.: A completeness theorem in the modal logic of programs. In: Traczyck, T. (ed.) Universal Algebra and Applications, pp. 36–46. Polish Science Publications (1982)
20. Stalnaker, R.C.: On logics of knowledge and belief. Philosophical Studies 128, 169–199 (2006)

Appendix 1: Soundness and Completeness of PDL$^+$

To define the proper program transformation for PDL$^+$ we need a function $^\cup$ that maps each PDL program to its converse (in the obvious sense that the interpretation of $\pi^\cup$ is the converse of that of π):

$$\begin{array}{ll}
(a^\smile)^\cup & = a \\
(?\phi)^\cup & = ?\phi \\
(\pi_1;\pi_2)^\cup & = {\pi_2}^\cup;{\pi_1}^\cup \\
(\pi_1 \cup \pi_2)^\cup & = {\pi_1}^\cup \cup {\pi_2}^\cup \\
(\pi^*)^\cup & = (\pi^\cup)^*
\end{array}$$

What is needed to get a completeness proof is a redefinition of the epistemic program transformation operation $T^{\mathbf{A}}_{ij}$ used in the LCC completeness to push an action model modality $[\mathbf{A}, e]$ through an epistemic program modality $[\pi]$.

$$\begin{array}{rcl}
\underline{T}^{\mathbf{A}}_{ij}(a) & = & \begin{cases} ?pre(e_i);\mathbf{SUB}(e_i)(a) & \text{if } e_i \mapsto_{\mathbf{SUB}(e_i)(a)} e_j \text{ in } \mathbf{A} \\ ?\bot & \text{otherwise} \end{cases} \\
\underline{T}^{\mathbf{A}}_{ij}(a^\smile) & = & \begin{cases} ?pre(e_i);(\mathbf{SUB}(e_i)(a))^\cup & \text{if } e_i \mapsto_{(\mathbf{SUB}(e_i)(a))^\cup} e_j \text{ in } \mathbf{A} \\ ?\bot & \text{otherwise} \end{cases} \\
\underline{T}^{\mathbf{A}}_{ij}(?\phi) & = & \begin{cases} ?(pre(e_i) \wedge [\mathbf{A}, e_i]\phi) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases} \\
\underline{T}^{\mathbf{A}}_{ij}(\pi_1;\pi_2) & = & \bigcup_{k=0}^{n-1}(\underline{T}^{\mathbf{A}}_{ik}(\pi_1);\underline{T}^{\mathbf{A}}_{kj}(\pi_2)) \\
\underline{T}^{\mathbf{A}}_{ij}(\pi_1 \cup \pi_2) & = & \underline{T}^{\mathbf{A}}_{ij}(\pi_1) \cup \underline{T}^{\mathbf{A}}_{ij}(\pi_2)) \\
\underline{T}^{\mathbf{A}}_{ij}(\pi^*) & = & K^{\mathbf{A}}_{ijn}(\pi)
\end{array}$$

where it is assumed that the action model $\mathbf{A}$ has n states, and the states are numbered $0, \ldots, n-1$. $K^{\mathbf{A}}_{ijn}$ is the Kleene path transformer, as in [9].

The proof system for PDL$^+$ consists of all axioms and rules of LCC except the reduction axiom:

$$[\mathbf{A}, e_i][\pi]\phi \leftrightarrow \bigwedge_{j=0}^{n-1} [T^{\mathbf{A}}_{ij}(\pi)][\mathbf{A}, e_j]\phi.$$

In addition, PDL$^+$ has the axioms for converse atomic programs as in section 2, and reduction axioms of the form:

$$[\mathbf{A}, e_i][\pi]\phi \leftrightarrow \bigwedge_{j=0}^{n-1} [\underline{T}^{\mathbf{A}}_{ij}(\pi)][\mathbf{A}, e_j]\phi.$$

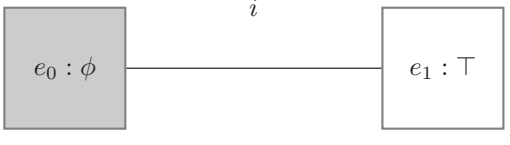
This is the patch we need to prove the completeness result (Theorem 2).

Appendix 2: Restricted Announcements Versus Restricted Belief Changes

A restricted announcement of ϕ is an announcement of ϕ that is not delivered to one of the agents i. Notation $!\phi^{-i}$. The action model for $!\phi^{-i}$ has event set $\{e_0, e_1\}$, with e_0 the actual event, where e_0 has precondition ϕ and e_1 precondition $\top$, and with the preference relation given by

$$P_i = \{(e_0, e_0), (e_1, e_1), (e_0, e_1), (e_1, e_0)\},$$

and $P_j = \{(e_0, e_0), (e_1, e_1)\}$ for all $j \neq i$.



A protocol for restricted announcements, for epistemic situation M, is a set of finite sequences of formula-agent pairs, such that each sequence

$$(\phi_0, i_0), \ldots, (\phi_n, i_n)$$

has the following property:

$$\forall k \in \mathbb{N} : 0 \leq k < n \rightarrow \exists i \in Ag : \mathbf{M}, w \models [!\phi_0^{-i_0}], \ldots, [!\phi_{k-1}^{-i_{k-1}}][\sim_i]\phi_k.$$

Intuitively, at every stage in the sequence of restricted announcements, some agent has to possess the required knowledge to make the next announcement in the sequence. We can now prove that such protocols can never establish common knowledge of purely propositional facts.

Theorem 4. *Let C express common knowledge among set of agents Ag. Let* $\mathbf{M}$ *be an epistemic model with actual world w such that* $\mathbf{M}, w \models \neg C\phi$*, with ϕ purely propositional. Then there is no protocol with*

$$\mathbf{M}, w \models [!\phi_0^{-i_0}], \ldots, [!\phi_n^{-i_n}]C\phi.$$

for any sequence $(\phi_0, i_0), \ldots, (\phi_n, i_n)$ in the protocol.

Proof. We show that $\neg C\phi$ is an invariant of any restricted announcement.

Assume $\mathbf{M}, w \models \neg C\phi$. Let $(\mathbf{A}, e)$ be an action model for announcement $!\psi^{-i}$, the announcement of ψ, restricted to $Ag - \{i\}$. Then $\mathbf{A}$ has events e and e', with $\mathbf{pre}(e) = \psi$ and $\mathbf{pre}(e') = \top$. If $\mathbf{M}, w \models \neg\psi$ then the update does not succeed, and there is nothing to prove. Suppose therefore that $\mathbf{M}, w \models \psi$. Since $\mathbf{pre}(e') = \top$, the model $\mathbf{M} \otimes \mathbf{A}$ restricted to domain $D = \{(w, e') \mid w \in W_{\mathbf{M}}\}$ is a copy of the original model $\mathbf{M}$. Thus, it follows from $\mathbf{M}, w \models \neg C\phi$ that

$$\mathbf{M} \otimes A \restriction D, (w, e') \models \neg C\phi.$$

Thus, there is an C-accessible world-event pair (w', e'') in D with

$$\mathbf{M} \otimes A \restriction D, (w', e'') \models \neg\phi.$$

Since ϕ is purely propositional, we get from this that:

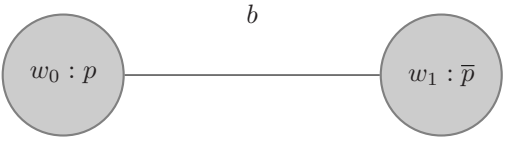
$$\mathbf{M} \otimes A, (w', e'') \models \neg\phi.$$

Observe that since common knowledge is preserved under model restriction, absence of common knowledge is preserved under model extension. The C-accessible world-event pair (w', e'') in $\mathbf{M} \otimes A \restriction D$ will still be C-accessible in $\mathbf{M} \otimes A$. Therefore, it follows that $\mathbf{M} \otimes A, (w, e') \models \neg C\phi$. By the construction of $\mathbf{M} \otimes A$, we get from this that $\mathbf{M} \otimes A, (w, e) \models \langle i\rangle\neg C\phi$, and therefore $\mathbf{M} \otimes A, (w, e) \models \neg C\phi$, by the definition of common knowledge.

It follows immediately that no protocol built from restricted announcements can create common knowledge of propositional facts.

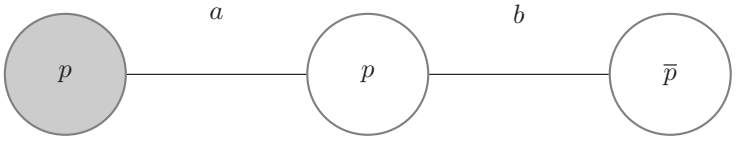
The case of the two generals planning a coordinated attack on the enemy, but failing to achieve common knowledge about it [16,17] can be viewed as a special case of this theorem.

If there are just two agents i, j, the only way for agent i to send a restricted message is by allowing uncertainty about the delivery. If i, j are the only agents, and i knows ϕ then the restricted message $!\phi^{-j}$ conveys no information, so the only reasonable restricted announcement of ϕ is $!\phi^{-i}$. The upshot of this announcement is that the message gets delivered to j, but i remains uncertain about this. According to the theorem, such messages cannot create common knowledge. Initial situation:

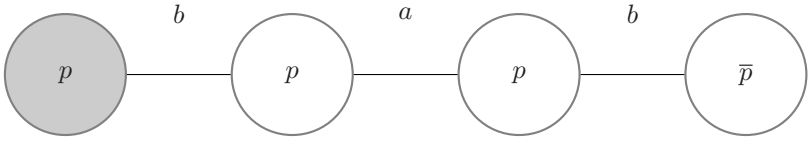


Update action for general a (left) and general b (right):

Situation after first message from general a:

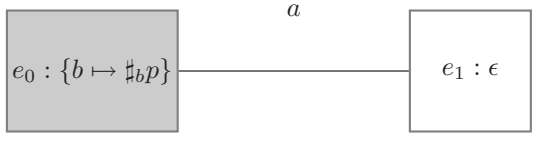


Situation after update by a followed by update by b:

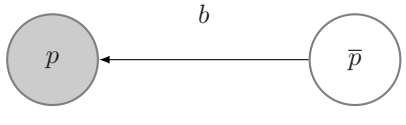


And so on ...

Now look at the case where restricted announcements are replaced by non-public belief revisions. Then the power of restricted belief change turns up in the following example. We start out from the initial situation again, and we update using the action model for non-public belief change:



Here is the update result (after minimalisation under bisimulation):



The example shows that it is possible to achieve common safe belief in p in a single step, by means of a non-public belief change.

Time Complexity and Convergence Analysis of Domain Theoretic Picard Method

Amin Farjudian and Michal Konečný*

Computer Science, Aston University
Aston Triangle, B4 7ET, Birmingham, UK
{a.farjudian,m.konecny}@aston.ac.uk

Abstract. We present an implementation of the domain-theoretic Picard method for solving initial value problems (IVPs) introduced by Edalat and Pattinson [1]. Compared to Edalat and Pattinson's implementation, our algorithm uses a more efficient arithmetic based on an arbitrary precision floating-point library. Despite the additional overestimations due to floating-point rounding, we obtain a similar bound on the convergence rate of the produced approximations. Moreover, our convergence analysis is detailed enough to allow a static optimisation in the growth of the precision used in successive Picard iterations. Such optimisation greatly improves the efficiency of the solving process. Although a similar optimisation could be performed dynamically without our analysis, a static one gives us a significant advantage: we are able to predict the time it will take the solver to obtain an approximation of a certain (arbitrarily high) quality.

1 Context

Edalat and Pattinson [1] have introduced a domain-theoretic interpretation of the Picard operator and implemented an initial value problem (IVP) solver based on this interpretation. Amongst its strong points is the property that not only it gives validated results, i. e. it gives an enclosure for the solution (assuming the IVP is Lipschitz) over the whole time range up to a certain point, but also it is *complete* in the sense that convergence is guaranteed. I.e., the enclosure can be improved to be arbitrarily close to the solution by repeating the Picard iteration step. Moreover, the distance from the solution is shrinking exponentially with the number of iterations.

Methods based on fixed precision floating-point interval arithmetic used commonly in validated solvers lack such convergence properties. Nevertheless, as the authors indicate, their method is not particularly efficient compared to time-step simulation methods (eg Euler or Runge-Kutta.[1]) This is caused mainly by the fact that each Picard iteration takes much longer than the previous one due to the doubling of the partition density and a fast increase in the size of the rational numbers that describe the enclosure.

We have addressed these problems, improving on [1] by:

– Using a more efficient arithmetic, i. e. arbitrary precision floating-point numbers instead of rationals, while obtaining very similar convergence results;

* Funded by EPSRC grant EP/C01037X/1.

[1] For a domain-theoretic account of such a method, see [2].

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 149–163, 2008.

- Identifying and decoupling various sources of approximation errors (i. e. Picard iteration, integration partition density, field truncation and rounding error, integration rounding error);
- Allowing a finer control over increases in aspects of approximation quality and associated increases in computation effort in subsequent iterations;
- Identifying a scheme to determine how the effort should be increasing in subsequent iterations so that it is close to optimal (this scheme is static—effort increases for all planned iterations are determined before performing the first iteration);
- Providing a fairly accurate prediction of how long each iteration of this optimised algorithm will take, parametrised only by the duration of basic arithmetic operations and several numerical properties of the IVP.

We first formalise the basic solver algorithm based on Picard iterations (Section 2), then conduct a detailed analysis of its convergence properties (Section 3), deduce the promised static effort increasing scheme (Section 4) and analyse its timing properties (Section 5). In Section 6 we compare our solver and results to some other available validated IVP solvers and outline our plans.

Remark 1. Due to lack of space, we have only included the proof of Theorem 1 in Appendix A. More details and proofs can be found in the extended version [3].

2 Implementation of Picard Iteration

Recall that an IVP is given by equations $y' = f(y)$, $y(0) = a_0$ with $a_0 \in \mathbb{R}^\eta$ and the *field* $f\colon \mathbb{R}^\eta \to \mathbb{R}^\eta$. If the field is Lipschitz, the IVP has a unique solution $y\colon \mathbb{R} \to \mathbb{R}^\eta$. A Domain-Theoretic Picard solver consists in constructing a sequence of improving enclosures by an approximate interval version of the Picard operator

$$y_{k+1}(t) = a_0 + \int_0^t f(y_k(x))\,\mathrm{d}x.$$

2.1 Approximating Reals and Functions

Before we can present the solver, we need to introduce a few concepts related to the underlying arithmetic. Firtly, we provide notation for interval arithmetic based on floating-point numbers that is used to implement exact real arithmetic in the style of iRRAM [4][2] and RealLib [5].

Definition 1 (floating-point number, granularity). *In our framework, a* floating-point number *is either one of the special values* $+0$, -0, $+\infty$, $-\infty$, *NaN*[3] *or a quadruple* $f = (s, g, e, m)$ *consisting of: sign* $s \in \{-1, +1\}$, *granularity* $g \in \mathbb{N}$, *exponent* $e \in \{-2^g, -2^g + 1, \ldots, 2^g - 1, 2^g\}$ *and mantissa* $m \in \{0, 1, \ldots, 2^g - 1\}$. *The intended value of* f *is* $[\![f]\!] := s \times (1 + \frac{m}{2^g}) \times 2^e$.

[2] `www.informatik.uni-trier.de/iRRAM/`

[3] Not a Number.

Note that the overall bit size of the representation is determined by the *granularity* g. Also the computation time taken by primitive arithmetic operations depends chiefly on g.

Definition 2 ($\mathbb{F}_g$, $\mathbb{F}$, $\mathbb{IF}_g, \mathbb{IF}$). *For each natural number $g \in \mathbb{N}$, we write the set of floating-point numbers with granularity $g \in \mathbb{N}$ as $\mathbb{F}_g$, and by $\mathbb{F}$ we mean the union $\bigcup_{g\in\mathbb{N}} \mathbb{F}_g$. Also let $(\mathbb{IF}_g, \sqsubseteq)$ be the* poset *of all the intervals with floating-point numbers of granularity g as their end-points ordered under the* superset *relation, ie:*

$$\mathbb{IF}_g := \{[x, y] \mid x, y \in \mathbb{F}_g\} \qquad \textit{and} \qquad \forall \alpha, \beta \in \mathbb{IF}_g \colon \alpha \sqsubseteq \beta \Leftrightarrow \beta \subseteq \alpha. \tag{1}$$

Let $\mathbb{IF} := \bigcup_{g\in\mathbb{N}} \mathbb{IF}_g$ be partially ordered under the analogous superset relation.

It is clear that $\mathbb{IF}_0 \subset \mathbb{IF}_1 \subset \cdots \subset \mathbb{IF} \subset \mathbb{IR}_\infty$ where $\mathbb{IR}_\infty$ denotes the interval Scott-domain over $\mathbb{R} \cup \{-\infty, +\infty\}$. Moreover, $\mathbb{IF}$ is a countable basis of $\mathbb{IR}_\infty$.

Assume that we need to compute a function $f \colon \mathbb{R}^\eta \to \mathbb{R}$. In our floating-point arithmetic we would represent such a function by a computable $\widehat{If} \colon \mathbb{N}_\perp \times \mathbb{IF}^\eta \to \mathbb{IF}$. The first parameter is an index of *accuracy*, which determines the *precision* of the computation process[4] and the granularity of the result. By fixing the accuracy to i, we get $\widehat{If}(i, \_) \colon \mathbb{IF}^\eta \to \mathbb{IF}$, which we simply write as $\widehat{If}_i$. We require that with increasing i, these functions converge to some $If \colon \mathbb{IR}_\infty^\eta \to \mathbb{IR}_\infty$, which has to be an extension of f.

Definition 3 (width of interval and function)
For any interval $\alpha = [\underline{\alpha}, \overline{\alpha}] \in \mathbb{IR}_\infty$, $w(\alpha) := \overline{\alpha} - \underline{\alpha}$ is the width *of α. We extend this to the width of a box $\boldsymbol{\alpha} = \left(\alpha_1, \ldots, \alpha_\eta\right) \in \mathbb{IR}_\infty^\eta$ by $w(\boldsymbol{\alpha}) := \max\{w(\alpha_i) \mid i \in \{1, \ldots, \eta\}\}$. Finally, the width of a function $If \colon \mathbb{IR}_\infty^\eta \to \mathbb{IR}_\infty^\eta$ is defined as $w(f) := \sup\{w(f(x)) \mid x \in \mathbb{R}_\infty^\eta\}$.*

Definition 4 (Interval Lipschitz, Hausdorff Lipschitz from Below (HLFB) [7])

1. *Assume that width is defined over sets of intervals A and B. The function $f \colon A \to B$ is* interval Lipschitz *with constant $\mathcal{L}_f$ if $\forall \alpha \in A \colon w(f(\alpha)) \leq \mathcal{L}_f \cdot w(\alpha)$*
2. *If posets A and B are endowed with distance functions $d_X \colon X \times X \to \mathbb{R}$, ($X \in \{A, B\}$), the monotone function $f \colon A \to B$ is* Hausdorff Lipschitz from Below *if and only if $\exists\, \mathcal{L}_f \in \mathbb{R}\ \forall \alpha, \beta \in A \colon \quad \alpha \sqsubseteq \beta \quad \Rightarrow \quad d_B(f(\alpha), f(\beta)) \leq \mathcal{L}_f \cdot d_A(\alpha, \beta)$*

Definition 5 (Uniformly Hausdorff Lipschitz from Below (UHLFB))
Assume that A and B have width defined over their elements. The function $f \colon A \to B$ is uniformly Hausdorff Lipschitz from Below *if*
$\exists\, \mathcal{L}_f, \mathcal{E}_f \in \mathbb{R} \colon 0 < \mathcal{L}_f, \mathcal{E}_f < \infty \ \&\ \forall \alpha \in A \colon w(f(\alpha)) \leq \mathcal{L}_f w(\alpha) + \mathcal{E}_f$

2.2 Overview of the Solver

Recall that we want to solve an IVP $y' = f(y)$, $y(0) = a_0$ for $y : \mathbb{R} \to \mathbb{R}^\eta$, ($\eta \geq 1$) over some time domain. In what follows, we focus on the case $\eta = 1$ and the time domain

[4] For our purposes, the term *accuracy* refers to the width of the computed interval while *precision*, on the other hand, is a measure of accuracy for the basic field operations over floating-point numbers. For more details, see [6].

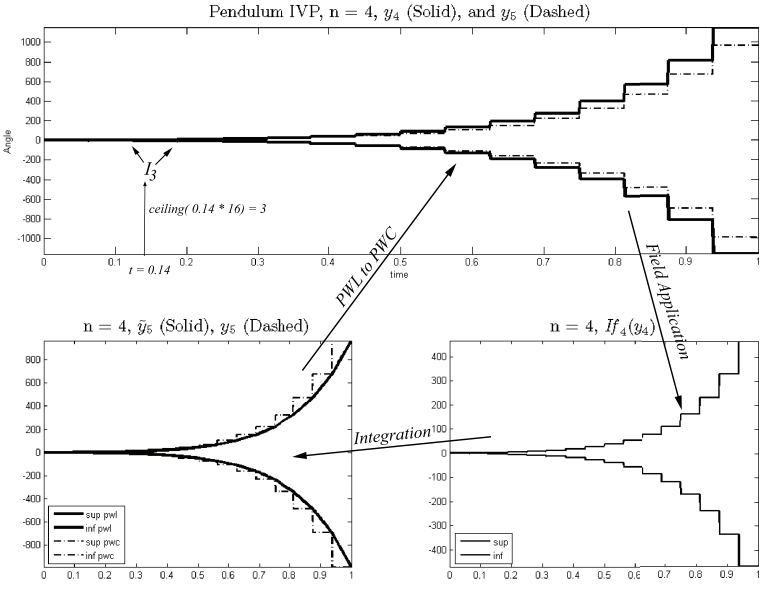


Fig. 1. Three main stages of the Picard algorithm

[0, 1]. Generalizing to any higher dimension or other time domain is straightforward. To implement the field $f : \mathbb{R} \to \mathbb{R}$, we pick a sequence $\mathcal{I}f_j$ satisfying $\sqcup\{\mathcal{I}f_j \mid j \in \mathbb{N}\} = \mathcal{I}f$ as discussed in subsection 2.1. Recall that accuracy index j determines the granularity of the calculation, which we shall denote by g_j. Thus we have $x \in \mathbb{IF}_{g_j} \implies \mathcal{I}f_j(x) \in \mathbb{IF}_{g_j}$. We also assume $\lim_{j\to\infty} g_j = \infty$.

Each solution enclosure y_k computed by the solver is a pair of piece-wise constant (PWC) fucntions with the domain [0, 1] partitioned into 2^n segments of the same size. We call the number n the *depth* of the partition. In each Picard step, the field is applied point-wise to the enclosure y_k, giving another PWC function. This function is integrated, which leads to a piece-wise linear (PWL) function $\tilde{y}_{k+1}$. Finally, a process of conservative flattening converts this PWL object into a PWC one, ready to go through the next step of the iteration (see Figure 1).[5]

More formally, the solver follows the algorithm shown in Figure 2 on the facing page. In the algorithm, PWLtoPWC refers to the operator that turns a piece-wise linear function into a piece-wise constant one (see Figure 1). The depths $\{n_j\}$ and iterations per depth $\{k_j\}$ are left unspecified at this stage except that we require $\lim_{j\to\infty} n_j = \infty$. In Section 4 we discuss how the increases in depth and granularity can be determined statically.

[5] We have not specified how one obtains the initial enclosure y_0. In our tests we were able to guess a uniform bound b over our time domain, i.e. we could set $y_0(t) = b$. In general, one can obtain y_0 using a validated time-step method with fairly large time step. For an invalid y_0, the enclosures will eventually become inconsistent.

$y_0^{n_0}$ = initial piece-wise constant enclosure of depth n_0
for $j = 0..\infty$ **loop**
  for $k = 1..k_j$ **loop**
    $\tilde{y}_k^{n_j}$ = integrate $\mathcal{I}f_j(y_{k-1}^{n_j})$
    $y_k^{n_j}$ = PWLtoPWC $(\tilde{y}_k^{n_j})$
  end loop
  $y_0^{n_{j+1}}$ = $y_{k_j}^{n}$ converted to depth n_{j+1}
end loop

Fig. 2. Implementation of Picard method

In most applications, we can determine bounds for the field's domain and range. Thus we will assume that there are numbers $N, M > 0$ such that

$$\forall j \in \mathbb{N}\colon \widehat{\mathcal{I}f_j}\colon \mathbb{IF} \cap [-N, N] \to \mathbb{IF} \cap [-M, M] \tag{2}$$

3 Convergence Analysis

In this section, we analyse the convergence properties of the inner loop of the algorithm in Figure 2. This will first and foremost confirm that the whole process converges but, more importantly, will offer an important tool for *static*, i. e. *a priori*, analysis of the convergence properties, allowing us to make informed choices for the increases in depth and granularity.

While we focus on the inner loop only, we can drop the subscript j and work with a fixed depth n and granularity g. Moreover, we drop the superscript and write y_k and $\tilde{y}_k$, instead of $y_k^{n_j}$ and $\tilde{y}_k^{n_j}$, respectively.

3.1 Convergence Analysis: No Round-Off Errors

Each PWC solution enclosure partitions the time domain into 2^n sub-intervals, which we denote $I_1, I_2, \ldots, I_{2^n}$. Thus, any point $t \in (0, 1]$ falls into the sub-interval with index $\lceil t2^n \rceil$, i. e. $t \in \mathcal{I}_{\lceil t2^n \rceil}$ (See Figure 1).[6]

According to the algorithm in Figure 2, we would expect $\forall t \in \mathcal{I}_p$:

$$\underline{\tilde{y}_k(t)} = \underline{a_0} + \sum_{m=1}^{p-1} w(\mathcal{I}_m)\underline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_m))} + (t - \underline{\mathcal{I}_p})\underline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_p))} \tag{3.a}$$

$$\overline{\tilde{y}_k(t)} = \overline{a_0} + \sum_{m=1}^{p-1} w(\mathcal{I}_m)\overline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_m))} + (t - \underline{\mathcal{I}_p})\overline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_p))} \tag{3.b}$$

Next, the algorithm traps the piece-wise linear result $\tilde{y}_k$ in a tight piece-wise constant approximation y_k using the following formulae:

[6] $\lceil x \rceil$ is the smallest integer greater than or equal to x, usually known as the *ceiling* of x.

$$\underline{y_k(t)} = \min\left\{\underline{\tilde{y}_k(\underline{\mathcal{I}_p})}, \underline{\tilde{y}_k(\overline{\mathcal{I}_p})}\right\} \tag{4.a}$$

$$\overline{y_k(t)} = \max\left\{\overline{\tilde{y}_k(\underline{\mathcal{I}_p})}, \overline{\tilde{y}_k(\overline{\mathcal{I}_p})}\right\} \tag{4.b}$$

Note that $\forall m \in \{1, \ldots, 2^n\} : w(\mathcal{I}_m) = 2^{-n}$. We also make the following assumptions before arriving at the final formula:

1. The field f is Lipschitz and its approximation $\mathcal{I}f_j$ is uniformly Hausdorff Lipschitz from Below. We fix some numbers Λ and $\mathcal{E}$ with the property declared in Definition 5:
$$\forall x \in \mathbb{IR}_\infty : w(\mathcal{I}f_j)(x) \leq \Lambda \cdot w(x) + \mathcal{E} \tag{5}$$
2. For a function h which is constant on an interval $x = [\underline{x}, \overline{x}]$, we abuse the notation and define $h(x) \coloneqq h(m(x))$, where $m(x) = (\underline{x} + \overline{x})/2$. As the solution is considered to be piece-wise constant over each interval $\mathcal{I}_m$, we write $y_k(\mathcal{I}_m)$ instead of $y_k(t)$, where $t \in \mathcal{I}_m$, accordingly.

Theorem 1 (Inner loop width improvement without round-off errors)
Under the above assumptions, we can bind the width of the solution enclosure as follows:

Case ($\Lambda > 0$) and ($\mathcal{E}/\Lambda < 1$)

$$w(y_k(\mathcal{I}_p)) \leq w(y_0)\left(\frac{\Lambda}{2^n}\right)^k \binom{p+k-1}{k} + \left(w(a_0) + \frac{M}{2^n} + \left(\frac{\mathcal{E}}{\Lambda}\right)\right)\mathrm{e}^r \tag{6.a}$$

Case ($\Lambda < \mathcal{E}$) including ($\Lambda = 0$)

$$w(y_k(\mathcal{I}_p)) \leq w(y_0)\left(\frac{\Lambda}{2^n}\right)^k \binom{p+k-1}{k} + \left(w(a_0) + \frac{M}{2^n} + \mathcal{E}\left(\frac{p}{2^n}\right)\right)\mathrm{e}^r \tag{6.b}$$

where in both cases $r = (p + k - 1)\Lambda/2^n$.

Proof See Appendix A.

Remark 2 Please note that the index $p = \lceil t2^n \rceil$ depends on *both* t and n.

Note that formulae (6.a) and (6.b) deal with the inner loop only, hence they rely on the parameter j. In particular, to discuss the outer loop, Λ, $\mathcal{E}$ and r should be thought of as Λ_j, $\mathcal{E}_j$ and r_j, respectively.

Theorem 1 gives an upper bound on the width of the result after one run of the outer loop. In what follows it will be demonstrated that even at this stage with appropriate choices of k and n one may get as narrow an interval as one wishes. However, as we will discuss in Section 4, a well worked out strategy would provide a list of these parameters for each run of the outer loop so that the convergence to the result is attained in a nearly *optimal* manner.

Proposition 1. *Assume that* $\forall j\colon \Lambda_j < \Lambda$ *for some* $\Lambda < \infty$[7] *and* $w(a_0) = 0$. *Then for any point* $t \in (0,1)$ *and* $\epsilon > 0$, *we can have* $w(y_k^n(t)) < \epsilon$ *with suitable choices of n, k and j.*

Proof (Sketch). Here we only consider the case $\Lambda > 0$, therefore formula (6.a) will be the appropriate one. Note that $\lim_{j\to\infty} \mathcal{E}_j = 0$, as we have $\sqcup\{\mathcal{I}f_j \mid j \in \mathbb{N}\} = \mathcal{I}f$. Considering that $w(a_0) = 0$, we rewrite the right hand side as follows:

$$w(y_0)\left(\frac{\Lambda^k}{k!}\right)\left(\frac{\prod_{\ell=1}^{k}(p+\ell)}{2^n}\right) + \left(\frac{M}{2^n} + \left(\frac{\mathcal{E}_j}{\Lambda}\right)\right)\mathrm{e}^{r_j}$$

Let us take $t \in (0,1)$ and assume some $\epsilon > 0$ is given. We first pick some k for which $\frac{(\Lambda+1)^k}{k!} < \frac{\epsilon}{w(y_0)}$. Then for this k we find j_0 and n_0 such that for all $j > j_0$ and $n > n_0$:

1. $n > \log_2(\max\{(\frac{k}{1-t}), \frac{4Me^{-(\Lambda_0+1)}}{\epsilon}\})$, which enforces $\frac{p+k-1}{2^n} < 1$, and $(\frac{M}{2^n})e^{r_j} < \frac{\epsilon}{4}$
2. $\mathcal{E}_j < \frac{\epsilon\Lambda e^{-(\Lambda_0+1)}}{4}$, whic enforces $\left(\frac{\mathcal{E}_j}{\Lambda}\right)\mathrm{e}^{r_j} < \frac{\epsilon}{4}$

Straightforward calculation shows that these conditions make (6.a) smaller than ϵ. □

So far we have assumed that the basic arithmetic operations have no *round-off errors*. In other words, if $\square$ is any of addition, subtraction, multiplication or division:

$$\forall \alpha, \beta \in \mathbb{IF}\colon\ \alpha \,\square\, \beta = \{t_1 \,\square\, t_2 \mid t_1 \in \alpha, t_2 \in \beta\} \tag{7}$$

where $\square$ on the left hand side is the "*interval version*" of the binary operator $\square$ on the right hand side. Our framework accommodates *imprecision*, the case of which will be analysed next.

3.2 Convergence Analysis: Round-Off Errors

We devise operation $\mathring{\square}$ in such a way that $\forall \alpha \in \mathbb{IF}_{g_1}, \beta \in \mathbb{IF}_{g_2}\colon \alpha \,\mathring{\square}\, \beta = [r,s]$ where $[r,s] \in \mathbb{IF}_{g_3}$ and $g_3 = \max\{g_1, g_2\}$. The operation $\mathring{\square}$ is seen as the *imprecise* counterpart of $\square$. Being imprecise does not mean giving up on soundness, i. e. $\forall t_1 \in \alpha, t_2 \in \beta\colon t_1 \,\square\, t_2 \in [r,s]$. In other words, instead of aiming for an exact result, we contend with an interval $[r,s]$ which encloses the result as tightly as possible without increasing the number of bits as dictated by the input arguments.

In order not to let this liberty take the better of the convergence, we postulate that for each natural number $g \in \mathbb{N}$, a bound $\epsilon_g > 0$ exists such that:

$$\forall \alpha \in \mathbb{IF}_{g_1}, \beta \in \mathbb{IF}_{g_2}\colon\ w(\alpha \,\mathring{\square}\, \beta) \le (1+\epsilon_g)\, w(\alpha \,\square\, \beta) \tag{8}$$

in which the operator $\mathring{\square}$ is the imprecise counterpart of $\square$ and $g = \max\{g_1, g_2\}$.

Definition 6 (Enclosure Constant C)
Let $f\colon \mathbb{IF}^\eta \to \mathbb{IF}$ *with* $\mathring{f}\colon \mathbb{IF}^\eta \to \mathbb{IF}$ *as its imprecise counterpart such that:*

$$\begin{cases} \forall \boldsymbol{\alpha} \in \mathbb{IF}g_1 \times \ldots \times \mathbb{IF}g_\eta\colon \mathring{f}(\boldsymbol{\alpha}) \in \mathbb{IF}g, \quad g = \max\{g_1, \ldots, g_\eta\} \\ \forall g \in \mathbb{N}\colon \exists \epsilon_{\mathring{f},g} > 0\colon \forall \boldsymbol{\alpha} \in \mathbb{IF}g_1 \times \ldots \times \mathbb{IF}g_\eta\colon \quad w(\mathring{f}(\boldsymbol{\alpha})) \le (1+\epsilon_{\mathring{f},g})\, w(f(\boldsymbol{\alpha})) \end{cases}$$

[7] This assumption holds in almost all cases of practical interest. In fact, we can just take this upper bound and assume $\forall j\colon \Lambda_j = \Lambda$.

We call $(1 + \epsilon_{\mathring{f},g})$ an enclosure constant of $\mathring{f}$ *at granularity* g, *and denote it by* $C(g, \mathring{f})$. *When* f *is unambiguously understood from the context we simply write* ϵ_g *and* C_g.

Again, we stipulate that:

1. There exists $\epsilon_g > 0$ such that for all basic arithmetic operators $\mathring{f}$, we have $\epsilon_{\mathring{f},g} < \epsilon_g$.
2. There exists $\delta_g > 0$ such that for all $\mathring{f} \in \{\widehat{\mathring{\mathcal{I}}f_i}\}_{i\in\mathbb{N}}$ the inequality $\epsilon_{\mathring{f},g} < \delta_g$ holds.
3. $\lim_{g\to\infty}(\epsilon_g + \delta_g) = 0$

In the analysis that follows, it is assumed that for some minimum $g_0 \in \mathbb{N}$, the whole computation is carried out with numbers having granularities at least as big as g_0. Therefore, knowing that g_0 may be increased as needed to reduce the effect of round-off error, we define:

Definition 7 (*C*, *D*). *For the minimum granularity* g_0 *used during one run of the main algorithm, we define* $C = 1 + \epsilon_{g_0}$ *and* $D = 1 + \delta_{g_0}$.

Once again, the convergence analysis is split into two cases, similar to those in Theorem 1.

Theorem 2 (Inner loop width improvement including round-off errors)
Under the assumptions of Theorem 1 but taking account of rounding, we have:

Case ($\Lambda > 0$) and ($\mathcal{E}/\Lambda < 1$)

$$w(\mathring{y}_k(\mathcal{I}_p)) \le w(\mathring{y}_0)\left(DC^{p+4}\frac{\Lambda}{2^n}\right)^k\binom{p+k-1}{k} + \left(C^{p+5}w(a_0) + C^3\frac{M}{2^n} + C^{p+4}D\left(\frac{\mathcal{E}}{\Lambda}\right)\right)\mathrm{e}^r \quad (9.a)$$

Case ($\Lambda < \mathcal{E}$) including ($\Lambda = 0$)

$$w(\mathring{y}_k(\mathcal{I}_p)) \le w(\mathring{y}_0)\left(DC^{p+4}\frac{\Lambda}{2^n}\right)^k\binom{p+k-1}{k} + \left(C^{p+5}w(a_0) + C^3\frac{M}{2^n} + C^{p+4}D\mathcal{E}\left(\frac{p}{2^n}\right)\right)\mathrm{e}^r \quad (9.b)$$

where in both cases $r = (p + k - 1)\left(DC^{p+4}\frac{\Lambda}{2^n}\right)$.

Proof See the extended version [3].

4 Optimising the IVP Solver

Theorem 2 is used to great effect as we finalise the solver algorithm, trying to finetune the sequences $\{n_j\}$ $\{k_j\}$ and $\{g_j\}$ and thus distributing the computational effort between integration depth and floating-point granularity.

Firstly, we set $n_j = j$ without loss of generality since k_j can be 0 for certain j's. Now k_j will indicate after how many Picard steps the depth should be increased by one. The numbers g_j indicate whether or not to raise granularity and by how much whenever depth is increased.

To determine $\{k_j\}$ and $\{g_j\}$ one could use a modified algorithm that differs from the original one in that it terminates the inner loop only when the width improvements fall below some threshold. Also, whenever the increase in depth does not lead to a significant width improvement, the algorithm will backtrack and restart the inner loop with an increased granularity. If even increasing both the depth and granularity does not lead to a significant improvement, the algorithm will try various combinations of increases in depth and granularity in independent executions until it finds a combination that produces an improvement above the set threshold.

The problem with the above algorithm is that it is very inefficient and its run-time is rather unpredictable. We execute this algorithm but replace a genuine Picard step with a simulation based on the appropriate formula from Theorem 2. To make this possible, we also have to replace solution enclosures with upper bounds on the width of the enclosures at the furthest time point of interest.

This simulation is very fast as the formula takes *linear* time with respect to the number of iterations k to evaluate.[8] Thus we can say that the simulation provides a viable static prediction of the convergence behaviour and a method to *a priori* determine finite sequences $\{k_j\}$, $\{g_j\}$ that are guaranteed to produce an enclosure with a desired precision using a nearly optimal computation effort.

5 Prediction of Computation Time

In Section 3, we studied the overall rate at which the shrinking intervals generated by the main algorithm converge to the solution. Here in this section we try to grasp the number of operations needed to run the algorithm with certain input arguments. To this end, we ought to put forward one acceptable way of breaking up the procedure.

Theorem 3 (complexity function ν). *Assume that for any required binary operator $\square\colon \mathbb{F}^2 \to \mathbb{F}$, the complexity function $\nu_\square\colon \mathbb{N} \to \mathbb{N}$ gives an upper bound for the number of computational steps carried out to calculate $x \square y$.*

Then the function $\nu\colon \mathbb{N}^4 \to \mathbb{N}$ defined by

$$\forall k \in \mathbb{N} \wedge p \in \{0, \ldots, 2^n\}: \quad \nu(k, p, n, g) = \vartheta \sum_{j=1}^{k} \binom{p + j - 1}{j} \tag{10}$$

where

$$\vartheta := 2\,(\nu_{+}(g) + \nu_{\cdot}(g)) + \nu_{\widehat{If_i}}(g) + \nu_{\min}(g) + \nu_{\max}(g)$$

gives an upper bound on the number of computation steps needed to compute the value of the solution of the IVP over the sub-interval with index p at iteration k and integration depth n with granularity g.

[8] ... a time negligible compared to that of Picard iterations, which according to formula (10) below is exponential in k.

Whenever g and n are clear from the context, we just write $\nu(k,p)$ instead of $\nu(k,p,n,g)$.

Proof (Main idea)

$$\forall k \in \mathbb{N} \wedge p \in \{0,\ldots,2^n\} : \begin{cases} \nu(k+1,p+1) = \nu(k+1,p) + \nu(k,p+1) + \vartheta \\ \nu(k,0) = \nu(0,p) = 0 \end{cases} \tag{11}$$

□

6 Conclusion

This work has mainly been built on the work of Edalat and Pattinson [1]. However, works on validated solutions for initial value problems abound. Notable examples in our view are ValEncIA [8], VNODE [9] and COSY [10].

The main difference between our work and that of Edalat and Pattinson's [1] on the one hand, and the aforementioned on the other is the fact that theirs are not only based on fixed-point floating-point arithmetic, but also the emphasis is mostly on practical use. We believe that our implementation enjoys the positive points of all other works in that not only it lends itself well to analysis while sitting nicely in a domain theoretic model, but also it avoids the worst causes of inefficiency encountered by methods similar to Edalat and Pattinson's [1].

Yet another important motivation for us lies in what we believe is the first major consideration of *parallel* validated IVP solving, for which we have found the Picard method most suitable. The convergence and time-complexity analysis as presented here is readily extended to the parallel scenario, where each decision to split a domain and assign the computation task to different processors for each time sub-domain can be guided prior to the actual computation process occurring.

Notwithstanding the fact that we believe the framework has vast potentials for development, there are specific tangible choices for future directions:

1. In more immediate future, experiments must be carried out to show how closely the bounds from Theorems 1 and 2 reflect real scenarios arisen from various classes of IVPs. Our experiments so far have given promising results but they are not sufficiently extensive to draw reliable conclusions.
2. The approximation and representation of functions can take various forms. Most notably, piece-wise linear or piecewise polynomial representations usually give much better convergence when incorporated in our method of IVP solving. Nevertheless, extending our results to these representations seems to need considerable effort due to the substantial increase in complexity.

References

1. Edalat, A., Pattinson, D.: A domain theoretic account of Picard's theorem. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 494–505. Springer, Heidelberg (2004)
2. Edalat, A., Pattinson, D.: A domain theoretic account of Euler's method for solving initial value problems. In: Dongarra, J., Madsen, K., Waśniewski, J. (eds.) PARA 2004. LNCS, vol. 3732, pp. 112–121. Springer, Heidelberg (2006)

3. Farjudian, A., Konečný, M.: Time complexity and convergence analysis of domain theoretic picard method (March 2008), http://www-users.aston.ac.uk/~farjudia/AuxFiles/2008-Picard.pdf
4. Müller, N.T.: The iRRAM: Exact arithmetic in C++. In: Blank, J., Brattka, V., Hertling, P. (eds.) CCA 2000. LNCS, vol. 2064, pp. 222–252. Springer, Heidelberg (2001)
5. Lambov, B.: Reallib: An efficient implementation of exact real arithmetic. Mathematical Structures in Computer Science 17(1), 81–98 (2007)
6. Higham, N.J.: Accuracy and Stability of Numerical Algorithms, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia (2002)
7. Edalat, A., Pattinson, D.: Domain theoretic solutions of initial value problems for unbounded vector fields. In: Escardó, M. (ed.) Proc. MFPS XXI. Electr. Notes in Theoret. Comp. Sci, vol. 155, pp. 565–581 (2005)
8. Rauh, A., Hofer, E.P., Auer, E.: Valencia-ivp: A comparison with other initial value problem solvers. In: CD-Proc. of the 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN 2006, Duisburg, Germany. IEEE Computer Society, Los Alamitos (2007)
9. Nedialkov, N.S.: Vnode-lp: A validated solver for initial value problems in ordinary differential equations. Technical Report CAS-06-06-NN, Department of Computing and Software, McMaster University (July 2006)
10. Makino, K., Berz, M.: Cosy infinity version 9. Nuclear Instruments and Methods A558, 346–350 (2005)

A Proof of Theorem 1

Based on formulae (3.a–3.b) and (4.a–4.b) on pages 153–154, one can get an estimate on the width of the piece-wise constant approximation to the solution:

$$
\begin{aligned}
w(y_k(t)) &= |\,\overline{y_k(t)} - \underline{y_k(t)}\,| \\
\text{(Subtracting (3.a) from (3.b))} &\le w(a_0) + \\
&\quad \sum_{m=1}^{p} w(\mathcal{I}_m)\left(\overline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_m))} - \underline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_m))}\right] \\
\text{(Accommodating (4.a) and (4.b))} &\quad + w(\mathcal{I}_p)\min\left\{|\,\overline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_p))}\,|, |\,\underline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_p))}\,|\right\} \\
&\le w(a_0) + \\
&\quad \sum_{m=1}^{p} w(\mathcal{I}_m)\left(\overline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_m))} - \underline{\mathcal{I}f_j(y_{k-1}(\mathcal{I}_m))}\right] \\
\text{(Using (2) on page 153)} &\quad + w(\mathcal{I}_p)M
\end{aligned}
$$

We can safely consider each $\mathcal{I}f_j$ to be uniformly Hausdorff Lipschitz from Below (Definition 5 on page 151). Thus, we get:

$$
\begin{aligned}
w(y_k(\mathcal{I}_p)) &\le w(a_0) + w(\mathcal{I}_p)M \\
&\quad \sum_{m=1}^{p} w(\mathcal{I}_m)\left(\mathcal{L}_{\mathcal{I}f_j} w(y_{k-1}(\mathcal{I}_m)) + \mathcal{E}_{\mathcal{I}f_j}\right) \qquad (12)
\end{aligned}
$$

According to assumption (5) on page 154 and as we have $\forall m \in \{1, \ldots, 2^n\} : w(\mathcal{I}_m) = 2^{-n}$, an easier to handle bound on the term (12) on the preceding page would be:

$$\sum_{m=1}^{p} w(\mathcal{I}_m)\Big(\mathfrak{L}_{If_j} w(y_{k-1}(\mathcal{I}_m)) + \mathcal{E}_{If_j}\Big) \leq \frac{\Lambda}{2^n} \sum_{m=1}^{p} w(y_{k-1}(\mathcal{I}_m)) + \frac{p}{2^n}\mathcal{E}$$

Therefore, by defining

$$\Upsilon_\alpha \coloneqq \frac{\alpha}{2^n}\mathcal{E} + w(a_0) + \frac{M}{2^n} \tag{13}$$

one can derive:

$$\begin{aligned}
w(y_k(\mathcal{I}_p)) &\leq \frac{\Lambda}{2^n} \sum_{m=1}^{p} w(y_{k-1}(\mathcal{I}_m)) + \Upsilon_p \\
(\textit{induction on } k) &\leq \frac{\Lambda}{2^n} \sum_{m_1=1}^{p} \left(\left(\frac{\Lambda}{2^n} \sum_{m_2=1}^{m_1} w(y_{k-2}(\mathcal{I}_{m_2})) \right) + \Upsilon_{m_1} \right) + \Upsilon_p \\
&= \left(\frac{\Lambda}{2^n}\right)^2 \sum_{m_1=1}^{p} \sum_{m_2=1}^{m_1} w(y_{k-2}(\mathcal{I}_{m_2})) + \\
&\quad \left(\left(\frac{\Lambda}{2^n}\right) \sum_{m_1=1}^{p} \Upsilon_{m_1} \right) + \Upsilon_p \\
&= \left(\frac{\Lambda}{2^n}\right)^3 \sum_{m_1=1}^{p} \sum_{m_2=1}^{m_1} \sum_{m_3=1}^{m_2} w(y_{k-3}(\mathcal{I}_{m_3})) + \\
&\quad \left(\frac{\Lambda}{2^n}\right)^2 \sum_{m_1=1}^{p} \sum_{m_2=1}^{m_1} \Upsilon_{m_2} + \\
&\quad \left(\frac{\Lambda}{2^n}\right) \sum_{m_1=1}^{p} \Upsilon_{m_1} + \\
&\quad \Upsilon_p \\
&\vdots \\
(\text{Expanding (13)}) &\leq w(y_0)\left(\frac{\Lambda}{2^n}\right)^k \sum_{m_1=1}^{p} \sum_{m_2=1}^{m_1} \cdots \sum_{m_k=1}^{m_{k-1}} 1 + && \text{(14.a)} \\
&\quad \left(\frac{\mathcal{E}}{2^n}\right) \sum_{\ell=0}^{k-1} \left(\left(\frac{\Lambda}{2^n}\right)^\ell \sum_{m_1=1}^{p} \sum_{m_2=1}^{m_1} \cdots \sum_{m_{\ell+1}=1}^{m_\ell} 1 \right) + && \text{(14.b)} \\
&\quad \left(w(a_0) + \frac{M}{2^n}\right) \sum_{\ell=0}^{k-1} \left(\left(\frac{\Lambda}{2^n}\right)^\ell \sum_{m_1=1}^{p} \sum_{m_2=1}^{m_1} \cdots \sum_{m_\ell=1}^{m_{\ell-1}} 1 \right) && \text{(14.c)}
\end{aligned}$$

Note that $w(y_0)$ in term (14.a) on the preceding page is the width of the interval function y_0 — the initial estimate — as defined in subsection 2.1.

Now, in order to be able to carry the derivation forward, we state:

Lemma 1 *For any* $p, k \in \mathbb{N} \setminus \{0\}$*:*

$$\begin{aligned}\sum_{m_1=1}^{p}\sum_{m_2=1}^{m_1}\cdots\sum_{m_k=1}^{m_k} 1 \quad &\leq \quad \frac{p(p+1)\ldots(p+k-1)}{k!} \\ &= \binom{p+k-1}{k} \\ &= \binom{p+k-1}{p-1}\end{aligned}$$

Proof Left to the reader. □

Lemma 2 *Consider the real number* $\Lambda \geq 0$ *together with the natural numbers* $p, k, n \geq 1$*, then:*

$$\sum_{j=0}^{k}\binom{p+j-1}{j}\left(\frac{\Lambda}{2^n}\right)^j \leq \mathrm{e}^r$$

where

$$r = \left(\frac{\Lambda(p+k-1)}{2^n}\right) \tag{15}$$

Proof

$$\begin{aligned}\sum_{j=0}^{k}\binom{p+j-1}{j}\left(\frac{\Lambda}{2^n}\right)^j &= 1 + \sum_{j=1}^{k}\frac{p(p+1)\ldots(p+j-1)}{j!}\left(\frac{\Lambda}{2^n}\right)^j \\ &\leq \sum_{j=0}^{k}\frac{r^j}{j!} \leq \sum_{j=0}^{\infty}\frac{r^j}{j!} \leq \mathrm{e}^r\end{aligned}$$

□

Using lemmata 1 and 2 it is easier to get bounds on terms (14.a), (14.b) and (14.c) on the facing page. For term (14.a) one gets:

$$w(y_0)\left(\frac{\Lambda}{2^n}\right)^k\sum_{m_1=1}^{p}\sum_{m_2=1}^{m_1}\cdots\sum_{m_k=1}^{m_{k-1}} 1 \leq w(y_0)\left(\frac{\Lambda}{2^n}\right)^k\frac{p(p+1)\ldots(p+k-1)}{k!} \tag{16.a}$$

while term (14.c) can be bounded by:

$$\left(w(a_0)+\frac{M}{2^n}\right)\sum_{\ell=0}^{k-1}\left(\left(\frac{\Lambda}{2^n}\right)^\ell\sum_{m_1=1}^{p}\sum_{m_2=1}^{m_1}\cdots\sum_{m_\ell=1}^{m_{\ell-1}} 1\right) \leq \left(w(a_0)+\frac{M}{2^n}\right)\mathrm{e}^r \tag{16.b}$$

using Lemma 2 with r as in 15. To get a bound on term (14.b), let us first consider

$$T = \sum_{\ell=0}^{k-1}\left(\left(\frac{\Lambda}{2^n}\right)^{\ell}\sum_{m_1=1}^{p}\sum_{m_2=1}^{m_1}\cdots\sum_{m_{\ell+1}=1}^{m_\ell}1\right)$$

We develop two bounds which cover all foreseeable cases:

($\Lambda > 0$) and ($\mathcal{E}/\Lambda < 1$) : In this case, we go down the following route and consider:

$$\left(\frac{\Lambda}{2^n}\right)T = \sum_{\ell=1}^{k}\left(\frac{\Lambda}{2^n}\right)^{\ell}\binom{p+\ell-1}{\ell}$$
$$\text{(Lemma 2)} \leq \mathrm{e}^r$$

where again r is as in 15. Thus $T \leq \mathrm{e}^r 2^n/\Lambda$ and as term (14.b) is just $\mathcal{E}T/2^n$ the bound is

$$\left(\frac{\mathcal{E}}{2^n}\right)\sum_{\ell=0}^{k-1}\left(\left(\frac{\Lambda}{2^n}\right)^{\ell}\sum_{m_1=1}^{p}\sum_{m_2=1}^{m_1}\cdots\sum_{m_{\ell+1}=1}^{m_\ell}1\right) \leq \left(\frac{\mathcal{E}}{\Lambda}\right)\mathrm{e}^r \tag{16.c}$$

By combining (16.a), (16.b) and (16.c) we arrive at the first version of the bound:

$$w(y_k(\mathcal{I}_p)) \leq w(y_0)\left(\frac{\Lambda}{2^n}\right)^{k}\binom{p+k-1}{k} + \left(w(a_0) + \frac{M}{2^n} + \left(\frac{\mathcal{E}}{\Lambda}\right)\right)\mathrm{e}^r \tag{16.d}$$

($\Lambda < \mathcal{E}$) including the case ($\Lambda = 0$) : In this case we cannot simply multiply and divide by terms having Λ as a factor in their numerator. Instead, we factorize T in another way. First we consider the simple fact that for any $p \geq 1$:

$$\forall \ell \in \{1,\ldots,k\} : \ \frac{p+\ell-1}{\ell} \leq p \tag{†}$$

Therefore:

$$T = \sum_{\ell=1}^{k}\left(\frac{\Lambda}{2^n}\right)^{\ell-1}\binom{p+\ell-1}{\ell}$$
$$\text{(expanding the binomial term)} = \sum_{\ell=1}^{k}\left(\frac{\Lambda}{2^n}\right)^{\ell-1}\left(\frac{p(p+1)\ldots(p+\ell-1)}{\ell!}\right)$$
$$\text{(using (†) above)} \leq p\sum_{\ell=1}^{k}\left(\frac{\Lambda}{2^n}\right)^{\ell-1}\left(\frac{p(p+1)\ldots(p+\ell-2)}{(\ell-1)!}\right)$$
$$\text{(substituting } j \text{ for } \ell-1) = p\sum_{j=0}^{k-1}\left(\frac{\Lambda}{2^n}\right)^{j}\left(\frac{p(p+1)\ldots(p+j-1)}{j!}\right)$$
$$\text{(Lemma 2 on the preceding page)} \leq p\mathrm{e}^r$$

which implies that in this case, the bound on term (14.b) is:

$$\left(\frac{\mathcal{E}}{2^n}\right)\sum_{\ell=0}^{k-1}\left(\left(\frac{\Lambda}{2^n}\right)^{\ell}\sum_{m_1=1}^{p}\sum_{m_2=1}^{m_1}\cdots\sum_{m_{\ell+1}=1}^{m_\ell}1\right)\leq\mathcal{E}\left(\frac{p}{2^n}\right)\mathrm{e}^r \tag{16.e}$$

Thus, combining (16.a), (16.b) and (16.e) will result in the second version of the bound (16.d) on the facing page:

$$w(y_k(\mathcal{I}_p))\leq w(y_0)\left(\frac{\Lambda}{2^n}\right)^k\binom{p+k-1}{k}+\left(w(a_0)+\frac{M}{2^n}+\mathcal{E}\left(\frac{p}{2^n}\right)\right)\mathrm{e}^r$$

On the Formal Semantics of IF-Like Logics

Santiago Figueira[1,3], Daniel Gorín[1], and Rafael Grimson[2]

[1] Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina
[2] Departamento de Matemática, FCEyN, Universidad de Buenos Aires, Argentina, Hasselt University and Transnational University of Limburg
[3] CONICET, Argentina

Abstract. In classical logics, the meaning of a formula is invariant with respect to the renaming of bound variables. This property, normally taken for granted, has been shown not to hold in the case of Information Friendly (IF) logics. In this work we propose an alternative formalization under which invariance with respect the renaming of bound variables is restored. We show that, when one restricts to formulas where each variable is bound only once, our semantics coincide with those previously used in the literature. We also prove basic metatheoretical results of the resulting logic, such as compositionality and truth preserving operations on valuations. We work on Hodges' *slash logic* (from which results can be easily transferred to other IF-like logics) and we also consider his flattening operator, for which we give a game-theoretical semantics.

1 Introduction

Independence Friendly logic (IF, for short) was introduced and promoted as a new foundation for mathematics by Jaako Hintikka over a decade ago [9,10]. Closely related to Henkin's logic of *branching quantifiers* [8,16,7,2], IF is an extension of first-order logic where disjunctions and existential quantifiers may be decorated with denotations of universally-quantified variables. The intended meaning of a formula $\forall x \exists y /_{\forall x} \varphi$ is that the value for y may not *depend* on x (in other words, it may not be function of x). This notion is nicely formalized using a two player game between Abélard and Eloïse, which, because of the independence restrictions, is of *imperfect information*.

It was conjectured by Hintikka that one could not formulate IF semantics in a *composable* way [9]. This was promptly rebutted by Hodges in [11], where he achieves compositionality by taking as the interpretation of a formula $\varphi(x_1, \ldots, x_n)$ over the domain A, the set of sets of n-tuples (called *trumps*) for which Eloïse has a uniform winning strategy.

Two things are worth observing. First, in [11] Hodges introduced two slight modifications in syntax and semantics, namely: conjunctions and universal quantifiers may also be decorated with restrictions, and restrictions on any of the player's choices may range also over any of his previous choices[1]. Hodges later

[1] In Hintikka's presentation [9], Eloïse is not allowed to take into account her previous choices. For implications of this fact see, e.g. [14].

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 164–178, 2008.

coined the name *slash logic* for his formulation and noticed that many writers have transferred the name 'IF logic' to slash logic, often without realising the difference [12]. We will use the term IF-*like logics* to encompass this variety of related logics. In [13], Hodges shows that even if one restricts to Hintikka and Sandu's original formulation of IF, compositionality can be obtained. The second thing to note is that in both papers Hodges considers only the syntactic fragment where each variable may be bound only once. The underlying assumption is that, given an arbitrary formula, one can appropriately rename its variables, so no generality is loss. In the light of later findings, it is not obvious whether this is was a reasonable assumption.

Caicedo and Krynicki [4] proved a prenex normal form theorem for slash logic. To account for arbitrary formulas, where variables occur in any order, and may get rebound, they used compositional semantics in the line of Hodges, but with n-tuples replaced with valuations. This extension seemed so natural that in later papers it was taken as the standard semantics of slash logic.

Based on this formulation, in [14], Janssen pointed out several strange properties of these logics. At the root of them lies the idea of *signaling*, i.e., "the phenomenon that the value of a variable one is supposed not to know, is available through the value of another variable" [15]. He observes that if variables are reused, signaling may be blocked and, thus, the truth-value of formulas that only differ on bound-variables may differ. This can even be the case of formulas of IF-logic without restrictions, which would challenge Hintikka's claim of IF being a *conservative extension* of classical logic [9].

A systematic analysis of signaling in IF-like logics was later performed in [15], where several claims of "equivalence of formulas under syntactic transformations" made in [4] are questioned due to signalings that may get unexpectedly blocked. These results were fixed in [3] by restating them in a much weaker sense.

Summing up, on the one hand, we have a family of logics, aiming to be a conservative extension of first-order logic, for which several results have been proved, but that hold only for the regular fragment. On the other hand, we have that the attempts to formulate general results for the whole fragment failed. In the face of this, Dechesne advocated for the restriction of IF-like logics to the *regular fragment*, where no rebinding of variables occur (cf. Section 7.5 of [6]).

In this paper, we argue that there is no real need to restrict IF-like logics to regular formulas and that, in fact, most, if not all, of previous results can be generalized to the irregular case in a safe and natural way. In a nutshell, we claim that classical valuations are simply not adequate to formalize independence restrictions in a context where variables can get rebound.

In Section 2 we discuss briefly the interaction between irregular formulas and classical valuations with respect to signaling. This motivates Section 3, where we avoid these problems using alternative semantics, that are equivalent for the regular fragment. In Section 4 we consider also the *flattening* operator $\downarrow$, introduced by Hodges in [11] and illustrate that irregular formulas can be handled uniformly also in this setting; while doing this, we provide a new (to the best of our knowledge) game semantics for this logic. All the proofs are in Appendix A.

2 Preliminaries

2.1 Syntax

From here on, we restrict ourselves to Hodges' *slash logic* (but without indexed disjunctions) [11,12], in which Hintikka's IF logic can be trivially embedded. Formulas are built out of an infinite supply of constant symbols, function symbols and relation symbols just like in first-order logic, using the following set of connectives: $\sim$, $\vee/_{y_1,\ldots,y_k}$ and $\exists x/_{y_1,\ldots,y_k}$, where $y_1,\ldots,y_k$ stands for a set of variables. The derived connectives $\wedge/_{y_1,\ldots,y_k}$ and $\forall x/_{y_1,\ldots,y_k}$ are defined in the usual way. We will also write $\wedge$, $\vee$, $\exists x$ and $\forall x$ for $\wedge/_{\emptyset}$, $\vee/_{\emptyset}$, $\exists x/_{\emptyset}$ and $\forall x/_{\emptyset}$. Following [4] we don't impose any restriction on the variables occurring under the slashes.

The sets of free and bound variables of φ, $\mathrm{Fv}(\varphi)$ and $\mathrm{Bv}(\varphi)$ respectively, are defined in the usual way. Of course, variables that occur under slashes must be taken into consideration; observe, for example, that if $\theta := \exists x/_{x,y}[x = z]$ then $\mathrm{Fv}(\theta) = \{x, y, z\}$ and $\mathrm{Bv}(\theta) = \{x\}$.

Following Dechesne [6], we will say that a formula φ is *regular* whenever $\mathrm{Fv}(\varphi) \cap \mathrm{Bv}(\varphi) = \emptyset$ and there is no nested quantification over the same variable. To follow Hodges' presentation, when referring to regular formulas we will sometimes make the *context* (i.e. the free variables in scope) a parameter of the formula by writing: $\varphi(x_1,\ldots,x_n)$, where $(x_1,\ldots,x_n)$ is an n-tuple of distinct variables such that $\mathrm{Fv}(\varphi) \subseteq \{x_1,\ldots,x_n\}$. Observe that this means that for a fixed φ, $\varphi(x,y)$ and $\varphi(x,y,z)$ will generally denote two non-equivalent formulas. See [11] for further details.

2.2 Semantics

We will consider two related semantics. On the one hand, there is Hodges' trump semantics, which we will call *T-semantics*. It is compositional and based on sets of tuples but its formalization requires regular formulas with the context as a parameter. On the other, we have Caicedo and Kynicki's extension of trump semantics to arbitrary formulas, which we will call *V-semantics*. It is based on sets of valuations and has a natural game-based formulation from which compositionality can be proved [4,3].

Let us begin with V-semantics. A formula φ is true in a model $\mathcal{M}$ under a set of valuations V, written $\mathcal{M} \models^{+} \varphi[V]$, iff Eloïse has a valid strategy that, when followed, wins every instance $\mathsf{G}(\mathcal{M},\varphi,v)$ (for $v \in V$) of the classical satisfaction game between Abélard and Eloïse. Dually, a formula is false, written $\mathcal{M} \models^{-} \varphi[V]$, whenever Abélard has a valid strategy that is winning for every $\mathsf{G}(\mathcal{M},\varphi,v)$, $v \in V$. For a strategy to be valid, it has to satisfy additional independence conditions. For a formal presentation refer to [4,3].

Hodges avoided valuations in the first place by restricting to regular formulas where the context is a parameter: a valuation for $\varphi(x_1,\ldots,x_n)$ is simply an n-tuple $(a_1,\ldots,a_n)$. Let us say that $v_{(a_1,\ldots,a_n)}$ is a valuation such that $v_{(a_1,\ldots,a_n)}(x_i) = a_i$ when $1 \leq i \leq n$ and $v(x) = c$, for some fixed c, otherwise; then, intuitively, a *trump* (resp. *cotrump*) T for $\varphi(x_1,\ldots,x_n)$ in $\mathcal{M}$, written

$\mathcal{M} \models^+ \varphi(x_1, \ldots, x_n)[T]$ (resp. $\mathcal{M} \models^- \varphi(x_1 \ldots x_n)[T]$), is just a set of n-tuples for which Eloïse (resp. Abélard) has a strategy that is winning for every instance of the game $\mathsf{G}(\mathcal{M}, \varphi, v_{(a_1 \ldots, a_n)})$ for $(a_1 \ldots a_n) \in T$. This can be alternatively defined in a composable way; we include for reference such a formulation in Appendix A and refer the reader to [11] for further details.

Notation. Thoughout this paper, "$\mathcal{M} \models^{\pm} X$ iff $\mathcal{M} \models^{\pm} Y$" will stand for "$\mathcal{M} \models^+ X$ iff $\mathcal{M} \models^+ Y$, and $\mathcal{M} \models^- X$ iff $\mathcal{M} \models^- Y$".

As usual, each of these semantics gives rise to a notion of formula equivalence.

V-equivalence: $\varphi_1 \equiv_V \varphi_2$ iff $\mathrm{Fv}(\varphi_1) = \mathrm{Fv}(\varphi_2)$ and for all $\mathcal{M}$ and every set of valuations V, $\mathcal{M} \models^{\pm} \varphi_1[V]$ iff $\mathcal{M} \models^{\pm} \varphi_2[V]$.

T-equivalence: Let $\overline{x} = x_1, \ldots, x_n$. $\varphi_1(\overline{x}) \equiv_T \varphi_2(\overline{x})$ iff $\mathrm{Fv}(\varphi_1) = \mathrm{Fv}(\varphi_2)$ and for all $\mathcal{M}$ and every $T \subseteq |\mathcal{M}|^n$, $\mathcal{M} \models^{\pm} \varphi_1(\overline{x})[T]$ iff $\mathcal{M} \models^{\pm} \varphi_2(\overline{x})[T]$.

2.3 Signaling Kicks in

It was first observed by Jannsen [14] that V-semantics and signaling don't interact well. Consider, for instance, the following example (from [14], section 7, formulas (32) and (33)): $\theta_1 := \forall x \forall y \forall z [x = y \vee \exists u \exists w /_x [w \neq x \wedge u = z]]$ and $\theta_2 := \forall x \forall y \forall z [x = y \vee \exists y \exists w /_x [w \neq x \wedge y = z]]$ Clearly, θ_1 is a regular formula while θ_2 is not. Moreover, they only differ in the symbol used for a bound variable: u vs. y. Since variable symbols are expected to be simple placeholders, both formulas should be equivalent. Now, Eloïse has a winning strategy for θ_1, regardless the structure: $f_\vee(v) = L$ if $v(x) = v(y)$ and $f_\vee(v) = R$ otherwise; $f_{\exists u}(v) = v(z)$; $f_{\exists w/_x}(v) = v(y)$. Observe that Eloïse's strategy for θ_1 relies heavily on *signaling*: she needs a value other than $v(x)$ but her strategy function may not depend on x; however, y is *signaling* such a value.

The problem is that this strategy is not winning for θ_2: whenever Abélard picks different initial values for x and y, Eloïse will be forced to reset the value of y to that of z, breaking the global invariant of her strategy (i.e., blocking the signal). In fact, it is not hard to show that for arbitrary structures, Eloïse has no winning strategy for θ_2 which implies that $\theta_1 \not\equiv_V \theta_2$.

Now, although this is an already known example, we feel its significance has been overlooked. Variables (and specially those that are bound) ought to be a mere syntactic device, a simple placeholder. They should bear no meaning in itself. The only thing we should care about two bound variables x and y is that they are distinct and, as such, stand for distinct placeholders. In that sense u, v or w should be as good as y. In fact, we should expect to be able to drop variables altogether and replace them with some equivalent syntactic device, such as de Bruijn indices [5].

This notion is so crucial that it even has a name: *α-equivalence*. (for formal definitions see any textbook on λ-calculus, e.g. [1]). In every sensible formalism, α-equivalence implies equivalence. We already saw this does not hold in slash logic under V-semantics in general and the following example shows that it neither does restricted to regular formulas. Consider these α-equivalent, regular

formulas: $\theta_3 := \exists y \exists z /_{x,y} [z = x]$ and $\theta_4 := \exists u \exists z /_{x,u} [z = x]$. For $||\mathcal{M}|| \geq 2$ and $V = \{v \mid v(x) = v(u)\}$ it is easy to see that $\mathcal{M} \models^+ \theta_3[V]$ but $\mathcal{M} \not\models^+ \theta_4[V]$.

Invariance under α-equivalence is such a basic property that it is not surprising that neither Hodges nor Caicedo and Krynicki mention it in their papers. However the latter two assumed it to hold and this lead to some flawed results (see [15]). In the face of this, it is worth verifying that, fortunately, α-equivalence does hold under T-semantics (the proof is on Section A.2).

Proposition 1. *Let* $\overline{x} = (x_1, \ldots, x_n)$*; if* $\varphi_1(\overline{x}) \equiv_\alpha \varphi_2(\overline{x})$ *then* $\varphi_1(\overline{x}) \equiv_T \varphi_2(\overline{x})$.

The fact that invariance under α-equivalence holds on regular formulas under T-semantics but fails under V-semantics is, in our opinion, a clear indication that this is neither a feature of these logics nor they should be restricted to the regular fragment. V-semantics simply fail to generalize properly the meaning given to the slashed connectives by the T-semantics.

3 Uniform Semantics for Regular and Irregular Formulas

Classical valuations are an inadequate device to formalize the semantics of unrestricted IF-like formulas: under rebinding of variables, they simply fail to keep track of all the previous choices, which is crucial in a setting of independence restrictions. Our plan is, roughly, to replace valuations with tuples $\langle s, p \rangle$, where $s \in |\mathcal{M}|^\omega$ is an infinite sequence of choices, and p is a mapping of variables into positions of s. A variable x gets thus interpreted as $s(p(x))$. Observe one can think of the composition $s \circ p$ as denoting a classical valuation[2].

Using games, we will define what we call *S-semantics*, that is, the relations $\mathcal{M} \models^+ \varphi[S, p, h]$ and $\mathcal{M} \models^- \varphi[S, p, h]$ where S is a nonempty set of sequences taken from $|\mathcal{M}|^\omega$, and $h < \omega$ can be regarded as indicating how many "previous choices" are in scope. After checking that under this formalization some of the nice properties of classical logics hold, we will verify that, on regular formulas, S-semantics and T-semantics coincide.

The game $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$ we are about to define deviates from the customary semantic game for IF-like logics: it is a one-turn game where Abélard and Eloïse pick functions instead of elements. There are two reasons for this. On the one hand, we prefer this formulation since in this way the higher-order nature of the logic becomes arguably more apparent. On the other, this game will be generalized to an n-turn game in Section 4 to provide natural game-theoretical semantics for Hodges' flattening operator.

Before we go into the definitions, we need some notation for the manipulation of functions (and, in particular, infinite sequences). Let $f : X \to Y$, we denote with $f[x \mapsto y]$ the function such that $f[x \mapsto y](x) = y$ and $f[x \mapsto y](z) = f(z)$ for all $z \neq x$. As usual, if $X' \subseteq X$ then $f \restriction X' : X' \to Y$ will be the restriction of f to X'.

[2] Almost all of our presentation can probably be done using sequences of finite length. Apart from an arguably more cumbersome presentation, a downside of this would be that $s \circ p$ would then represent a classical valuation but one with finite image.

The board. The game is played over the syntactic tree of a formula. Every node of the tree, except the $\sim$-nodes, belong to one of the players: those initially under an even number of $\sim$-nodes belong to Eloïse, the rest belongs to Abélard. The initial assignment of nodes to a player will be remembered along the game. Furthermore, some nodes may be decorated with functions during the game: $\exists$-nodes can be decorated with any function $f : |\mathcal{M}|^\omega \to |\mathcal{M}|$; $\vee$-nodes can be decorated with any function $f : |\mathcal{M}|^\omega \to \{L, R\}$. Initially, these nodes have no decoration. Plus, there is a triple $\langle S, p, h\rangle$ and a placeholder (initially empty) for a sequence in $|\mathcal{M}|^\omega$.

The turn. The turn is composed of two clearly distinguished phases. In the first phase, both players decorate all their nodes with proper functions. The order in which they tag their nodes is not important as long as they don't get to see their opponent's choices in advance. For simplicity, we will assume they both play simultaneously. For the second phase, we introduce a third agent, sometimes known as Nature, that can be seen as random choices. Nature first picks some sequence from S and puts it in the placeholder. Next, it proceeds to *evaluate* the result of the turn using the following recursive procedure:

R1. If the tree is of the form $\sim\psi$, Nature replaces it with ψ and evaluation continues.

R2. If the tree is of the form $\psi_1 \vee/_{y_1,\ldots,y_k} \psi_2$, then its root must have been decorated with some $f : |\mathcal{M}|^\omega \to \{L, R\}$. Nature then picks a sequence $r \in |\mathcal{M}|^\omega$ such that $r(i) = s(i)$ for every $i \notin \{p(y_1), \ldots, p(y_n)\} \cup \{k \mid k \geq h\}$, where s stands for the sequence on the placeholder, and evaluates $f(r)$. Observe that the values the player was not supposed to consider are replaced with arbitrary values prior to evaluating the function. The tree then is replaced with ψ_1 if the result is L or with ψ_2 otherwise, and evaluation proceeds.

R3. If the tree is of the form $\exists x/_{y_1,\ldots,y_k}\psi$, then it must be decorated with some $f : |\mathcal{M}|^\omega \to |\mathcal{M}|$. Nature here also picks a sequence $r \in |\mathcal{M}|^\omega$ such that $r(i) = s(i)$ for every $i \notin \{p(y_1), \ldots, p(y_n)\} \cup \{k \mid k \geq h\}$, where s stands for the sequence on the placeholder, and evaluates $f(r)$. Let us call this value b. Nature records this choice by replacing the sequence in the placeholder with $s[h \mapsto b]$; x is bound to b by replacing p with $p[x \mapsto h]$ and h is incremented by one. Finally, the tree is replaced with ψ and evaluation proceeds.

R4. Finally, if the root of the tree is of the form $R(t_1, \ldots, t_k)$, evaluation ends. Eloïse is declared the winner of the match whenever this node belongs to her and $\mathcal{M} \models R(t_1, \ldots, t_k)[s \circ p]$, or the node belongs to Abélard and $\mathcal{M} \not\models R(t_1, \ldots, t_k)[s \circ p]$. In any other case, the winner is Abélard.

Winning strategies. A *strategy* for a player of the game $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$ is just the collection of functions used to decorate the syntactic tree of φ. Furthermore, the strategy is *winning* if it guarantees that the player will win every match of the game, regardless the strategy of the opponent and the choices made by Nature. Observe this game is of *imperfect information*: Abélard and Eloïse must

play *simultaneously* (i.e. ignoring the opponent move) and the initial valuation is "randomly" picked by Nature. Therefore, some games are probably undetermined, that is, none of the players have a winning strategy.

We are now ready to give our game-semantic notion of *truth* and *falsity*. Observe, though, that this will be restricted to only certain p and h. The rationale for this will become clear later (cf. Example 1 and Lemma 1).

Definition 1. *We say that* $p : \mathrm{Vars} \to \omega$ *and* $h < \omega$ *are a* proper context *for a formula* φ *if* $p \restriction \mathrm{Fv}(\varphi)$ *is injective and* $\{p(x) \mid x \in \mathrm{Fv}(\varphi)\} \subseteq \{0, \ldots, h-1\}$.

Definition 2 ($\models^+$ and $\models^-$ for S-semantics). *Given a formula* φ*, a suitable model* $\mathcal{M}$*, a nonempty set* $S \subseteq |\mathcal{M}|^\omega$ *and a proper context for* φ*,* $p : \mathrm{Vars} \to \omega$ *and* $h < \omega$*, we define:* $\mathcal{M} \models^+ \varphi[S, p, h]$ *iff Eloïse has a winning strategy for* $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$*;* $\mathcal{M} \models^- \varphi[S, p, h]$ *iff Abélard has a winning strategy for* $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$.

When S is the singleton set $\{s\}$ we may alternatively write $\mathcal{M} \models^+ \varphi[s, p, h]$ and $\mathcal{M} \models^- \varphi[s, p, h]$. Furthermore, we will write $\mathcal{M} \models^+ \varphi$ if $\mathcal{M} \models^+ \varphi[|\mathcal{M}|^\omega, p, h]$ whenever p and h are a proper context for φ (and analogously for $\mathcal{M} \models^- \varphi$).

Example 1. Consider $\theta := \exists x\, [x \neq y]$. One would expect that for any $\mathcal{M}$ with at least two elements, $\mathcal{M} \models^+ \theta$ show hold. However, Eloïse has no winning strategy on $\mathsf{G}(\mathcal{M}, \theta, S, p, h)$ when $p(y) = h$. The problem here is that the value selected by Eloïse's function for x, whatever it is, will be recorded in position h, thus *overwriting* the value of y. Observe, though, that if p and h are a proper context for θ, then it cannot be the case that $p(y) \geq h$.

Example 2. Let us revisit the irregular formula θ_2 from Section 2.3. We shall verify that for any model $\mathcal{M}$, $\mathcal{M} \models^+ \theta_2$. For this, consider the following strategy for Eloïse: $f_\vee(s) = L$ if $s(h) = s(h+1)$ and $f_\vee(s) = R$ otherwise; $f_{\exists y}(s) = s(h+2)$; $f_{\exists u/x}(s) = s(h+1)$. The reader should verify that this is essentially the same strategy used for θ_1 in Section 2. Observe that, for example, $s(h+1)$ plays the same role that $v(y)$ played in the latter, except that by using an offset from h (i.e., from the position in s where the value for the outermost quantifier was recorded) instead of the variable name, we escape from the deathtraps created by the rebinding of variables. In fact, Eloïse's winning strategy in this example works for any renaming of variables of θ_2.

So far we have defined $\models^+$ and $\models^-$ with respect to sets of sequences using a game theoretical approach. We can also give a compositional characterization, in the line of [11] and [4] (the proof of Theorem 1 is on Section A.3).

Definition 3. *Let* $f : A^B \to C$ *and let* $Y \subseteq B$*. We say that* f *is* Y*-independent if for all* $g_1, g_2 \in A^B$ *such that* $g_1(x) = g_2(x)$ *whenever* $x \notin Y$*,* $f(g_1) = f(g_2)$.

Theorem 1 (Compositionality of S-semantics). *Let* $\mathcal{M}$ *be a suitable model, let* $S \subseteq |\mathcal{M}|^\omega$ *be nonempty and let* $p : \mathrm{Vars} \to \omega$ *and* $h < \omega$ *be a proper context.*

1. $\mathcal{M} \models^+ R(t_1, \ldots, t_k)[S, p, h]$ *iff* $\mathcal{M} \models R(t_1, \ldots, t_k)[s \circ p]$ *for all* $s \in S$
2. $\mathcal{M} \models^- R(t_1, \ldots, t_k)[S, p, h]$ *iff* $\mathcal{M} \not\models R(t_1, \ldots, t_k)[s \circ p]$ *for all* $s \in S$
3. $\mathcal{M} \models^+ {\sim}\psi[S, p, h]$ *iff* $\mathcal{M} \models^- \psi[S, p, h]$
4. $\mathcal{M} \models^- {\sim}\psi[S, p, h]$ *iff* $\mathcal{M} \models^+ \psi[S, p, h]$
5. $\mathcal{M} \models^+ \psi_1 \vee\!/_{y_1,\ldots,y_k} \psi_2[S, p, h]$ *iff there is an* $f : S \to \{L, R\}$ *such that*
   - f *is* $\{p(y_1), \ldots, p(y_k)\} \cup \{k \mid k \geq h\}$*-independent;*
   - $\mathcal{M} \models^+ \psi_1[S_L, p, h]$*, where* $S_L = \{s \mid s \in S, f(s) = L\}$*; and*
   - $\mathcal{M} \models^+ \psi_2[S_R, p, h]$*, where* $S_R = \{s \mid s \in S, f(s) = R\}$
6. $\mathcal{M} \models^- \psi_1 \vee\!/_{y_1,\ldots,y_k} \psi_2[S, p, h]$ *iff* $\mathcal{M} \models^- \psi_1[S, p, h]$ *and* $\mathcal{M} \models^- \psi_1[S, p, h]$
7. $\mathcal{M} \models^+ \exists x/_{y_1,\ldots,y_k} \psi[S, p, h]$ *iff there is a function* $f : S \to |\mathcal{M}|$ *such that*
   - f *is* $\{p(y_1), \ldots, p(y_k)\} \cup \{k \mid k \geq h\}$*-independent; and*
   - $\mathcal{M} \models^+ \psi[\widetilde{S}, p[x \mapsto h], h + 1]$*, where* $\widetilde{S} = \{s[h \mapsto f(s)] \mid s \in S\}$
8. $\mathcal{M} \models^- \exists x/_{y_1,\ldots,y_k} \psi[S, p, h]$ *iff* $\mathcal{M} \models^- \psi[\widetilde{S}, p[x \mapsto h], h + 1]$ *for* $\widetilde{S} = \{s[h \mapsto a] \mid s \in S, a \in |\mathcal{M}|\}$

Definition 4 ($\equiv_h$ and $\equiv$). *Given* $h < \omega$*, we write* $\varphi_1 \equiv_h \varphi_2$ *if* $\mathrm{Fv}(\varphi_1) = \mathrm{Fv}(\varphi_2)$ *and for every suitable model* $\mathcal{M}$*, every nonempty* $S \subseteq |\mathcal{M}|^\omega$ *and every* $p : \mathrm{Vars} \to \omega$ *such that* p*,* h *is a proper context for* φ_1*,* $\mathcal{M} \models^\pm \varphi_1[S, p, h]$ *iff* $\mathcal{M} \models^\pm \varphi_2[S, p, h]$*. Furthermore, if for every* $h < \omega$ *we have* $\varphi_1 \equiv_h \varphi_2$ *, then we say that the formulas are* S-equivalent*, notated* $\varphi_1 \equiv \varphi_2$*.*

Since strategies for the game $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$ must deal with sequences but not with variable values, it is straightforward to verify the following:

Proposition 2. *If* $\varphi_1 \equiv_\alpha \varphi_2$ *then* $\varphi_1 \equiv \varphi_2$*.*

In first-order logic, the truth of a formula depends only on the value of its free variables (i.e., if $\mathcal{M} \models \varphi[v]$ and v and v' differ only on variables that are not free in φ, then $\mathcal{M} \models \varphi[v']$). We will show next that in our setting, there are three operations on valuations that preserve satisfaction. In what follows, for $S \subseteq A^\omega$, we define $S \restriction n = \{(s(0), \ldots, s(n-1)) \mid s \in S\}$; we call h-permutation to any bijective function $\pi : \omega \to \omega$ such that $\pi(i) = i$ for all $i \geq h$; and $S \circ \pi = \{s \circ \pi \mid s \in S\}$.

Theorem 2. *For all suitable* $\mathcal{M}$*, nonempty* $S \subseteq |\mathcal{M}|^\omega$ *and proper contexts for* φ*,* p *and* h*:*

1. *If* $\widetilde{p} \restriction \mathrm{Fv}(\varphi) = p \restriction \mathrm{Fv}(\varphi)$*, then* $\mathcal{M} \models^\pm \varphi[S, p, h]$ *iff* $\mathcal{M} \models^\pm \varphi[S, \widetilde{p}, h]$*.*
2. *If* $\widetilde{S}$ *is such that* $\widetilde{S} \restriction h = S \restriction h$*, then* $\mathcal{M} \models^\pm \varphi[S, p, h]$ *iff* $\mathcal{M} \models^\pm \varphi[\widetilde{S}, p, h]$*.*
3. *If* π *is an* h*-permutation then* $\mathcal{M} \models^\pm \varphi[S, p, h]$ *iff* $\mathcal{M} \models^\pm \varphi[S \circ \pi, \pi \circ p, h]$*.*

We are now ready to show that, when restricted to regular formulas, the equivalence notions of S-semantics and T-semantics match. Of course, this implies that the set of valid (regular) formulas of both logics is the same and, because of Proposition 2, S-semantics is a proper generalization of T-semantics (the proof is on Section A.4).

Theorem 3. *Let* φ_1 *and* φ_2 *be regular formulas. Then* $\varphi_1(x_0, \ldots, x_{h-1}) \equiv_T \varphi_2(x_0, \ldots, x_{h-1})$ *iff* $\varphi_1 \equiv_h \varphi_2$*.*

4 Game Theoretical Semantics for IF with Flattening

We say that φ is *true* in $\mathcal{M}$ when $\mathcal{M} \models^+ \varphi$ and that is *false* if $\mathcal{M} \models^- \varphi$. Clearly, if $\mathcal{M} \models^+ \varphi$ then $\mathcal{M} \not\models^- \varphi$, and if $\mathcal{M} \models^- \varphi$ then $\mathcal{M} \not\models^+ \varphi$. However there are *sentences* which may be neither true nor false in a model. Hodges considers the problem of adding classical negation to slash logic. He wants, for instance, $\mathcal{M} \models^\pm \neg\varphi$ iff $\mathcal{M} \not\models^\pm \varphi$ to hold; restoring, for sentences, the identity between being not-true and being false. To this end, he introduces the *flattening operator* $\downarrow$, and stipulates $\neg\psi \equiv {\sim}{\downarrow}\psi$ [11].

Since in this section we move to slash logic enriched with the flattening operator, we assume from here on that $\downarrow$ may occur freely in a formula. First of all, we need to specify its semantics. Hodges used a compositional definition; therefore, we will take Theorem 1 to be a compositional definition of $\models^+$ and $\models^-$ for slash logic and extend it to handle $\downarrow$. Observe we are simply adapting his notation according to our presentation.

Definition 5 ($\models^+$ and $\models^-$ for S-semantics with $\downarrow$). *We define $\models^+$ and $\models^-$ as the relation induced by clauses 1–8 of Theorem 1, plus*

9. $\mathcal{M} \models^+ {\downarrow}\psi[S, p, h]$ *iff* $\mathcal{M} \models^+ \psi[s, p, h]$ *for every* $s \in S$
10. $\mathcal{M} \models^- {\downarrow}\psi[S, p, h]$ *iff* $\mathcal{M} \not\models^+ \psi[s, p, h]$ *for every* $s \in S$

Hodges seems to suggest that no natural game-theoretical semantics can be given for this logic[3]. In any case, this can indeed be done. We define next the game $\mathsf{G}_\downarrow(\mathcal{M}, \varphi, S, p, h)$, which extends the rules of the game described in Section 3 to deal with formulas containing arbitrary occurrences of $\downarrow$.

The board. The board is essentially the same one used for $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$. The syntactic tree of the formula now may contain $\downarrow$-nodes; these are assigned to players using the same criteria: those under an even number of $\sim$-nodes belong to Eloïse, the remaining ones to Abélard. Just like the leafs of the tree, $\downarrow$-nodes will not be decorated.

The turns. Unlike the one of Section 3, this game may last more than one turn. At any point of the game, the remaining number of turns will be bounded by the number of nested occurrences of $\downarrow$-nodes in the game-board. The opening turn is played exactly like in Section 3, although we still need to stipulate what happens, during the evaluation phase, if Nature arrives to a formula of the form ${\downarrow}\psi$. Observe that this means that if no $\downarrow$ occurs in φ, then $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$ and $\mathsf{G}_\downarrow(\mathcal{M}, \varphi, S, p, h)$ are essentially the same game.

So, summing up, when the game starts, both players decorate their nodes simultaneously; then Nature picks a sequence and puts it in the placeholder, and finally starts the evaluation phase (cf. rules **R1**–**R4** in Section 3). If evaluation reaches a leaf (i.e., an atom), then the game ends, and the winner is determined according to rule **R4**. For the extra case we add the following rule:

R5. If the tree is of the form ${\downarrow}\psi$, then the turn ends.

[3] The exact quote is: "In the presence of $\downarrow$, we can't define a game $G(\phi, A)$ for arbitrary A and ϕ." [11, p. 556].

The initial turn differs slightly from the subsequent ones, where the formula on the board will be always of the form $\downarrow\psi$. Now both players get to redecorate their nodes, except that in this case, they proceed one after the other. The player who owns the $\downarrow$-node at the root gets to do it first. After this, Nature replaces the tree with ψ and proceeds to the evaluation phase following rules **R1**–**R5**.

We won't go into a formal description of a winning strategy for this game. We simply take it to be some form of oracle that, when followed, guarantees that the game ends in a winning position.

Theorem 4 (Game semantics for $\downarrow$). *Given a formula φ, a suitable $\mathcal{M}$, a nonempty $S \subseteq |\mathcal{M}|^\omega$ and a proper context for φ, $\langle p, h\rangle$, the following holds: $\mathcal{M} \models^+ \varphi[S, p, h]$ iff Eloïse has a winning strategy for $\mathsf{G}_\downarrow(\mathcal{M}, \varphi, S, p, h)$; $\mathcal{M} \models^- \varphi[S, p, h]$ iff Abélard has a winning strategy for $\mathsf{G}_\downarrow(\mathcal{M}, \varphi, S, p, h)$.*

5 Conclusions

We think that invariance under α-equivalence is a property that no sane formalism can disregard. By decoupling *values* from *name for values* we have been able to successfully generalize Hodges' T-semantics from regular formulas to unrestricted ones. To achieve this we had to pay a small price: abandon the well-established use of classical valuations.

In [3], Caicedo, Dechesne and Janssen took a different path and investigated a weaker notion of equivalence for V-semantics. They say, for instance, that $\varphi \equiv_{xz} \psi$ if $\{x, y\} \not\subseteq \mathrm{Fv}(\varphi) \cup \mathrm{Fv}(\psi)$ and $\mathcal{M} \models^\pm \varphi[V]$ iff $\mathcal{M} \models^\pm \psi[V]$, *provided that x and z are excluded from the domain of the valuations in V* and go into great technical efforts to properly characterize the normal form equivalences initially presented in [4]. We believe this route leads ultimately to a dead-end: V-semantics are buying very little and are too hard to reason about.

We favor a simpler approach, akin to the usual practice in classical logics. For convenience, stick to regular formulas, use a lightweight formalism, like T-semantics, and finally resort to Theorem 3 and Propositions 1 and 2 to generalize the result. This way, for instance, the normal forms results of [4] can easily be shown to hold under S-semantics, in a much more general way than in [3].

In the last part of the paper we looked at the $\downarrow$-operator from a novel game-theoretical perspective. We believe this will ultimately help to gain more insight about this operator.

Acknowledgments. This work was partially supported by grant PICTR 0172 from the Agencia Nacional de Promoción Científica y Tecnológica (Argentina); by grant G.0344.05 from the Research Foundation Flanders (FWO-Vlaanderen) and by the European Union under the FP6-IST-FET programme, Project n. FP6-14915, GeoPKDD: Geographic Privacy-Aware Knowledge Discovery and Delivery. D. Gorín is partially supported by a grant from CONICET, Argentina.

References

1. Barendregt, H.: The Lambda Calculus: its syntax and semantics, 2nd edn. Studies in logic and the foundations of mathematics, vol. 103. North Holland, Amsterdam (1985)
2. Barwise, J.: On branching quantifiers in English. Journal of Philosophical Logic 8, 47–80 (1979)
3. Caicedo, X., Dechesne, F., Jansen, T.M.V.: Equivalence and quantifier rules for logic with imperfect information. Prepublication Series PP-2007-20. ILLC (2007)
4. Caicedo, X., Krynicki, M.: Quantifiers for reasoning with imperfect information and Σ_1^1-logic. Contemporary Mathematics 235, 17–31 (1999)
5. de Bruijn, N.G.: Lambda calculus notation with nameless dummies. A tool for automatic formula manipulation with application to the Church-Rosser theorem. Indagationes Mathematicae 34, 381–392 (1972)
6. Dechesne, F.: Game, Sets, Maths: formal investigations into logic with imperfect information. PhD thesis, Department of Philosophy, University of Tilburg, The Netherlands (2005)
7. Enderton, H.: Finite partially ordered quantifiers. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 16, 393–397 (1970)
8. Henkin, L.: Some remarks on infinitely long formulas. In: Infinitistic Methods: Proceedings of the Symposium on Foundations of Mathematics, pp. 167–183. Pergamon Press, Oxford (1961)
9. Hintikka, J.: The Principles of Mathematics Revisited. Cambridge University Press, Cambridge (1996)
10. Hintikka, J., Sandu, G.: Game-theoretical semantics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of logic and language, ch. 6. MIT Press, Cambridge (1997)
11. Hodges, W.: Compositional semantics for a language of imperfect information. Logic Journal of the IGPL 5(4) (1997)
12. Hodges, W.: Logics of imperfect information: why sets of assignments? In: Proceedings of the 7th Augustus de Morgan Workshop Interactive Logic: Games and Social Software (2005)
13. Hodges, W.: Some strange quantifiers. In: Mycielski, J., Rozenberg, G., Salomaa, A. (eds.) Structures in Logic and Computer Science. LNCS, vol. 1261, pp. 51–65. Springer, Heidelberg (1997)
14. Janssen, T.M.V.: Independent choices and the interpretation of IF logic. Journal of Logic, Language and Information 11(3), 367–387 (2002)
15. Janssen, T.M.V., Dechesne, F.: Signalling in IF games: a tricky business. In: The age of alternative logics, ch. 15, pp. 221–241. Springer, Heidelberg (2006)
16. Walkoe Jr., W.: Finite partially-ordered quantification. Journal of Symbolic Logic 35(4), 535–555 (1970)

A Technical Appendix

A.1 Compositionality for T-Semantics

For completeness, we introduce the compositional formulation of T-semantics. Our presentation is closer to the one due to Caicedo and Krynicki [4], but they can easily be shown to be equivalent.

In the following definition, if $t = (t_1, \ldots, t_n) \in |\mathcal{M}|^n$, then $v_t : \mathrm{Vars} \to |\mathcal{M}|$ stands for any classical first order valuation such that $v(x_i) = t_i$ for $i = 1, \ldots, n$. For a definition of Y-independence, refer to Definition 3 on page 170.

Definition 6 (Compositionality of T-semantics). *Let $\overline{x} = x_1, \ldots, x_n$, let $\psi(\overline{x})$ be a regular formula, let $\mathcal{M}$ be a suitable model and let $T \subseteq |\mathcal{M}|^n$ be a set of deals of length n. We define $\models^+$ and $\models^-$ as follows:*

1. *if $\varphi(\overline{x})$ is atomic or negated atomic,*
   - $\mathcal{M} \models^+ \varphi(\overline{x})[T]$ *iff* $\mathcal{M} \models \varphi(\overline{x})[v_t]$ *for all* $t \in T$
   - $\mathcal{M} \models^- \varphi(\overline{x})[T]$ *iff* $\mathcal{M} \not\models \varphi(\overline{x})[v_t]$ *for no* $t \in T$
2. *if $\varphi(\overline{x}) = {\sim}\psi(\overline{x})$,*
   - $\mathcal{M} \models^+ \varphi(\overline{x})[T]$ *iff* $\mathcal{M} \models^- \psi(\overline{x})[T]$
   - $\mathcal{M} \models^- \varphi(\overline{x})[T]$ *iff* $\mathcal{M} \models^+ \psi(\overline{x})[T]$
3. *if $\varphi(\overline{x}) = \psi_1(\overline{x}) \vee /_{x_{n_1},\ldots,x_{n_k}} \psi_2(\overline{x})$ for some $\{n_1, \ldots, n_k\} \subseteq \{1, \ldots, n\}$,*
   - $\mathcal{M} \models^+ \varphi(\overline{x})[T]$ *iff there is a function $g : T \to \{L, R\}$ such that*
     - *g is $\{n_1, \ldots, n_k\}$-independent;*
     - *$\mathcal{M} \models^+ \psi_1(\overline{x})[T_L]$, where $T_L = \{t \mid t \in T, g(t) = L\}$; and*
     - *$\mathcal{M} \models^+ \psi_2(\overline{x})[T_R]$, where $T_R = \{t \mid t \in T, g(t) = R\}$*
   - $\mathcal{M} \models^- \varphi(\overline{x})[T]$ *iff* $\mathcal{M} \models^- \psi_1(\overline{x})[T]$ *and* $\mathcal{M} \models^- \psi_1(\overline{x})[T]$
4. *if $\varphi(\overline{x}) = \exists y /_{x_{n_1},\ldots,x_{n_k}} \psi(\overline{x}, y)$ and $y \notin \{x_1, \ldots, x_n\}$ for some $\{n_1, \ldots, n_k\} \subseteq \{1, \ldots, n\}$,*
   - $\mathcal{M} \models^+ \varphi(\overline{x})[T]$ *iff there is a function $g : T \to |\mathcal{M}|$ such that*
     - *g is $\{n_1, \ldots, n_k\}$-independent; and*
     - *$\mathcal{M} \models^+ \psi(\overline{x}, y)[T']$, where*

$$T' = \{(t_1, \ldots, t_n, g(t_1, \ldots, t_n)) \mid (t_1, \ldots, t_n) \in T\}$$

   - $\mathcal{M} \models^- \varphi(\overline{x})[T]$ *iff* $\mathcal{M} \models^- \psi(\overline{x}, y)[T']$ *for*

$$T' = \{(t_1, \ldots, t_n, a) \mid (t_1, \ldots, t_n) \in T, a \in |\mathcal{M}|\}$$

A.2 Proof of Proposition 1

Let $\overline{x} = x_1, \ldots, x_n$. It is enough to consider the case where $\varphi_1(\overline{x})$ is equal to $\varphi_2(\overline{x})$ except that the bound variable u of $\varphi_1(\overline{x})$ is replaced by v in $\varphi_2(\overline{x})$, within the scope of the same quantifier. By the Full Abstraction Theorem for T-semantics [11, Theorem 7.6] it suffices to prove that $\exists u /_{x_{n_1},\ldots,x_{n_k}} \psi_1(\overline{x}, u) \equiv_T \exists v /_{x_{n_1},\ldots,x_{n_k}} \psi_2(\overline{x}, v)$, where $\psi_2(\overline{x}, v)$ is obtained from $\psi_1(\overline{x}, v)$ when replacing the free variable u with v. One can prove by induction that $\psi_1(\overline{x}, v) \equiv_T \psi_2(\overline{x}, u)$. The key point here is that at item 1 of Definition 6 the name u or v is irrelevant, as long as they came in the same order in the lists $(\overline{x}, u)$ and $(\overline{x}, v)$.

A.3 Proof of Theorem 1

For the right-to-left implication, one proceeds by structural induction and shows that, for the $\exists$ and $\vee$ cases, the function f plus the strategy for the subformula(s) constitute a winning strategy. For the left-to-right implication, one only needs to see that if a player has a winning strategy on the game $\mathsf{G}(\mathcal{M}, \varphi, S, p, h)$, then he also has a winning strategy where all the functions that constitute it satisfy the independence restriction, and this is relatively straightforward (the full details can be seen, e.g., in [3, Theorems 4.7 and 4.8]). In every case, one also has to check that contexts are proper, but this is trivial.

A.4 Proof of Theorem 3

We first need to establish the following lemma.

Lemma 1. *Let $\varphi(x_0, \ldots, x_{h-1})$ be a regular formula such that in every branch of its syntactic tree, variables are bound in the same order. Furthermore, let $p : \mathrm{Vars} \to \omega$ be such that $p(x_i) = i$ for $0 \leq i < h$. Then $\mathcal{M} \models^{\pm} \varphi[S, p, h]$ iff $\mathcal{M} \models^{\pm} \varphi(x_0 \ldots x_{h-1})[S \upharpoonright h]$.*

Proof. Let $\overline{x} = x_0, \ldots, x_{h-1}$. Suppose the list of occurrences of bound variables appearing in each branch of the syntactic tree of $\varphi(\overline{x})$ (from the root to the leaves) is a prefix of $x_h, x_{h+1}, x_{h+2}, \ldots$ The proof goes by induction in the complexity of φ. The atomic and negation are straightforward. Let us analyze the case $\varphi = \exists x_h /_{x_{n_1}, \ldots, x_{n_k}} \psi(\overline{x}, x_h)$, for some $\{n_1, \ldots, n_k\} \subseteq \{0, \ldots, h-1\}$.

For the left to right implication, suppose $\mathcal{M} \models^{+} \varphi(\overline{x})[S, p, h]$. By Theorem 1 (item 7), there is a function $f : S \to |\mathcal{M}|$ such that f is $\{p(x_{n_1}), \ldots, p(x_{n_k})\} \cup \{k \mid k \geq h\}$-independent and $\mathcal{M} \models^{+} \psi(\overline{x}, x_h)[S', p[x_h \mapsto h], h+1]$, where $S' = \{s[h \mapsto f(s)] \mid s \in S\}$. Since $p = p[x_h \mapsto h]$, by inductive hypothesis we get $\mathcal{M} \models^{+} \psi(\overline{x}, x_h)[S' \upharpoonright h+1]$. Fix $z \in S$ and define $g : S \upharpoonright h \to |\mathcal{M}|$ as $g(s_0, \ldots, s_{h-1}) = f(s_0 \ldots s_{h-1} z(h) z(h+1) \ldots)$ for every $(s_0, \ldots, s_{h-1}) \in S \upharpoonright h$. Since f is $\{n_1, \ldots, n_k\} \cup \{k \mid k \geq h\}$-independent then g is clearly well-defined and $\{n_1, \ldots, n_k\}$-independent. Furthermore,

$$S' \upharpoonright h+1 = \{(s_0, \ldots, s_{h-1}, g(s)) \mid (s_0, \ldots, s_{h-1}) \in S \upharpoonright h\}.$$

By Definition 6 (item 4), $\mathcal{M} \models^{+} \varphi(\overline{x})[S \upharpoonright h]$.

For the other direction, suppose $\mathcal{M} \models^{+} \varphi(\overline{x})[S \upharpoonright h]$. By Definition 6 (item 4), there exists some function $g : S \upharpoonright h \to |\mathcal{M}|$ that is $\{n_1, \ldots, n_k\}$-independent and such that $\mathcal{M} \models^{+} \psi(\overline{x}, x_h)[T']$, where

$$T' = \{(t_1, \ldots, t_h, g(t_1, \ldots, t_h)) \mid (t_1, \ldots, t_h) \in S \upharpoonright h\}.$$

Observe that $T' = S' \upharpoonright h+1$, where

$$S' = \{s[h \mapsto g(s(0), \ldots, s(h-1))] \mid s \in S\}.$$

By inductive hypothesis and the fact that $p[x_h \mapsto h] = p$, we have $\mathcal{M} \models^+ \psi(\overline{x}, x_h)[S', p[x_h \mapsto h], h+1]$. Define $f : S \to |\mathcal{M}|$ as $f(s) = g(s(0), \ldots, s(h-1))$ for $s \in S$. By definition, f is clearly $\{k \mid k \geq h\}$-independent, and since g is $\{n_1, \ldots, n_k\}$-independent, f also is. By Theorem 1 (item 7) we conclude $\mathcal{M} \models^+ \varphi(\overline{x})[S, p, h]$.

The case for $\models^-$ and $\varphi = \exists x_h /_{x_{n_1}, \ldots, x_{n_k}} \psi(\overline{x}, x_n)$ is straightforward. A similar argument can be used for the case $\varphi(\overline{x}) = \psi_1(\overline{x}) \vee /_{x_{n_1}, \ldots, x_{n_k}} \psi_1(\overline{x})$.

We are now ready to prove the theorem. We will only show it for $\models^+$, the argument for $\models^-$ is similar. In what follows $\overline{x}$ will stand for $x_0, \ldots, x_{h-1}$. From left to right, by the counterpositive, suppose that $\mathcal{M} \models^+ \varphi_1(\overline{x})[S, p, h]$ and $\mathcal{M} \not\models^+ \varphi_2(\overline{x})[S, p, h]$, for some suitable model $\mathcal{M}$ and some $p : \mathrm{Vars} \to \omega$ such that p, h is a proper context for φ_1 (and for φ_2, since $\mathrm{Fv}(\varphi_1) = \mathrm{Fv}(\varphi_2)$). One can build an h-permutation π such that $\pi(p(x_i)) = i$ for $0 \leq i < h$ and using Theorem 2 one gets $\mathcal{M} \models^+ \varphi_1(\overline{x})[S \circ \pi, \pi \circ p, h]$ but $\mathcal{M} \not\models^+ \varphi_2(\overline{x})[S \circ \pi, \pi \circ p, h]$. By Proposition 2, we can pick regular $\varphi_1' \equiv_\alpha \varphi_1$ and $\varphi_2' \equiv_\alpha \varphi_2$ where variables are bound in the same order on every branch of their syntactic trees and, using Lemma 1 we obtain $\mathcal{M} \models^+ \varphi_1'(\overline{x})[S \circ \pi \restriction h]$ and $\mathcal{M} \not\models^+ \varphi_2'(\overline{x})[S \circ \pi \restriction h]$, which implies $\varphi_1(\overline{x}) \not\equiv_T \varphi_2(\overline{x})$ using Proposition 1.

From right to left, suppose $\varphi_1(\overline{x}) \not\equiv_T \varphi_2(\overline{x})$, i.e., $\mathcal{M} \models^+ \varphi_1(\overline{x})[T]$ and $\mathcal{M} \not\models^+ \varphi_2(\overline{x})[T]$, for some suitable model $\mathcal{M}$ and some trump $T \subseteq |\mathcal{M}|^h$. Define

$$S = \{t_1 \cdots t_h s \mid (t_1, \ldots, t_h) \in T, s \in |\mathcal{M}|^\omega\}$$

and $p(x_i) = i$. Again, using invariance under α-equivalence and Lemma 1 we conclude $\mathcal{M} \models^+ \varphi_1[S, p, n]$ and $\mathcal{M} \not\models^+ \varphi_2[S, p, n]$.

A.5 Proof of Theorem 4

The proof goes by induction on φ and is, essentially equivalent to the one for Theorem 1 except that we also have to account for the case where φ is $\downarrow\psi$. Suppose first $\mathcal{M} \models^+ \downarrow\psi[S, p, h]$; this means that $\mathcal{M} \models^+ \psi[s, p, h]$ for all $s \in S$. We want to construct a winning strategy for Eloïse for the game $\mathsf{G}_\downarrow(\mathcal{M}, \downarrow\psi, S, p, h)$. The first turn is irrelevant; for the second one, Eloïse simply has to consider the valuation s in the placeholder and use the winning strategy for $\mathsf{G}_\downarrow(\mathcal{M}, \psi, \{s\}, p, h)$ that, by inductive hypothesis, she has. For the other direction, suppose Eloïse has a winning strategy for $\mathsf{G}_\downarrow(\mathcal{M}, \downarrow\psi, S, p, h)$. This implies she has a winning strategy for $\mathsf{G}_\downarrow(\mathcal{M}, \psi, \{s\}, p, h)$ for all $s \in S$: play whatever she would play as her second turn in $\mathsf{G}_\downarrow(\mathcal{M}, \downarrow\psi, S, p, h)$ if Nature happened to pick s. By inductive hypothesis, this means $\mathcal{M} \models^+ \psi[s, p, h]$ for all $s \in S$ and, thus, $\mathcal{M} \models^+ \downarrow\psi[S, p, h]$.

Suppose now $\mathcal{M} \models^- \downarrow\psi[S, p, h]$; then for every $s \in S$, $\mathcal{M} \not\models^+ \psi[s, p, h]$. From here we derive a winning strategy for Abélard on $\mathsf{G}_\downarrow(\mathcal{M}, \downarrow\psi, S, p, h)$ as follows. The first turn is irrelevant; for the second one, an $s \in S$ has been picked and Eloïse has played first following some strategy. Observe that this strategy is also a possible strategy for $\mathsf{G}_\downarrow(\mathcal{M}, \psi, \{s\}, p, h)$. But by inductive hypothesis, since $\mathcal{M} \not\models^+ \psi[s, p, h]$, it cannot be a winning strategy for this game, i.e. Abélard has some strategy that defeats hers. Abélard simply has to use this strategy

from this point on and will win the game. Analogously, if Abélard has a winning strategy for $\mathsf{G}_{\downarrow}(\mathcal{M}, \downarrow\psi, S, p, h)$, then for every $s \in S$ picked by Nature and any strategy followed by Eloïse, there is a way in which Abélard can play and win the game. But this means that for no $s \in S$, Eloïse has a winning strategy for $\mathsf{G}_{\downarrow}(\mathcal{M}, \psi, \{s\}, p, h)$ and, thus, by inductive hypothesis, $\mathcal{M} \not\models^{+} \psi[s, p, h]$ and, finally, $\mathcal{M} \models^{-} \downarrow\psi[S, p, h]$.

One-and-a-Halfth Order Terms: Curry-Howard and Incomplete Derivations

Murdoch J. Gabbay[1] and Dominic P. Mulligan[2]

[1] http://www.gabbay.org.uk
[2] http://www.macs.hw.ac.uk/~dpm8/

Abstract. The Curry-Howard correspondence connects Natural Deduction derivation with the lambda-calculus. Predicates are types, derivations are terms. This supports reasoning from assumptions to conclusions, but we may want to reason 'backwards' from the desired conclusion towards the assumptions. At intermediate stages we may have an 'incomplete derivation', with 'holes'.

This is natural in informal practice; the challenge is to formalise it. To this end we use a one-and-a-halfth order technique based on nominal terms, with *two* levels of variable. Predicates are types, derivations are terms — and the two levels of variable are respectively the assumptions and the 'holes' of an incomplete derivation.

1 Introduction

The Curry-Howard correspondence [US06, PCW05] connects logic with typed λ-calculus: predicates are types; derivations are terms; discharge is λ-abstraction; modus ponens is application; β-reduction is proof-normalisation. For example,[1]

$$\dfrac{\dfrac{\dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}}{\mathsf{B}} \quad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\mathsf{C}}}{\mathsf{A}\Rightarrow\mathsf{C}}\,{}^a \qquad \text{corresponds with} \quad \lambda a.((pa)qa) \qquad (1)$$

where a has type A, p has type $\mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}$, and q has type $\mathsf{A}\Rightarrow\mathsf{B}$.

The λ-calculus supports 'forwards' reasoning, where we plug together complete derivations to form larger ones. However, we may wish to reason 'backwards': We start from an incomplete derivation of the desired conclusion and we work backwards to construct a derivation. Then we may have 'half a derivation', like as below left with a 'hole' called X:

$$\dfrac{\dfrac{\vdots\, X \quad \mathsf{B}}{\mathsf{C}} \quad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\mathsf{A}\Rightarrow\mathsf{C}}\,{}^a \qquad \dfrac{\dfrac{[\mathsf{A}]^a \quad \dfrac{\vdots\, X}{\mathsf{A}\Rightarrow\mathsf{B}}}{\mathsf{B}} \quad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\mathsf{C}}}{\mathsf{A}\Rightarrow\mathsf{C}}\,{}^a \qquad \dfrac{\dfrac{[\mathsf{A}]^a \quad \dfrac{\vdots\, X}{\mathsf{A}\Rightarrow\mathsf{B}}}{\mathsf{B}} \quad \dfrac{[\mathsf{A}]^a \quad \mathsf{A}\Rightarrow\mathsf{B}\Rightarrow\mathsf{C}}{\mathsf{B}\Rightarrow\mathsf{C}}}{\mathsf{C}}}{\mathsf{A}\Rightarrow\mathsf{C}}\,{}^a$$

[1] We are grateful to Jojgov for the examples in his paper [Joj02]. We thank Iman Poernomo and two anonymous referees for helpful comments.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 179–193, 2008.

Here, λ-calculus syntax is less helpful. X corresponds with qa in the complete derivation, so (being straighforward about it) the incomplete derivation corresponds with '$\lambda a.((pa)X)$'. But X is under a λ-binder and should be *instantiated*; substituted for without avoiding capture. This is impossible within the λ-calculus. Most interesting logics are undecidable so theorem-proving is often interactive (like AUTOMATH [dB80] and its many descendents). This leads us to study calculi tailored to represent incomplete derivations.

In this paper we build on previous work by the first author and others on nominal techniques [GP01] and specifically nominal terms [UPG04] and one-and-a-halfth order logic [GM07]. These were designed specifically to study binding (in unification up to α-equivalence, and derivation-schema in first-order logic respectively). They feature two levels of variables, freshness conditions, and permutations; details are in this paper, and in the work cited above. In this paper we extend this pallette of ideas to represent binding in incomplete derivations. We are reasonably ambitious in our choice of logic for which to represent incomplete derivations: we will consider first-order predicate logic; this is a significantly more complex target than propositional logic, and it leads to quite a rich syntax.

In the style of Miller [Mil92], McBride's OLEG system [McB99], and a collection of λ-calculi by Bognar [Bog02], we can represent X by fa where f is a 'normal' variable, perhaps recording in a context f should be instantiated; $f \vdash \lambda a.((pa)fa)$. A problem from our point of view is, for example, that the representation of the incomplete derivation above left is identical to that of distinct incomplete derivations above centre and above right, in which X is refined.

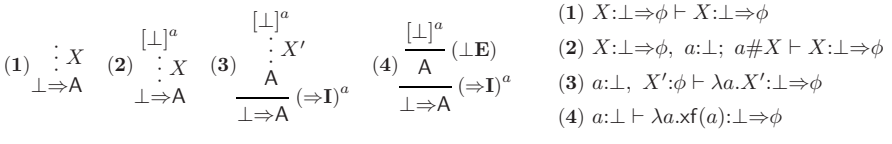
Another approach is to extend the λ-calculus with hereditarily parameterised meta-variables (hereditarily, since the parameters may themselves have 'holes'). This path is taken by Jojgov [Joj02], and for a non-hereditary notion of parameters, by Severi and Poll [SP94], and Bloo *et al* [BKLN02].

Following one-and-a-halfth order logic [GM07] we propose an approach based on *nominal terms* [UPG04]. Nominal terms have *atoms* $a, b, c, \ldots$ and *unknowns* $X, Y, Z, \ldots$. Crucially, substitution of unknowns does not avoid capture by atoms, and we reason on what unknowns do not depend on, rather than using parameters to record what they might depend on ($a\#X$ versus fa; see 'freshness' below). The first author, Urban, Pitts, and Cheney amongst others have argued in favour this approach [UPG04, FG07, Mat07, Pit02, Che05].[2]

We further this argument and show that atoms, unknowns and freshness model assumptions, holes, and discharge in incomplete derivations.

Consider an example; definitions are in the body of the paper:

[2] In one-and-a-halfth order logic, unknowns populate predicates, and model predicate meta-variables in derivation schemas. Here, unknowns are used differently, to model holes in terms representing derivations. The common 'one-and-a-halfth order' idea — also present, implicitly, in Curry-style types for nominal terms [FG06] — is that atoms can be *variables*. This is different from [UPG04] where atoms populate a sort of atoms and are variable *symbols*.



On the left is a refinement of an incomplete derivation of $\bot \Rightarrow \mathsf{A}$ to a complete derivation represented by $\lambda a.\mathsf{xf}(a)$. Here xf (for *ex-falsum*) is a constant representing $\bot$-elimination. On the right is their representation as terms-in-context in one-and-a-halfth order Curry-Howard. Note that:

– *Assumptions* are represented by atoms. Types are predicates assumed.
– *Incomplete* parts of the derivation, or (using terminology from the theorem-proving community) *subgoals*, are represented by unknowns. Types are predicates to be proved.
– *Freshness conditions* $a\#X$, read in the literature as 'a is fresh for X' [UPG04] mean here that 'a must be discharged in whatever X is instantiated to'.

This paper is 'just' about a type system for nominal terms. Has this not been done before? Not in a way that helps us for constructing Curry-Howard for first-order logic. A sorting system for nominal terms from [UPG04] is not suitable; it is designed to construct abstract syntax and atoms have sort 'the sort of atoms'. A typing system [FG06] is not suitable; types corresponded to propositional logic with quantifiers whereas here, we want first-order logic and; we also want to represent $(\forall\mathbf{I})$ and $(\forall\mathbf{E})$ (Figure 2) so terms may λ-abstract over and be applied to type variables and we require freshness for type variables.

Some words on what this paper is not: it is not proof-search [PR05, MS06]. We study binding in incomplete derivations, but not the act of stepping from one derivation to another. We also give no semantics to our syntax: There is no denotational semantics (Scott domains spring to mind; we would require an extended version, perhaps like FM (nominal) domain theory [SP05]). There is not even an operational semantics (reduction of derivations), though we do plan this for a later paper; see the Conclusions.

2 Terms, Types, and Natural Deduction

2.1 Terms and Types

We give definitions, then discuss examples in Remark 10 and Subsection 2.2.

Fix disjoint countably infinite sets of **atoms** $\mathbb{A}$ and **unknowns**. We let $a, b, c, d, \ldots$ range over atoms. We use a **permutative convention**; they range *permutatively* over atoms, so for example 'a and b' means 'a pair of two *distinct* atoms'. Similarly we let $X, Y, Z, \ldots$ range permutatively over unknowns.[3]

[3] When we write 'x' and 'y' we intend them to be distinct symbols; for example '$\lambda x.y$' is always taken to be different from '$\lambda x.x$'; likewise '$x = y$' is different syntax than '$x = x$'. This is distinct from the denotation of x being equal to that of y.

Fix **atomic type-formers** $\mathsf{P}, \mathsf{Q}, \mathsf{R}$, to each of which is associated an arity $ar(\text{-})$ which is a nonnegative integer (0, 1, 2, ...).

Definition 1. Let **types** be: $\phi, \psi, \xi ::= \bot \mid \phi \Rightarrow \phi \mid \mathsf{P}(\overbrace{a, \ldots}^{ar(\mathsf{P})\text{ var'bles}}) \mid \forall a.\phi$.

For example $\forall a.(\mathsf{P}(a,a) \Rightarrow \mathsf{P}(a,b))$ is a valid type if $ar(\mathsf{P}) = 2$.

We equate types up to $\forall$-bound atoms. We write $\equiv$ for syntactic identity of types. We write $\phi[a := b]$ for the usual capture-avoiding substitution action of b for a. Implication associates to the right; for example $\phi \Rightarrow \psi \Rightarrow \xi \equiv \phi \Rightarrow (\psi \Rightarrow \xi)$.

Intuitively, types are first-order logic with the trivial term-language (a logic whose terms are just variable symbols).

Definition 2. Define the **free atoms** of ϕ as standard by:

$$fa(\mathsf{P}(a,\ldots))=\{a,\ldots\} \quad fa(\phi \Rightarrow \psi)=fa(\phi) \cup fa(\psi) \quad fa(\bot)=\emptyset \quad fa(\forall a.\phi)=fa(\phi)\backslash\{a\}$$

Definition 3. Let **terms** be: $r, s, t, \ldots ::= a \mid X \mid \lambda a.r \mid r'r \mid \mathsf{f}(r)$.
Following [GL08] we identify terms up to α-equivalence of a in $\lambda a.r$ provided that r mentions no unknowns.[4] We write $\equiv$ for syntactic equivalence of terms.

For example $\lambda a.a \equiv \lambda b.b$ and $\lambda a.X \not\equiv \lambda b.X$. We may write $(\lambda a.r)t$ as $r[a \mapsto t]$, for example $(\lambda a.b)a \equiv b[a \mapsto a]$. We may write $r'r$ as $r'(r)$. Application associates to the left, so $r''r'r \equiv (r''r')r$; sometimes we will bracket anyway.

Definition 4. A **type assignment** is a pair of the form $a : \phi$, or $X : \phi$, or $a : *$. A **typing context** Γ is a finite set of type assignments, which is functional in the sense that:

- If $a : \phi \in \Gamma$ then $a : * \notin \Gamma$. If $a : * \in \Gamma$ then $a : \phi \notin \Gamma$.
- If $a : \phi \in \Gamma$ and $a : \phi' \in \Gamma$ then $\phi = \phi'$. Similarly for X.

As is standard we may drop set brackets, writing for example $\Gamma, a:\phi$ for $\Gamma \cup \{a:\phi\}$. We use this convention later without comment. Intuitively, $a : \phi$ means 'a has type ϕ'; $a : *$ means 'a is a type variable'; $X : \phi$ means 'X has type ϕ'.

Remark 5. We use the same syntactic class (atoms) to represent type variables and term variables. The typing context differentiates them; $a : \phi \in \Gamma$ means a behaves like a term variable; $a : * \in \Gamma$ means a behaves like a type variable.

We could make a syntactic separation between atoms that can have types ($a : \phi \in \Gamma$), and atoms that can appear in types ($a : * \in \Gamma$). However, we would duplicate the treatments of λ-abstraction, application, and freshness. Our approach keeps the machinery significantly shorter.

Definition 6. Call a pair $a\#r$ of an atom and a term a **freshness**. Call a freshness of the form $a\#X$ **primitive**. Call a finite set of primitive freshnesses a **freshness context**. Δ will range over freshness contexts.

[4] This allows us to rename bound atoms in 'normal' syntax — the part without unknowns — but it stops short of a nominal terms style α-equivalence with unknowns based on permutations. For our purposes in this paper, what we give ourselves is enough. See the Conclusions.

Definition 7. Call $\Gamma; \Delta \vdash r$ a **term-in-context**. Call $\Gamma; \Delta \vdash r : \phi$ a **typing sequent**. Call $\Gamma; \Delta \vdash a\#r$ a **freshness sequent**.

We may write '$\Gamma; \Delta \vdash r : \phi$' for '$\Gamma; \Delta \vdash r : \phi$ is a derivable typing sequent', and similarly for $\Gamma; \Delta \vdash a\#r$.

Definition 8.
– If Φ is a set of types, write $fa(\Phi)$ for $\bigcup\{fa(\phi) \mid \phi \in \Phi\}$.
– If $\mathcal{X}$ is a set of unknowns, write $a\#\mathcal{X}$ for the freshness context $\{a\#X \mid X \in \mathcal{X}\}$.
– Write $b \notin \Delta$ when $b\#X \notin \Delta$ for all X.

Definition 9. Let the **derivable** typing and freshness sequents be inductively defined by the rules in Figure 1. We use the following notation here and later:

– Side-conditions are written in brackets.
– A ranges over typings or freshnesses, so $A \in \{r : \phi,\ a : *,\ a\#r\}$.
– If a sequent - $\vdash$ - is *not* derivable we write - $\nvdash$ -.
– We write $important(\Gamma; \Delta \vdash r)$ for $\{\phi \mid a : \phi \in \Gamma,\ \Gamma; \Delta \nvdash a\#r\}$.

If ϕ exists such that $\Gamma; \Delta \vdash r : \phi$ is derivable, call $\Gamma; \Delta \vdash r$ **typable**.

Remark 10. We compare the rules in Figures 1 and 2:

– Compare (**T**$\bot$**E**) with ($\bot$**E**). 'xf' stands for *ex falsum*. (**T**$\bot$**E**) corresponds with ($\bot$**E**) in a standard way. No surprises here.
– Compare (**T**$\Rightarrow$**I**) with ($\Rightarrow$**I**). (**T**$\Rightarrow$**I**) does not discharge $a : \phi$ because r may contain an unknown X. We intend X to be instantiated to t which (because instantiation need not avoid capture) may mention a; see Definition 16. We remember $a : \phi$ in the typing context so that we can use it to build t, if we like. We can mimic ($\Rightarrow$**I**) using (**T**$\Rightarrow$**I**) and (**Tfr**).
– (**Tfr**) is an explicit discharge rule. It connects $b\#r$, which we can read as 'b will discharged in the (possibly incomplete) derivation represented by r' with actual discharge of b; after discharge, we cannot use b to construct any further derivations. As we just argued above, in the presence of unknowns it is convenient to separate these two notions.
– Compare (**T**$\forall$**I**) with ($\forall$**I**). $a \notin fa(\Phi)$ is intuitively 'a is not free in any of the assumptions Φ used to prove ϕ'. $a \notin fa(important(\Gamma, a{:}*; \Delta \vdash r))$ generalises this to take account of unknowns and freshness assumptions on them.
– Compare (**a**$\#$**b**), (**a**$\#$**b**$'$), and (**a**$\#$**b**$''$). (**a**$\#$**b**) and (**a**$\#$**b**$''$) are as in [UPG04]; distinct atoms are fresh. In (**a**$\#$**b**$'$) we account for the *type* of b. For example:

$$a : \mathsf{P}(c),\ X : \mathsf{P}(c),\ c : *;\ a\#X \vdash a\#X$$
$$a{:}\mathsf{P}(c),\ X{:}\mathsf{P}(c),\ c{:}*;\ a\#X \nvdash c\#X \qquad a{:}\mathsf{P}(c),\ X{:}\mathsf{P}(c),\ c{:}*;\ a\#X \nvdash c\#a$$

2.2 Examples

The derivations below type terms representing derivations from the Introduction; one is complete, the other incomplete. At each stage the term being typed

represents a (possibly incomplete) Natural Deduction derivation. Write '$\Gamma;\emptyset \vdash r$' as '$\Gamma \vdash r$'. Write Γ for $a : \mathsf{A},\ p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C},\ q : \mathsf{A} \Rightarrow \mathsf{B}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash a : \mathsf{A}}(\mathbf{Tax}) \quad \dfrac{}{\Gamma \vdash q : \mathsf{A} \Rightarrow \mathsf{B}}(\mathbf{Tax})}{\Gamma \vdash qa : \mathsf{B}}(\mathbf{Ty{\Rightarrow}E}) \quad \dfrac{\dfrac{}{\Gamma \vdash a : \mathsf{A}}(\mathbf{Tax}) \quad \dfrac{}{\Gamma \vdash p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C}}(\mathbf{Tax})}{\Gamma \vdash pa : \mathsf{B} \Rightarrow \mathsf{C}}(\mathbf{T{\Rightarrow}E})}{\Gamma \vdash (pa)qa : \mathsf{C}}(\mathbf{Ty{\Rightarrow}E})}{\Gamma \vdash \lambda a.((pa)qa) : \mathsf{A} \Rightarrow \mathsf{C}}(\mathbf{T{\Rightarrow}I})}{p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C},\ q : \mathsf{A} \Rightarrow \mathsf{B} \vdash \lambda a.((pa)qa) : \mathsf{A} \Rightarrow \mathsf{C}}(\mathbf{Tfr})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,\ X : \mathsf{B} \vdash X : \mathsf{B}}(\mathbf{Tax}) \quad \dfrac{\dfrac{}{\Gamma,\ X : \mathsf{B} \vdash a : \mathsf{A}}(\mathbf{Tax}) \quad \dfrac{}{\Gamma,\ X : \mathsf{B} \vdash p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C}}(\mathbf{Tax})}{\Gamma,\ X : \mathsf{B} \vdash pa : \mathsf{B} \Rightarrow \mathsf{C}}(\mathbf{T{\Rightarrow}E})}{\Gamma,\ X : \mathsf{B} \vdash (pa)X : \mathsf{C}}(\mathbf{Ty{\Rightarrow}E})}{\Gamma,\ X : \mathsf{B} \vdash \lambda a.((pa)X) : \mathsf{A} \Rightarrow \mathsf{C}}(\mathbf{T{\Rightarrow}I})}{p : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C},\ q : \mathsf{A} \Rightarrow \mathsf{B},\ X : \mathsf{B} \vdash \lambda a.((pa)X) : \mathsf{A} \Rightarrow \mathsf{C}}(\mathbf{Tfr})$$

Derivations of $\Gamma \vdash a\#\lambda a.((pa)qa)$ and $\Gamma,\ X : \mathsf{B} \vdash a\#\lambda a.((pa)X)$ are elided.[5]

Another example illustrates the side-condition on $(\mathbf{T}\forall\mathbf{I})$. The two derivations

$$\dfrac{\mathsf{A} \qquad \dfrac{\dfrac{\forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c))}{\mathsf{A} \Rightarrow \mathsf{P}(c)}(\forall\mathbf{E})}{\mathsf{P}(c)}(\Rightarrow\mathbf{E})}{\dfrac{\forall c.\mathsf{P}(c)}{\mathsf{B}}(\forall\mathbf{I})} \qquad \dfrac{\dfrac{\vdots X}{(\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B}} \quad \dfrac{}{\forall c.\mathsf{P}(c)}}{\mathsf{B}}(\Rightarrow\mathbf{E}) \tag{2}$$

are represented, writing Γ for $a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B},\ c : *$, by:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash a{:}\mathsf{A}}(\mathbf{Tax}) \quad \dfrac{\dfrac{}{\Gamma \vdash p : \forall c.(\mathsf{A}{\Rightarrow}\mathsf{P}(c))}(\mathbf{Tax})}{\Gamma \vdash pc : \mathsf{A}{\Rightarrow}\mathsf{P}(c)}(\mathbf{T\forall E})}{\Gamma \vdash pca : \mathsf{P}(c)}(\mathbf{T{\Rightarrow}E}) \quad \begin{array}{l}(c \# fa(\mathsf{A}),\\ c \# fa(\forall c.(\mathsf{A}{\Rightarrow}\mathsf{P}(c)))\end{array}}{\Gamma \vdash \lambda c.(pca) : \forall c.\mathsf{P}(c)}(\mathbf{T\forall I}) \quad \dfrac{}{\Gamma \vdash q : (\forall c.\mathsf{P}(c)){\Rightarrow}\mathsf{B}}(\mathbf{Tax})}{\Gamma \vdash q(\lambda c.(pca)) : \mathsf{B}}(\mathbf{T{\Rightarrow}E})}{a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B} \vdash q(\lambda c.(pca)) : \mathsf{B}}(\mathbf{Tfr})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,\ X : \mathsf{P}(c) \vdash X : \mathsf{P}(c)}(\mathbf{Tax}) \quad \begin{array}{l}(c \# fa(\mathsf{A})\\ c \# fa(\forall c.(\mathsf{A}{\Rightarrow}\mathsf{P}(c)))\\ c \# fa((\forall c.\mathsf{P}(c)){\Rightarrow}\mathsf{B}))\end{array}}{\Gamma,\ X : \mathsf{P}(c) \vdash \lambda c.X : \forall c.\mathsf{P}(c)}(\mathbf{T\forall I}) \quad \dfrac{}{\Gamma,\ X : \mathsf{P}(c) \vdash q : (\forall c.\mathsf{P}(c)){\Rightarrow}\mathsf{B}}(\mathbf{Tax})}{\Gamma,\ X : \mathsf{P}(c) \vdash q(\lambda c.X) : \mathsf{B}}(\mathbf{T{\Rightarrow}E})}{a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B},\ X : \mathsf{P}(c) \vdash q(\lambda c.X) : \mathsf{B}}(\mathbf{Tfr})$$

[5] The $(\mathbf{Tfr})$ in the second derivation is ill-advised, in the sense that intuitively we cannot then instantiate X to qa; we might prefer to conclude the derivation with $(\mathbf{T{\Rightarrow}I})$. We leave this kind of choice to a derivation search algorithm, or we might favour a 'safe' variant of $(\mathbf{Tfr})$ which insists r be closed (that r mention no unknowns).

Derivations of freshnesses are elided.

2.3 Natural Deduction

We outline Natural Deduction and prove forms of soundness and completeness.

Definition 11. Call a finite set of types a (Natural Deduction) **context**. Let Φ, Φ' range over contexts.

Write $\Phi \vdash \phi$ when ϕ may be derived using the rules in Figure 2 allowing elements of Φ as assumptions.[6] In accordance with our convention, side-conditions are in brackets. As is standard, square brackets in ($\Rightarrow$**I**) denote *discharge* of assumptions; note that we may choose to discharge ϕ zero times (*empty* discharge).

Lemma 12. *If $\Phi \vdash \psi$ and $\Phi \subseteq \Phi'$ then $\Phi' \vdash \psi$.*

Definition 13. Call $\Gamma; \Delta \vdash r$ **closed** when $\Delta = \emptyset$ and Γ mentions no unknowns. Recall that we write '$\Gamma; \emptyset \vdash r$' as '$\Gamma \vdash r$'.

Theorem 14. *Suppose $\Gamma \vdash r$ is closed and suppose $\Gamma \vdash r : \phi$ is derivable. Then $\mathit{important}(\Gamma \vdash r) \vdash \phi$ (Definition 9) is derivable in Natural Deduction. (Proof in the Appendix.)*

Theorem 15. *If $\Phi \vdash \phi$ is derivable in Natural Deduction then there exists some closed $\Gamma \vdash r$ such that $\mathit{important}(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$. (Proof in the Appendix.)*

2.4 Admissible Rules

Definition 16. Define a **substitution** action $r[X := t]$ by:

$$a[X := t] \equiv a \quad X[X := t] \equiv t \quad Y[X := t] \equiv Y \quad \mathsf{xf}(r)[X := t] \equiv \mathsf{xf}(r[X := t])$$
$$(\lambda a.r)[X := t] \equiv \lambda a.(r[X := t]) \quad (r'r)[X := t] \equiv (r'[X := t])(r[X := t])$$

Write $\mathit{unkn}(\Gamma)$ for the unknowns mentioned in Γ. Figure 3 presents two kinds of weakening and a form of Cut.

Theorem 17. (**WeakX**) *is **admissible** (if the sequents above the line are derivable, so are those below).*

Proof. By a routine induction on derivations.

(**Weaka**) *states* that the atom is fresh for the incomplete parts in the derivation:

Theorem 18. (**Weaka**) *is an admissible rule. (Proof in the Appendix.)*

Lemma 19 states how instantiating unknowns is sound:

[6] Note that in Natural Deduction, $\Phi, \phi \vdash \phi$ is automatic — 'if we allow Φ, ϕ as assumptions, then we assume ϕ, and so we have ϕ'. There is no need for a derivation rule.

Lemma 19. *Suppose that:*

- $\Gamma, X : \psi; \Delta \vdash r : \phi$ *and* $\Gamma; \Delta \vdash t : \psi$.
- $\Gamma; \Delta' \vdash a\#t$ *for every* $a\#X \in \Delta$.

Then:

- $\Gamma; \Delta' \vdash r[X := t] : \phi$.
- $\Gamma, X : \psi; \Delta \vdash a\#r$ *implies* $\Gamma; \Delta' \vdash a\#r[X := t]$, *for every* a.

(Proof in the Appendix.)

Cut in natural deduction is no more than 'plugging the conclusion of one derivation into the assumption(s) of another'. However, now assumptions may be holes in incomplete derivations and we can 'plug' in a capturing manner. The rule (**Cut**) specifies that operation, and from Lemma 19 we have:

Theorem 20. (**Cut**) *is an admissible rule.*

3 Derivation-Search (Sketch)

If by 'backwards' reasoning we mean 'reasoning from conclusion towards assumptions' then the machinery so far is sufficient. To mix 'forwards' with 'backwards' reasoning we may need a little more; consider an incomplete derivation of $\mathsf{A},\ \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B} \vdash \mathsf{B}$ (cf. (2) in Subsection 2.2):

$$\dfrac{\mathsf{A} \qquad \dfrac{\forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c))}{\mathsf{A} \Rightarrow \mathsf{P}(c)}}{\mathsf{P}(c)} \qquad \dfrac{(\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B}}{\vdots} \qquad (3)$$
$$\mathsf{B}$$

Our syntax does not represent this as a single term-in-context because the 'hole' is not at the leaf of the derivation. We can represent this incomplete derivation as a set of sequents, all sharing the same typing and freshness context. Following theorem-provers and unification algorithms we present this as a set of *goals*, in rewriting style; the rewrites below can easily be converted into derivation trees:

Definition 21. Let Ξ range over finite sets of typings $r : \phi$, $a : *$, and freshnesses $a\#r$. We may call $A \in \Xi$ a **goal** and we may call Ξ a **goal set**.

$A \in \Xi$ has intuition 'we know A' — not 'we want to prove A' — but if A mentions an unknown X then what we know is incomplete and we would like to complete it, i.e. prove it. To derive ϕ from $\Gamma; \Delta$ we start rewriting from $X : \phi, \Gamma, \Delta$ for X not appearing in Γ or Δ, and we try to instantiate X. We can declare success when we arrive at a goal state of the form $\Xi,\ r : \phi$ such that $\Gamma; \Delta \vdash r : \phi$.

For example to prove B from $\mathsf{A},\ \forall a.(\mathsf{A} \Rightarrow \mathsf{P}(a)),\ (\forall a.\mathsf{P}(a)) \Rightarrow \mathsf{B}$ we can start with $X : \mathsf{B},\ a : \mathsf{A},\ p : \forall a.(\mathsf{A} \Rightarrow \mathsf{P}(a)),\ q : (\forall a.\mathsf{P}(a)) \Rightarrow \mathsf{B}$ and rewrite as follows (we may drop types to save space):

$$
\begin{array}{rl}
X : \mathsf{B},\ a : \mathsf{A},\ p : \forall c.(\mathsf{A} \Rightarrow \mathsf{P}(c)),\ q : (\forall c.\mathsf{P}(c)) \Rightarrow \mathsf{B} & \\
\xrightarrow{(\mathbf{Weaka})} & X : \mathsf{B},\ a,\ p,\ q,\ c : * \\
\xrightarrow{(\mathbf{T}\forall\mathbf{E})} & X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc : \mathsf{A} \Rightarrow \mathsf{P}(c) \\
\xrightarrow{(\mathbf{T}\Rightarrow\mathbf{E})} & X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc,\ pca : \mathsf{P}(c) \\
\xrightarrow{(\mathbf{T}\forall\mathbf{I})} & X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc,\ pca,\ \lambda c.(pca) : \forall c.\mathsf{P}(c) \\
\xrightarrow{(\mathbf{T}\Rightarrow\mathbf{E})} & X : \mathsf{B},\ a,\ p,\ q,\ c,\ pc,\ pca,\ \lambda c.(pca),\ q(\lambda c.(pca)) : B \\
\xrightarrow{(\mathbf{Cut})} & q(\lambda c.(pca)) : B,\ a,\ p,\ q,\ c,\ pc,\ pca,\ \lambda c.(pca)
\end{array}
$$

We read off $q(\lambda c.(pca))$ as our result. (3) is represented by the third line above: $X : \mathsf{B},\ a,\ p,\ q,\ pc,\ pca : \mathsf{P}(c)$.

The following series of rewrites generates the derivation in (1) from the Introduction, also discussed in Subsection 2.2 ($\rightarrow^*$ is multiple rewrites):

$$
\begin{array}{rl}
X : \mathsf{A} \Rightarrow \mathsf{C},\ p : \mathsf{A} \Rightarrow \mathsf{B},\ q : \mathsf{A} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C} & \\
\xrightarrow{(\mathbf{Weaka})} & X : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a : \mathsf{A},\ a\#X \\
\xrightarrow{(\mathbf{WeakX})} & X : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ a\#X,\ X' : \mathsf{C} \\
\xrightarrow{(\mathbf{T}\Rightarrow\mathbf{I})} & X : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ a\#X,\ X',\ \lambda a.X' : \mathsf{A} \Rightarrow \mathsf{C} \\
\xrightarrow{(\mathbf{Cut})} & \lambda a.X' : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ X' \\
\xrightarrow{(\mathbf{T}\Rightarrow\mathbf{E})}{}^{*} & \lambda a.X' : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ X',\ pa : \mathsf{B},\ qa : B \Rightarrow C,\ (pa)qa : C \\
\xrightarrow{(\mathbf{Cut})} & \lambda a.((pa)qa) : \mathsf{A} \Rightarrow \mathsf{C},\ p,\ q,\ a,\ pa,\ qa,\ (pa)qa
\end{array}
$$

4 Conclusions

We have seen how nominal terms, with a typing system, can model 'incomplete derivations' in first-order logic. We use a 'one-and-a-halfth order' syntax, building on ideas from nominal terms and one-and-a-halfth order logic: atoms model variable symbols and can be quantified (we use atoms to model both type and term variables); unknowns model 'holes' in the derivation. This directly reflects informal practice, in which holes in incomplete derivations are instantiated (substituted with capture).

We have tested our system on examples. We have shown the fragment without unknowns is sound and complete with respect to 'normal' derivations (Subsection 2.3). We have shown instantiating unknowns is sound, and explored what weakening means in the presence of the two levels of variable (Subsection 2.4).

This paper is part of a larger project which we expect to be a fruitful source of research. In roughly decreasing order of certainty, we envisage the following:

Curry-Howard supposes normalisation of derivations — this translates to an operational semantics for terms (Definition 3). This has to be more than 'remove

all β-reducts' because, for example, the β-reduct in $(\lambda a.X)Y$ cannot be reduced. To address this, an investigation into *two-and-a-halfth order λ-calculus* is ongoing. This has λa and also a λX, substitution for X does not avoid capture by λa, and nominal terms style α-equivalence. This paper would then be a rather powerful type system (more than Hindley-Milner for example) for the λX-free fragment of two-and-a-halfth order λ-calculus; we are reasonably confident this would extend to λX.

The rewrite system alluded to in Section 3 can be viewed as an independent system and studied. On that topic, we can ask whether the ideas in this paper can be useful for the theory or practice of writing theorem provers. Perhaps the representation itself will be useful, but nominal unification is known to be decidable [UPG04]; thus, nominal terms have some good computational properties which we may be able to exploit. Given the scale and complexity of modern theorem-provers, answers to such questions may take some time to emerge — but the situation is also far from hopeless, since in the first instance only the prover's 'kernel' is involved.

Indeed, we can simplify the types to propositional logic (simple types; we drop the predicate part) and attempt to develop the rewrite system into a unification algorithm *à la* Huet [Hue02]. We can also try to enrich types in the direction of a dependent type theory and attempt to develop the typing rules from Figure 1 into a dependent type theory *på samma* Martin-Löf [NPS90]. This would be distinct from a dependent type theory with elements of nominal techniques [SS04], which treats atoms as variable symbols.

References

[BKLN02] Bloo, R., Kamareddine, F., Laan, T., Nederpelt, R.: Parameters in pure type systems. In: Rajsbaum, S. (ed.) LATIN 2002. LNCS, vol. 2286, pp. 371–385. Springer, Heidelberg (2002)

[Bog02] Bognar, M.: Contexts in Lambda Calculus. PhD thesis, Vrije Universiteit Amsterdam (2002)

[Che05] Cheney, J.: Nominal logic and abstract syntax. SIGACT News (logic column 14) 36(4), 47–69 (2005)

[dB80] de Bruijn, N.G.: A survey of the project AUTOMATH. In: Hindley, Seldin (eds.) To H.B.Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism. Academic Press, London (1980)

[FG06] Fernández, M., Gabbay, M.J.: Curry-style types for nominal rewriting. In: Altenkirch, T., McBride, C. (eds.) TYPES 2006. LNCS, vol. 4502. Springer, Heidelberg (2007)

[FG07] Fernández, M., Gabbay, M.J.: Nominal rewriting. Information and Computation 205, 917–965 (2007)

[GL08] Gabbay, M.J., Lengrand, S.: The lambda-context calculus. ENTCS 196, 19–35 (2008)

[GM07] Gabbay, M.J., Mathijssen, A.: One-and-a-halfth-order logic. Journal of Logic and Computation (November 2007)

[GP01] Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. Formal Aspects of Computing 13(3–5), 341–363 (2001)

[Hue02] Huet, G.: Higher order unification 30 years later. In: Carreño, V.A., Muñoz, C.A., Tahar, S. (eds.) TPHOLs 2002. LNCS, vol. 2410, pp. 3–12. Springer, Heidelberg (2002)

[Joj02] Jojgov, G.I.: Holes with binding power. In: Geuvers, H., Wiedijk, F. (eds.) TYPES 2002. LNCS, vol. 2646, pp. 162–181. Springer, Heidelberg (2003)

[Mat07] Mathijssen, A.: Logical Calculi for Reasoning with Binding. PhD thesis, Technische Universiteit Eindhoven (2007)

[McB99] McBride, C.: Dependently Typed Functional Programs and their Proofs. PhD thesis, University of Edinburgh (1999)

[Mil92] Miller, D.: Unification under a mixed prefix. Journal of Symbolic Computation 14(4), 321–358 (1992)

[MS06] Miller, D., Saurin, A.: A game semantics for proof search: preliminary results. In: MFPS XXI. ENTCS, vol. 155, pp. 543–563 (2006)

[NPS90] Nordstrom, B., Petersson, K., Smith, J.M.: Programming in Martin-Lof's Type Theory. Int'l Series of Monographs on Computer Science, vol. 7. Clarendon Press, Oxford (1990)

[PCW05] Poernomo, I.H., Crossley, J.N., Wirsing, M.: Adapting Proofs-as-Programs: The Curry–Howard Protocol. Monographs in Computer Science, vol. XII (2005)

[Pit02] Pitts, A.M.: Equivariant syntax and semantics. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 32–36. Springer, Heidelberg (2002)

[PR05] Pym, D.J., Ritter, E.: A games semantics for reductive logic and proof-search. In: GALOP (Games for Logic and Programming Languages), pp. 107–123 (2005)

[SP94] Severi, P., Poll, E.: Pure type systems with definitions. In: Matiyasevich, Y.V., Nerode, A. (eds.) LFCS 1994. LNCS, vol. 813, pp. 316–328. Springer, Heidelberg (1994)

[SP05] Shinwell, M.R., Pitts, A.M.: On a monadic semantics for freshness. Theoretical Computer Science 342(1), 28–55 (2005)

[SS04] Schöpp, U., Stark, I.: A Dependent Type Theory with Names and Binding. In: Marcinkowski, J., Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 235–249. Springer, Heidelberg (2004)

[UPG04] Urban, C., Pitts, A.M., Gabbay, M.J.: Nominal unification. Theoretical Computer Science 323(1–3), 473–497 (2004)

[US06] Urzyczyn, P., Sørensen, M.: Lectures on the Curry-Howard isomorphism. Studies in Logic, vol. 149. Elsevier, Amsterdam (2006)

A Technical Appendix

We will use the following fact without comment:

Lemma 22. *If $\Gamma \vdash r$ is closed (so Γ mentions no unknowns and the freshness context is empty) and $\Gamma \vdash r$ is typable, then r mentions no unknowns.*

Proof (of Theorem 14). By induction on the derivation of $\Gamma \vdash r : \phi$.

– The case of (**Tax**). Suppose $\Gamma, a : \phi \vdash a : \phi$.
It is easy to calculate that $\mathit{important}(\Gamma, a : \phi \vdash a) = \{\phi\}$; then $\phi \vdash \phi$ is a fact (see Footnote 6).

– The case of (**T**⇒**I**). Suppose $\Gamma, a : \phi \vdash \lambda a.s : \phi \Rightarrow \psi$ and $\Gamma, a : \phi \vdash s : \psi$.
By inductive hypothesis $important(\Gamma, a : \phi \vdash s) \vdash \psi$. We use (⇒**I**).
If $important(\Gamma, a : \phi \vdash \lambda a.s) = important(\Gamma, a : \phi \vdash s) \setminus \{\phi\}$ then we discharge ϕ.
If $important(\Gamma, a : \phi \vdash \lambda a.s) = important(\Gamma, a : \phi \vdash s)$ then we discharge ϕ zero times. There are no other possibilities.
– The case of (**T**⇒**E**). Suppose $\Gamma \vdash r'r : \psi$ and $\Gamma \vdash r' : \phi \Rightarrow \psi$ and $\Gamma \vdash r : \phi$.
By inductive hypothesis $important(\Gamma \vdash r') \vdash \phi \Rightarrow \psi$ and $important(\Gamma \vdash r) \vdash \phi$.
By Lemma 12 and (**T**⇒**E**),

$$important(\Gamma \vdash r') \cup important(\Gamma \vdash r) \vdash \psi.$$

By the syntax-directed nature of the freshness rules in Figure 1, $\Gamma \vdash a\#r'r$ if and only if both of $\Gamma \vdash a\#r'$ and $\Gamma \vdash a\#r$ hold. Therefore,

$$important(\Gamma \vdash r'r) = important(\Gamma \vdash r') \cup important(\Gamma \vdash r)$$

The result follows.
– The case of (**T**∀**I**). Suppose $\Gamma, a : * \vdash \lambda a.r : \forall a.\phi$, where

$$\Gamma, a : * \vdash r : \phi \quad \text{and} \quad a \notin fa(important(\Gamma, a : * \vdash r)).$$

By inductive hypothesis $important(\Gamma, a : * \vdash r) \vdash \phi$. By (∀**I**),

$$important(\Gamma, a : * \vdash r) \vdash \forall a.\phi.$$

– The case of (**T**∀**E**). Suppose $\Gamma, b : * \vdash rb : \phi[a := b]$ and $\Gamma, b : * \vdash r : \forall a.\phi$.
By inductive hypothesis $important(\Gamma, b : * \vdash r) \vdash \forall a.\phi$. By (∀**E**),

$$important(\Gamma, b : * \vdash r) \vdash \phi[a := b].$$

By reasoning similar to the case of (**T**⇒**E**) we can calculate that

$$important(\Gamma, b : * \vdash rb) = important(\Gamma, b : * \vdash r) \cup important(\Gamma, b : * \vdash b : *).$$

Now it is a fact that $important(\Gamma, b : * \vdash b : *) = \emptyset$. The result follows.
– The case of (**Tfr**). Suppose $\Gamma \vdash r : \phi$, and

$$\Gamma, A \vdash r : \phi \text{ and } \Gamma, A \vdash b\#r \text{ where } A \in \{b : \phi, b : *\}.$$

By inductive hypothesis $important(\Gamma, A \vdash r) \vdash \phi$.
If $A = b : *$ then

$$important(\Gamma, A \vdash r) = important(\Gamma \vdash r)$$

and the result follows immediately. If $A = b : \psi$ then since $\Gamma, A \vdash b\#r$ again

$$important(\Gamma, A \vdash r) = important(\Gamma \vdash r).$$

The result follows.

By abuse of appendices, we place the proof of Theorem 18 *before* that of Theorem 15. There is no circularity in the proofs and it is convenient for brevity; the special case of Theorem 18 when Γ mentions no unknowns, is needed to prove Theorem 15.

Proof (of Theorem 18). By induction on derivations. Only the case of (**T**$\forall$**I**) is of interest.

– Suppose $\Gamma, a : *; \Delta \vdash r : \phi$ and $a \notin fa(important(\Gamma, a : *; \Delta \vdash r))$. By inductive hypothesis $\Gamma, a : *, B; \Delta, b\#\mathcal{X} \vdash r : \phi$ where $\mathcal{X} = unkn(\Gamma)$ and $b \notin \Gamma, a : *$. It is not hard to calculate that $\Gamma, a : *, B; \Delta, b\#\mathcal{X} \vdash b\#r$ and so

$$a \notin fa(important(\Gamma, a : *, B; \Delta, b\#\mathcal{X} \vdash r)).$$

The result follows.

Proof (of Theorem 15). We prove by induction on the derivation of $\Phi \vdash \phi$ that there exists some closed typable $\Gamma \vdash r$ such that:

– $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$.

– Γ satisfies a *uniqueness* property: if $a : \phi \in \Gamma$ and $x : \phi \in \Gamma$ then $x = a$ (so there is at most one atom of each type in Γ).[7]

We consider each possible rule in turn:

– The case of no rule; $\Phi \vdash \phi$ because $\phi \in \Phi$.
Suppose $fa(\phi) = \{b_1, \ldots, b_n\}$. We take $\Gamma = a : \phi,\ b_1 : *, \ldots, b_n : *$ and $r \equiv a$.

– The case (⇒**I**) Suppose $\Phi \vdash \phi \Rightarrow \psi$ and $\Phi,\ \phi \vdash \psi$.
By inductive hypothesis there exists $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi \cup \{\phi\}$ and $\Gamma \vdash r : \psi$.
If $a : \phi \in \Gamma$ for some a then let $\Gamma' = \Gamma$. If $a : \phi \in \Gamma$ for no a then let $\Gamma' = \Gamma, a : \phi$ for some a not appearing in Γ. By Theorem 18 (see the comment preceding this proof) $\Gamma' \vdash r : \phi$. It is also a fact that $important(\Gamma \vdash r) = important(\Gamma' \vdash r)$.

By (**T**⇒**I**) we have $\Gamma' \vdash \lambda a.r : \phi \Rightarrow \psi$. It is a fact that $\Gamma' \vdash a\#\lambda a.r$. Therefore by uniqueness, $important(\Gamma' \vdash \lambda a.r) = important(\Gamma' \vdash r) \setminus \{\phi\}$ The result follows.

– The case (⇒**E**). Suppose $\Phi \vdash \psi$ and $\Phi \vdash \phi \Rightarrow \psi$ and $\Phi \vdash \phi$.
By inductive hypothesis there exist:

- $\Gamma' \vdash r'$ such that $important(\Gamma' \vdash r') \subseteq \Phi$ and $\Gamma' \vdash r' : \phi \Rightarrow \psi$.
- $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$.

Without loss of generality we may assume that $\Gamma \cup \Gamma'$ satisfies our uniqueness condition; we rename atoms to make this true if necessary.
We use (**T**⇒**E**) and the fact that $important(\Gamma \cup \Gamma' \vdash r'r) \subseteq \Phi$.

– The case ($\forall$**I**). Suppose $\Phi \vdash \forall a.\phi$ where $a \notin fa(\Phi)$ and $\Phi \vdash \phi$.
By inductive hypothesis there exists $\Gamma \vdash r$ such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \phi$. Since $a \notin fa(\Phi)$ we know that $a \notin fa(important(\Gamma \vdash r))$.
If $a : * \in \Gamma$ then let $\Gamma' = \Gamma$. If $a : \xi \in \Gamma$ for some type ξ then we are in the pathological situation that $a \notin fa(\phi)$ and $a : \xi \in \Gamma$ 'by mistake'; we rename a. If $a : * \notin \Gamma$ then let $\Gamma' = \Gamma, a : *$. By Theorem 18 $\Gamma' \vdash r : \phi$. It is also a fact that $important(\Gamma \vdash r) = important(\Gamma' \vdash r)$.
We use (**T**$\forall$**I**) and the fact that $important(\Gamma' \vdash r) = important(\Gamma' \vdash \lambda a.r)$.

– The case ($\forall$**E**). Suppose $\Phi \vdash \phi[a := b]$ and $\Phi \vdash \forall a.\phi$.

[7] Here x ranges over all atoms, not necessarily permutatively, so perhaps $x = a$.

By inductive hypothesis there are Γ and r such that $important(\Gamma \vdash r) \subseteq \Phi$ and $\Gamma \vdash r : \forall a.\phi$.
If $b : * \in \Gamma$ then let $\Gamma' = \Gamma$. If $b : * \notin \Gamma$ then let $\Gamma' = \Gamma, b : *$. By Theorem 18 $\Gamma' \vdash r : \forall a.\phi$. It is also a fact that $important(\Gamma \vdash r) = important(\Gamma' \vdash r)$.

We use (**T**$\forall$**E**) and the fact that

$$important(\Gamma' \vdash r) = important(\Gamma' \vdash rb).$$

Lemma 23. *Suppose that:*

– $\Gamma, X : \psi; \Delta \vdash r : \phi$ *and* $\Gamma; \Delta \vdash t : \psi$.
– $\Gamma; \Delta' \vdash a\#t$ *for every* $a\#X \in \Delta$.

Then $important(\Gamma; \Delta' \vdash r[X := t]) \subseteq important(\Gamma, X : \psi; \Delta \vdash r)$.

Proof. By a routine induction on r. We consider cases:

– The case of a. Easy.
– The case of X. We calculate:

$$important(\Gamma, X : \psi; \Delta \vdash X) = \{\phi \mid a : \phi \in \Gamma,\ a\#X \notin \Delta\}$$

By assumption $important(\Gamma; \Delta' \vdash t) \subseteq \{\phi \mid a : \phi \in \Gamma,\ a\#X \notin \Delta\}$.

Other cases are easier.

Proof (of Lemma 19). By induction on the derivation of $\Gamma, X : \psi; \Delta \vdash r : \phi$. The first part is routine, we consider only two cases:

– The case (**a**#**b**′). Suppose $\Gamma, a : *, b : \phi', X : \psi; \Delta \vdash a\#b$ is derived by (**a**#**b**′). Then $a \notin fa(\phi')$ and it follows that

$$\Gamma, a : *, b : \phi'; \Delta' \vdash a\#b$$

Since $b[X := t] \equiv b$, we are done.

– The case (**a**#λ**b**). Suppose $\Gamma, X : \psi; \Delta \vdash a\#r$ and $\Gamma, X : \psi; \Delta \vdash a\#\lambda b.r$ is derived by (**a**#λ**b**). By inductive hypothesis $\Gamma; \Delta' \vdash a\#r[X := t]$ and so

$$\Gamma; \Delta \vdash a\#\lambda b.(r[X := t])$$

Since $\lambda b.(r[X := t]) \equiv (\lambda b.r)[X := t]$, we are done.

For the second part we consider some cases:

– The case (**Tax**). Suppose $Y : \xi \in \Gamma$ and $\Gamma, X : \psi; \Delta \vdash Y : \xi$ is derived by (**Tax**). Then $\Gamma; \Delta' \vdash Y : \xi$. Since $Y \equiv Y[X := t]$, we are done.
Similarly for $\Gamma, X : \psi; \Delta \vdash a : \xi$.
Suppose $\Gamma, X : \psi; \Delta \vdash X : \psi$ is derived by (**Tax**). By assumption $\Gamma; \Delta' \vdash t : \psi$. Since $t \equiv X[X := t]$, we are done.
– The case (**T**$\Rightarrow$**I**). Since $(\lambda a.r)[X := t] \equiv \lambda a.(r[X := t])$. The cases of (**T**$\Rightarrow$**E**) and (**T**$\forall$**E**) are similar.
– The case (**T**$\forall$**I**). Since $(\lambda a.r)[X := t] \equiv \lambda a.(r[X := t])$ and from Lemma 23.
– The case (**Tfr**).

Suppose $\Gamma,\ A,\ X{:}\psi; \Delta \vdash r : \phi$ because $\Gamma,\ A,\ X{:}\psi;\ \Delta,\ b\#\mathcal{X} \vdash r : \phi$ and suppose $\Gamma,\ A,\ X{:}\psi;\ \Delta, b\#\mathcal{X} \vdash b\#r$, where $b \notin \Delta$ and A is $b : *$ or $b : \xi$ for some ξ.

By inductive hypothesis and some calculations, $\Gamma,\ A;\ \Delta',\ b\#\mathcal{X}' \vdash r[X := t] : \phi$ and $\Gamma,\ A;\ \Delta',\ b\#\mathcal{X}' \vdash b\#r[X := t]$, for a suitable $\mathcal{X}'$ and Δ' where $b \notin \Delta'$. The result follows.

$$\frac{(A \in \{a : \phi,\ a : *,\ X : \phi\})}{\Gamma,\ A;\ \Delta \vdash A}\ (\mathbf{Tax}) \qquad \frac{\Gamma; \Delta \vdash r : \bot}{\Gamma; \Delta \vdash \mathsf{xf}(r) : \phi}\ (\mathbf{T}\bot\mathbf{E})$$

$$\frac{\Gamma,\ a : \phi;\ \Delta \vdash r : \psi}{\Gamma,\ a : \phi;\ \Delta \vdash \lambda a.r : \phi \Rightarrow \psi}\ (\mathbf{T}\Rightarrow\mathbf{I}) \qquad \frac{\Gamma;\ \Delta \vdash r' : \phi \Rightarrow \psi \quad \Gamma;\ \Delta \vdash r : \phi}{\Gamma;\ \Delta \vdash r'r : \psi}\ (\mathbf{T}\Rightarrow\mathbf{E})$$

$$\frac{\Gamma,\ a : *;\ \Delta \vdash r : \phi \quad a \notin fa(important(\Gamma,\ a : *; \Delta \vdash r))}{\Gamma,\ a : *;\ \Delta \vdash \lambda a.r : \forall a.\phi}\ (\mathbf{T}\forall\mathbf{I})$$

$$\frac{\Gamma,\ b : *;\ \Delta \vdash r : \forall a.\phi}{\Gamma,\ b : *;\ \Delta \vdash rb : \phi[a := b]}\ (\mathbf{T}\forall\mathbf{E})$$

$$\frac{\Gamma,\ A;\ \Delta, b\#\mathcal{X} \vdash r : \phi \quad \Gamma,\ A;\ \Delta, b\#\mathcal{X} \vdash b\#r \quad (A \in \{b : \psi,\ b : *\},\ b \notin \Delta)}{\Gamma; \Delta \vdash r : \phi}\ (\mathbf{Tfr})$$

$$\frac{}{\Gamma,\ a : \phi,\ b : \phi; \Delta \vdash a\#b}\ (\mathbf{a}\#\mathbf{b}) \qquad \frac{}{\Gamma,\ a : *,\ b : *; \Delta \vdash a\#b}\ (\mathbf{a}\#\mathbf{b}'')$$

$$\frac{}{\Gamma; \Delta \vdash a\#\lambda a.r}\ (\mathbf{a}\#\lambda\mathbf{a}) \qquad \frac{\Gamma; \Delta \vdash a\#r}{\Gamma; \Delta \vdash a\#\lambda b.r}\ (\mathbf{a}\#\lambda\mathbf{b}) \qquad \frac{}{\Gamma; \Delta,\ a\#X \vdash a\#X}\ (\mathbf{a}\#\mathbf{X})$$

$$\frac{(a \notin fa(\phi))}{\Gamma,\ a : *,\ b : \phi; \Delta \vdash a\#b}\ (\mathbf{a}\#\mathbf{b}') \qquad \frac{\Gamma; \Delta \vdash a\#r' \quad \Gamma; \Delta \vdash a\#r}{\Gamma; \Delta \vdash a\#r'r}\ (\mathbf{a}\#\mathbf{app}) \qquad \frac{\Gamma; \Delta \vdash a\#r}{\Gamma; \Delta \vdash a\#\mathsf{xf}(r)}\ (\mathbf{a}\#\mathbf{xf})$$

Fig. 1. Typing and freshness derivation rules

$$\frac{\bot}{\phi}\ (\bot\mathbf{E}) \qquad \frac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi \Rightarrow \psi}\ (\Rightarrow\mathbf{I}) \qquad \frac{\phi \Rightarrow \psi \quad \phi}{\psi}\ (\Rightarrow\mathbf{E}) \qquad \frac{\begin{array}{c}\Phi\\ \vdots\\ \phi \quad (a \notin fa(\Phi))\end{array}}{\forall a.\phi}\ (\forall\mathbf{I}) \qquad \frac{\forall a.\phi}{\phi[a := b]}\ (\forall\mathbf{E})$$

Fig. 2. Natural Deduction style derivation rules

$$\frac{\Gamma; \Delta \vdash r : \phi \quad (X \notin \Gamma)}{\Gamma,\ X : \psi; \Delta \vdash r : \phi}\ (\mathbf{WeakX}) \qquad \frac{\Gamma; \Delta \vdash r : \phi \quad (B \in \{b : \psi,\ b : *\},\ a \notin \Gamma)}{\Gamma,\ B; \Delta, b\#unkn(\Gamma) \vdash r : \phi}\ (\mathbf{Weaka})$$

$$\frac{\Gamma, X{:}\psi; \Delta, a_1\#X, \ldots, a_n\#X \vdash r : \phi \quad \Gamma; \Delta \vdash t{:}\psi \quad \Gamma; \Delta \vdash a_i\#t\ (1 \leq i \leq n) \quad (X \notin \Delta)}{\Gamma; \Delta \vdash r[X := t] : \phi}\ (\mathbf{Cut})$$

Fig. 3. Admissible rules

Labelled Calculi for Łukasiewicz Logics

D. Galmiche and Y. Salhi

LORIA – UHP Nancy 1
Campus Scientifique, BP 239
54 506 Vandœuvre-lès-Nancy, France

Abstract. In this paper, we define new decision procedures for Łukasiewicz logics. They are based on particular integer-labelled hypersequents and of logical proof rules for such hypersequents. These rules being proved strongly invertible our procedures naturally allow one to generate countermodels. From these results we define a "merge"-free calculus for the infinite version of Łukasiewicz logic and prove that it satisfies the sub-formula property. Finally we also propose for this logic a new terminating calculus by using a focusing technique.

1 Introduction

Łukasiewicz logics, including finite and infinite versions, are among the most studied many-valued logics [10] and the infinite version Ł is, like Gödel-Dummett logic (LC) and Product logic (Π), one of the fundamental *t-norm based* fuzzy logics [8]. There exist various calculi and methods dedicated to proof-search in these logics that are based on sequents [1,12], hypersequents [4,12] or relational hypersequents [3] and on tableaux [13] or goal-directed approach [11]. In this paper, we consider proof-search in propositional Łukasiewicz logics through a particular approach that consists firstly in reducing (by a proof-search process) a hypersequent into a set of so-called irreducible hypersequents and then secondly in deciding these specific hypersequents by a particular procedure. Such an approach has been studied for Gödel-Dummett logic [2] and also the infinite version Ł of Łukasiewicz logics [3] but not for the finite versions. In this context we are interested in deciding irreducible hypersequents through a countermodel search process and thus in providing new decision procedures that generate countermodels.

Therefore we define labelled hypersequents, called $\mathbb{Z}$-hypersequents, in which components are labelled with integers, such labels introducing semantic information in the search process. Then we define proof rules that deal with labels by using the addition and subtraction and then prove that they are strongly invertible. It is important to notice that we define a same set of simple proof rules for both finite and infinite versions of Łukasiewicz logic. By application of these rules we show how we can reduce the decision problem of every $\mathbb{Z}$-hypersequent to the decision problem of a set of so-called atomic $\mathbb{Z}$-hypersequents that only contain atomic formulae. To solve the later problem we associate a set of particular inequalities to these hypersequents and then strongly relate the existence of a countermodel to the existence of a solution for this set of inequalities. Thus, by using results from linear and integer programming [16], we can decide any atomic $\mathbb{Z}$-hypersequent and also generate a countermodel in case of non-validity. Thus, from the same set of rules, we provide a new decision procedure for the

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 194–207, 2008.

infinite version but also one for the finite versions of Łukasiewicz logic, both including countermodel generation. After this first contribution we focus, in the rest of the paper, on the infinite version denoted Ł. The next contribution is the definition of a new calculus for this logic that is characterized by a single form of axioms and the absence of the "merge" rule that is not appropriate for proof-search. In addition our labelling of components by integers can be seen as a kind of merge-elimination technique that could be applied to hypersequent calculi given in [4,12]. From a refinement of the notion $\mathbb{Z}$-hypersequent, by using a focusing technique defined in [12], the last contribution is a terminating calculus for Ł, that is proved sound and complete, in which only one rule is not (strongly) invertible. We complete these results by showing, in the appendix, how to obtain a labelled calculus for Bounded Łukasiewicz logics $\text{Ł}\mathsf{B}_\mathsf{n}$ with $n \geqslant 2$ [4].

2 Łukasiewicz Logics

We consider here the family of Łukasiewicz logics denoted Ł_n with $n \in \mathbb{N}^1 = \{2, \ldots\} \cup \{\infty\}$, set of natural numbers with its natural order $\leqslant$, augmented with a greatest element ∞. In the case $n = \infty$, Ł_∞, also denoted by Ł, is one of the most interesting multi-valued logics and one of the fundamental t-norms based fuzzy logics (see [8] for more details). In the case $n \neq \infty$, Ł_n denotes the finite versions of Łukasiewicz logics.

The set of propositional formulae, denoted Form, is inductively defined from a set of propositional variables with a bottom constant $\bot$ (absurdity) by using the connectives $\wedge, \vee$, $\odot$ (strong conjunction) and $\oplus$ (strong disjunction). All the connectives can be expressed by using the $\supset$ connective: $\neg A =_{def} A \supset \bot$, $A \oplus B =_{def} \neg A \supset B$, $A \odot B =_{def} \neg(A \supset \neg B)$, $A \vee B =_{def} (A \supset B) \supset B$ and $A \wedge B =_{def} \neg(\neg A \vee \neg B)$.

In the case of Ł, the logic has a following Hilbert axiomatic system:

Ł1 $A \supset (B \supset A)$
Ł2 $(A \supset B) \supset ((B \supset C) \supset (A \supset C))$
Ł3 $((A \supset B) \supset B) \supset ((B \supset A) \supset A)$
Ł4 $((A \supset \bot) \supset (B \supset \bot)) \supset (B \supset A)$

with the rule $\dfrac{A \supset B \qquad A}{B}$ $[mp]$

Another Hilbert axiomatic system can be obtained by adding axioms Ł1 and Ł3 to any axiomatization of the multiplicative additive fragment of Linear Logic [14].

For the finite versions Ł_n with $n \neq \infty$, a Hilbert axiomatic system is obtained by adding to the previous axioms of Ł the following axioms : $(n-1)A \supset nA \odot nA \supset (n-1)A$ and $(pA^{p-1})^n \supset mA^p \odot mA^p \supset (pA^{p-1})^n$ for every integer $p = 2, \ldots, n-2$ that does not divide $n-1$, with $kA = A \oplus \ldots \oplus A$ (k times) and $A^k = A \odot \ldots \odot A$ (k times).

A valuation for Ł_n is a function $[\![\cdot]\!]$ from the set of propositional variables Var to $[0,1]$ if $n = \infty$ and to $[0, 1/(n-1), \ldots, (n-2)/(n-1), 1]$ if $n \neq \infty$. It is inductively extended to formulae as follows:

$[\![A \supset B]\!] = min(1, 1 - [\![A]\!] + [\![B]\!])$
$[\![\bot]\!] = 0$
$[\![\neg A]\!] = 1 - [\![A]\!]$
$[\![A \odot B]\!] = max(0, [\![A]\!] + [\![B]\!] - 1)$
$[\![A \oplus B]\!] = min(1, [\![A]\!] + [\![B]\!])$
$[\![A \wedge B]\!] = min([\![A]\!], [\![B]\!])$
$[\![A \vee B]\!] = max([\![A]\!], [\![B]\!])$

A formula A is valid in Ł_n, written $\models_{\text{Ł}_\mathsf{n}} A$, iff $[\![A]\!] = 1$ for all valuations $[\![\cdot]\!]$ for Ł_n.

In this paper we study proof-search in the finite and infinite versions of Łukasiewicz logics. Our approach based on labelled calculi is an alternative to existing works based on sequents [1,12], on multisets of sequents, called hypersequents [4,12] and relational hypersequents [3] but also on tableaux [13] or goal-directed approach [11]. It consists first in reducing (by a proof-search process) a hypersequent into a set of so-called irreducible hypersequents and then in deciding these hypersequents. It has been studied for LC [2] and also the infinite version Ł [3] but not for the finite versions. Like for recent works in Gödel-Dummett Logics [7,9] we aim at deciding irreducible hypersequents through a countermodel search process and then at providing new calculi and decision procedures that allow us to generate countermodels.

3 Labelled Proof Rules for $Ł_n$

In this section, we present for $Ł_n$ the definition of integer-labelled hypersequents, labels introducing semantic information in the search process, and of labelled proof rules that are strongly invertible in order to generate countermodels. Let us remind that the hypersequent structure $\Gamma_1 \vdash \Delta_1 \mid \ldots \mid \Gamma_k \vdash \Delta_k$ has been introduced as a natural generalization of Gentzen's sequents [2]. It is a multiset of sequents, called components, with "|" denoting a disjunction at the meta-level.

Definition 1. *A* $\mathbb{Z}$-hypersequent *is a hypersequent of the form:* $\Gamma_1 \vdash_{n_1} \Delta_1 \mid \ldots \mid \Gamma_k \vdash_{n_k} \Delta_k$ *where for* $i = 1, \ldots, k$, $n_i \in \mathbb{Z}$, *and* Γ_i *and* Δ_i *are multisets of formulae.*

Definition 2. *A* $\mathbb{Z}$*-hypersequent* $G = \Gamma_1 \vdash_{n_1} \Delta_1 \mid \ldots \mid \Gamma_k \vdash_{n_k} \Delta_k$ *is* valid *in* $Ł_n$ *iff for any valution* $[\![\cdot]\!]$ *for* $Ł_n$, *there exists* $i \in \{1, \ldots, n\}$ *such that:* $\llbracket \Gamma_i \rrbracket \leqslant \lceil \Delta_i \rceil - n_i$ *where* $\llbracket \emptyset \rrbracket = 1$, $\lceil \emptyset \rceil = 0$, $\llbracket \Gamma_i \rrbracket = 1 + \sum_{A \in \Gamma_i} ([\![A]\!] - 1)$ *and* $\lceil \Delta_i \rceil = \sum_{B \in \Delta_i} [\![B]\!]$.

A formula A is valid in $Ł_n$ if and only if the $\mathbb{Z}$-hypersequent $\vdash_0 A$ is valid in $Ł_n$. Moreover the $\mathbb{Z}$-hypersequent $A^1_1, \ldots, A^1_{l_1} \vdash_0 B^1_1, \ldots, B^1_{m_1} \mid \ldots \mid A^k_1, \ldots, A^k_{l_1} \vdash_0 B^k_1, \ldots, B^k_{m_k}$ is valid in $Ł_n$ if and only if $(A^1_1 \odot \ldots \odot A^1_{l_1}) \supset (B^1_1 \oplus \ldots \oplus B^1_{m_1}) \vee \ldots \vee (A^k_1 \odot \ldots \odot A^k_{l_1}) \supset (B^k_1 \oplus \ldots \oplus B^k_{m_k})$ is valid in $Ł_n$.

In comparison with hypersequents in [4,12] where the interpretation of components is such that one has disjunctions ($\oplus$) on the both sides, our aim here is to recover the standard interpretation with conjunctions ($\odot$) on the left-hand side and disjunctions ($\oplus$) on the right-hand side.

Now we define a set of proof rules, presented in Figure 1, dealing with these structures. They mainly decompose the principal formula and simply modify the labels by addition or substraction of 1.

Considering a proof rule as composed of premises H_i with a conclusion C, it is *sound* if, for any instance of the rule, the validity of the premises H_i entails the validity of C. It is *strongly sound* if, for any instance of the rule and any valuation $[\![\cdot]\!]$, if $[\![\cdot]\!]$ is a model of all the H_i then it is a model of C. Moreover a proof rule is *invertible* if, for any instance of the rule, the non-validity of at least one H_i entails the non-validity of C. It is *strongly invertible* if, for any instance of the rule and any valuation $[\![\cdot]\!]$, if $[\![\cdot]\!]$ is a countermodel of at least one H_i then it is a countermodel of C. We can observe that strong invertibility (resp. soundness) implies invertibility (resp. soundness).

$$\frac{G \mid \Gamma, A, B \vdash_n \Delta \qquad G \mid \Gamma \vdash_{n-1} \Delta}{G \mid \Gamma, A \odot B \vdash_n \Delta}\ [\odot_L] \qquad \frac{G \mid \Gamma \vdash_n \Delta \mid \Gamma \vdash_{n+1} A, B, \Delta}{G \mid \Gamma \vdash_n A \odot B, \Delta}\ [\odot_R]$$

$$\frac{G \mid \Gamma \vdash_n \Delta \mid \Gamma, A, B \vdash_{n+1} \Delta}{G \mid \Gamma, A \oplus B \vdash_n \Delta}\ [\oplus_L] \qquad \frac{G \mid \Gamma \vdash_n A, B, \Delta \qquad G \mid \Gamma \vdash_{n-1} \Delta}{G \mid \Gamma \vdash_n A \oplus B, \Delta}\ [\oplus_R]$$

$$\frac{G \mid \Gamma \vdash_n \Delta \mid \Gamma, B \vdash_{n+1} A, \Delta}{G \mid \Gamma, A \supset B \vdash_n \Delta}\ [\supset_L] \qquad \frac{G \mid \Gamma, A \vdash_n B, \Delta \qquad G \mid \Gamma \vdash_{n-1} \Delta}{G \mid \Gamma \vdash_n A \supset B, \Delta}\ [\supset_R]$$

$$\frac{G \mid \Gamma, A \vdash_n \Delta \mid \Gamma, B \vdash_n \Delta}{G \mid \Gamma, (A \wedge B) \vdash_n \Delta}\ [\wedge_L] \qquad \frac{G \mid \Gamma \vdash_n A, \Delta \qquad G \mid \Gamma \vdash_n B, \Delta}{G \mid \Gamma \vdash_n A \wedge B, \Delta}\ [\wedge_R]$$

$$\frac{G \mid \Gamma, A \vdash_n \Delta \qquad G \mid \Gamma, B \vdash_n \Delta}{G \mid \Gamma, A \vee B \vdash_n \Delta}\ [\vee_L] \qquad \frac{G \mid \Gamma \vdash_n A, \Delta \mid \Gamma \vdash_n B, \Delta}{G \mid \Gamma \vdash_n A \vee B, \Delta}\ [\vee_R]$$

Fig. 1. Proof rules for $\mathbb{Z}$-hypersequents in $\mathbf{Ł_n}$

Theorem 1 (Soundness). *The rules of Figure 1 are strongly sound for* $\mathbf{Ł_n}$.

Proof. We only develop the cases of $[\supset_L]$ and $[\oplus_R]$ rules, the other cases being similar.

Case $[\supset_L]$. Let $[\![\cdot]\!]$ be a model of $G \mid \Gamma \vdash_k \Delta \mid \Gamma, B \vdash_{k+1} A, \Delta$ in $\mathbf{Ł_n}$. Then we have $[\![\cdot]\!]$ is a model of G, $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor \leqslant \lceil\!\lceil \Delta \rceil\!\rceil - k$ or $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor + ([\![B]\!] - 1) \leqslant \lceil\!\lceil \Delta \rceil\!\rceil + [\![A]\!] - (k+1)$. Thus, we obtain $[\![\cdot]\!]$ is a model of G, $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor + (1-1) \leqslant \lceil\!\lceil \Delta \rceil\!\rceil - k$ or $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor + ((1 - [\![A]\!] + [\![B]\!]) - 1) \leqslant \lceil\!\lceil \Delta \rceil\!\rceil - k$. We deduce that $[\![\cdot]\!]$ is a model of G or $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor + (min(1, 1 - [\![A]\!] + [\![B]\!]) - 1) \leqslant \lceil\!\lceil \Delta \rceil\!\rceil - k$. Therefore $[\![\cdot]\!]$ is a model of $G \mid \Gamma, A \supset B \vdash_n \Delta$.

Case $[\oplus_R]$. Let $[\![\cdot]\!]$ be a model of $G \mid \Gamma \vdash_k A, B, \Delta$ and of $G \mid \Gamma \vdash_{k-1} \Delta$ in $\mathbf{Ł_n}$. Thus, $[\![\cdot]\!]$ is a model of G, or $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor \leqslant \lceil\!\lceil \Delta \rceil\!\rceil + [\![A]\!] + [\![B]\!] - k$ and $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor \leqslant \lceil\!\lceil \Delta \rceil\!\rceil - (k-1)$ hold. Then $[\![\cdot]\!]$ is a model of G or the inequality $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor \leqslant \lceil\!\lceil \Delta \rceil\!\rceil + min(1, [\![A]\!] + [\![B]\!]) - k$ holds. Thus, $[\![\cdot]\!]$ is a model of $G \mid \Gamma \vdash_n A \oplus B, \Delta$.

Theorem 2. *The rules of Figure 1 are strongly invertible for* $\mathbf{Ł_n}$.

Proof. We only develop the cases of $[\supset_L]$ and $[\oplus_R]$ rules, the other cases being similar.

Case $[\supset_L]$. Let $[\![\cdot]\!]$ be a countermodel of $G \mid \Gamma \vdash_k \Delta \mid \Gamma, B \vdash_{k+1} A, \Delta$ in $\mathbf{Ł_n}$. Then $[\![\cdot]\!]$ is a countermodel of G and the inequalities $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor > \lceil\!\lceil \Delta \rceil\!\rceil - k$ or $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor + ([\![B]\!] - 1) > \lceil\!\lceil \Delta \rceil\!\rceil + [\![A]\!] - (k+1)$ hold. Therefore, $[\![\cdot]\!]$ is a countermodel of G and the inequality $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor + (min(1, 1 - [\![A]\!] + [\![B]\!]) - 1) > \lceil\!\lceil \Delta \rceil\!\rceil - k$ holds. We deduce that $[\![\cdot]\!]$ is a countermodel of $G \mid \Gamma, A \supset B \vdash_n \Delta$.

Case $[\oplus_R]$. Let $[\![\cdot]\!]$ be a countermodel of $G \mid \Gamma \vdash_k A, B, \Delta$ in $\mathbf{Ł_n}$. Then we have $[\![\cdot]\!]$ is a countermodel of G and $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor > \lceil\!\lceil \Delta \rceil\!\rceil + [\![A]\!] + [\![B]\!] - k$. Thus, the inequality $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor > \lceil\!\lceil \Delta \rceil\!\rceil + min(1, [\![A]\!] + [\![B]\!]) - k$ holds. Therefore, $[\![\cdot]\!]$ is a countermodel of $G \mid \Gamma \vdash_n A \oplus B, \Delta$. Let $[\![\cdot]\!]$ be a countermodel of $G \mid \Gamma \vdash_{k-1} \Delta$. Then $[\![\cdot]\!]$ is a countermodel of G and $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor > \lceil\!\lceil \Delta \rceil\!\rceil - (k-1)$ holds. Thus, $[\![\cdot]\!]$ is a countermodel of G and $\lfloor\!\lfloor \Gamma \rfloor\!\rfloor > \lceil\!\lceil \Delta \rceil\!\rceil + min(1, [\![A]\!] + [\![B]\!]) - k$ holds. Then we deduce that $[\![\cdot]\!]$ is a countermodel of $G \mid \Gamma \vdash_n A \oplus B, \Delta$.

Having proved these properties we now define what an atomic $\mathbb{Z}$-hypersequent is and show that we can reduce any $\mathbb{Z}$-hypersequent $\mathcal{H}$ into a set $\mathcal{S}$ of atomic $\mathbb{Z}$-hypersequents, such that $\mathcal{H}$ is valid iff the elements of $\mathcal{S}$ are valid.

Definition 3. *An* atomic $\mathbb{Z}$-hypersequent *is a $\mathbb{Z}$-hypersequent which only contains atomic formulae.*

Theorem 3. *The application of the rules of Figure 1 to a given $\mathbb{Z}$-hypersequent terminates with atomic $\mathbb{Z}$-hypersequents.*

Proof. To prove the termination, we show that for every rule, its conclusion is more complex than its premises by using a measures of complexity over the formulae [6]. This measure, called α, is defined by: $\alpha(A) = 1$ where $(A \in \mathsf{Var} \cup \{\top, \bot\})$; $\alpha(A \circledast B) = \alpha(A) + \alpha(B) + 1$ where $\circledast \in \{\wedge, \vee, \supset, \oplus, \odot\}$; and $\alpha(\neg A) = \alpha(A) + 1$. We can see that the order relation $<$ on formulae, defined by $A < B$ iff $\alpha(A) < \alpha(B)$, is well-founded. Let Γ_1 and Γ_2 two multisets of formulae, we have $\Gamma_1 >_m \Gamma_2$ iff Γ_2 is obtained form Γ_1 by replacing a formula by a finite number of formulae, each in which is of lower measure than the replaced formula. Since the relation order on pure formulae and sentences is well-founded, the order relation $>_m$ is well-founded, for more details [5]. Similarly, we define a well-founded relation $>>_m$ on $\mathbb{Z}$-hypersequents, induced by the order relation $>_m$, by: $G_1 >>_m G_2$ iff G_2 is obtained form G_1 by replacing a component of $G1$ by a smaller finite set of components, where a component $\Gamma_2 \vdash_{n_2} \Delta_2$ is smaller than $\Gamma_1 \vdash_{n_1} \Delta_1$ iff $\Gamma_1 \cup \Delta_1 >_m \Gamma_2 \cup \Delta_2$. By using this order relation, it is easy to prove for every rule, its premises are smaller than its conclusion. Finally, there is always a rule for any sequent which is not atomic. Therefore, we deduce that the application of our rules to a given $\mathbb{Z}$-hypersequent terminates with atomic $\mathbb{Z}$-hypersequents.

4 New Decision Procedures for Ł$_\mathsf{n}$

By using Theorem 3 we can generate, from a given $\mathbb{Z}$-hypersequent, to a set of atomic $\mathbb{Z}$-hypersequents by application of our logical rules. After this step of bottom-up proof-search we now consider the resulting set of atomic $\mathbb{Z}$-hypersequents in the perspective of countermodel generation. For respectively Ł and Ł$_n$ with $n \neq \infty$, we associate to each atomic $\mathbb{Z}$-hypersequent a set of particular inequalities and then relate the existence of a countermodel to the existence of solution for this set.

Definition 4 **($SI_{\mathcal{H}}$).** *Let $\mathcal{H} = \Gamma_1 \vdash_{n_1} \Delta_1 \mid \ldots \mid \Gamma_k \vdash_{n_k} \Delta_k$ be an atomic $\mathbb{Z}$-hypersequent and x_p be a real variable associated to every propositional variable p. We define the set of inequalities $SI_{\mathcal{H}}$ associated to $\mathcal{H}$ by: $SI_{\mathcal{H}} = \{(\bigodot \Gamma_1) > (\bigoplus \Delta_1) - n_1, \ldots, (\bigodot \Gamma_k) > (\bigoplus \Delta_k) - n_k\}$ where $\bigodot \emptyset = 1$, $\bigoplus \emptyset = 0$, $\bigodot(\Gamma_i) = 1 + \sum_{A \in \Gamma_i} (x_A - 1)$ and $\bigoplus(\Delta_i) = \sum_{A \in \Delta_i} x_A$ with $x_\bot = 0$.*

Theorem 4. *An atomic $\mathbb{Z}$-hypersequent $\mathcal{H}$ has a countermodel in Ł iff $SI_{\mathcal{H}}$ has a solution over $[0, 1]$.*

Definition 5 **($SI^n_{\mathcal{H}}$).** *Let $\mathcal{H} = \Gamma_1 \vdash_{m_1} \Delta_1 \mid \ldots \mid \Gamma_k \vdash_{m_k} \Delta_k$ be an atomic $\mathbb{Z}$-hypersequent and x_p be a real variable associated to every propositional variable p. We define the set of inequalities $SI^n_{\mathcal{H}}$ associated to $\mathcal{H}$ by: $SI^n_{\mathcal{H}} = \{(\bigodot_n \Gamma_1) - 1 \geq (\bigoplus_n \Delta_1) - ((n-1) * m_1), \ldots, (\bigodot_n \Gamma_k) - 1 \geq (\bigoplus_n \Delta_k) - ((n-1) * m_k)\}$, where $\bigodot_n \emptyset = n - 1$, $\bigoplus_n \emptyset = 0$, $\bigodot_n(\Gamma_i) = (n-1) + \sum_{A \in \Gamma_i} (x_A - (n-1))$ and $\bigoplus_n(\Delta_i) = \sum_{A \in \Delta_i} x_A$ where $x_\bot = 0$.*

Theorem 5. *An atomic $\mathbb{Z}$-hypersequent H has a countermodel in $\mathbf{Ł}_n$ with $n \neq \infty$ iff $SI^n_{\mathcal{H}}$ has a solution over the set of integers $\{0,\ldots,n-1\}$.*

The proofs of the above theorems are given in appendix B.

By using linear and integer programming [16], we can decide a $\mathbf{Ł}_n$ atomic $\mathbb{Z}$-hypersequent in polynomial time. If $(x_{A_1} = r_1, \ldots, x_{A_k} = r_k)$ is a solution of the set $SI^n_{\mathcal{H}}$ (resp. $SI_{\mathcal{H}}$), where $\{x_{A_1},\ldots,x_{A_k}\}$ is the set of all its variables, then the valuation $[\![\cdot]\!]$ such that $\forall i \in \{1,\ldots,k\}$, $[\![A_i]\!] = r_i/(n-1)$ (resp. $[\![A_i]\!] = r_i$) is a countermodel of $\mathcal{H}$ in $\mathbf{Ł}_n$ (resp. in Ł). For a given $\mathbb{Z}$-hypersequent, by Theorem 3 we can generate a set of atomic $\mathbb{Z}$-hypersequents by application of rules of Figure 1. Then we can build the set $SI_{\mathcal{H}}$ (resp. $SI^n_{\mathcal{H}}$) associated to each atomic $\mathbb{Z}$-hypersequent H and decide by using linear (resp. integer) programming if it has a countermodel or not and thus decide its validity in Ł (resp. $\mathbf{Ł}_n$ with $n \neq \infty$).

These two main steps, namely proof search followed by countermodel search (based on the above theorems) provide new decision procedures for Łukasiewicz logics. A key point here is the generation of countermodels because of the strong invertibility of rules: any countermodel of an atomic $\mathbb{Z}$-hypersequent on the leaf of the derivation tree is a countermodel of the initial $\mathbb{Z}$-hypersequent on the root of this tree.

We illustrate our new procedure through examples. If we consider $\mathcal{H}_1 = \vdash_0 A \supset (B \supset A)$ and $\mathcal{H}_2 = \vdash_0 A \vee (A \supset \bot)$, by application of proof rules we obtain the derivations:

$$\dfrac{\dfrac{\dfrac{A, B \vdash_0 A \quad A \vdash_{-1}}{A \vdash_0 B \supset A}\,[\supset_R] \qquad \vdash_{-1}}{\vdash_0 A \supset (B \supset A)}\,[\supset_R]}{} \qquad \dfrac{\dfrac{\dfrac{\vdash_0 A \mid A \vdash_0 \bot \quad \vdash_0 A \mid \vdash_{-1}}{\vdash_0 A \mid \vdash_0 A \supset \bot}\,[\supset_R]}{\vdash_0 A \vee (A \supset \bot)}\,[\vee_R]}{}$$

Thus, $\mathcal{H}_1$ has a countermodel in Ł if one of the inequalities $1 > 1$ ($\vdash_{-1}$), $x_A > 1$ ($A \vdash_{-1}$) and $x_B > 1$ ($A, B \vdash_0 A$) has a solution over $[0,1]$. Since $1 > 1$, $x_A > 1$ and $x_B > 1$ have no solution over $[0,1]$, we deduce that $\mathcal{H}_1$ is valid in Ł. For $\mathcal{H}_2$, since $x_A = 1$ is an integer solution of the system $\{2 > x_A, x_A > 0\}$, the valuation $[\![\cdot]\!]$ defined by $[\![A]\!] = \frac{1}{2}$ is a countermodel of $\vdash_0 A \mid A \vdash_0 \bot$ in $\mathbf{Ł}_3$. Then it is a countermodel of $\mathcal{H}_2$ in $\mathbf{Ł}_3$.

5 The $\mathbb{Z}$Ł Calculus

In this section we propose a new calculus for Ł called $\mathbb{Z}$Ł, that is defined by the rules in Figure 1 and the following axiom, special rules and structural rules:

$$\frac{}{\vdash_n}\;[Ax](n<0) \qquad \frac{G \mid \Gamma \vdash_{n-1} \Delta}{G \mid \Gamma, \bot \vdash_n \Delta}\;[\bot_L] \qquad \frac{G \mid \Gamma \vdash_{n-1} \Delta}{G \mid \Gamma, A \vdash_n \Delta, A}\;[SR]$$

$$\frac{G}{G \mid \Gamma \vdash_n \Delta}\;[EW] \qquad \frac{G \mid \Gamma \vdash_n \Delta}{G \mid \Gamma, A \vdash_n \Delta}\;[IW_L] \qquad \frac{G \mid \Gamma \vdash_n \Delta}{G \mid \Gamma \vdash_n \Delta, A}\;[IW_R]$$

$$\frac{G \mid \Gamma \vdash_n \Delta \mid \Gamma \vdash_n \Delta}{G \mid \Gamma \vdash_n \Delta}\;[EC] \qquad \frac{G \mid \Gamma_1, \Gamma_2 \vdash_{n_1+n_2+1} \Delta_1, \Delta_2}{G \mid \Gamma_1 \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2}\;[SP]$$

Theorem 6 (Soundness). *The rules of the $\mathbb{Z}$Ł calculus are sound.*

Proof. From Theorem 1 the logical rules of $\mathbb{Z}$Ł are sound. Similar arguments are used for the other rules.

Theorem 7 (Completeness). *If a $\mathbb{Z}$-hypersequent is valid in Ł then it is derivable in the $\mathbb{Z}$Ł calculus.*

Proof. See appendix B.

We illustrate our calculus by considering our example $\mathcal{H}_1 = \vdash_0 A \supset (B \supset A)$. By application of proof rules we obtain the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash_{-1}}{B\vdash_{-1}}[IW_L]}{A,B\vdash_0 A}[SR] \quad \dfrac{\vdash_{-1}}{A\vdash_{-1}}[IW_L]}{A\vdash_0 B\supset A}[\supset_R] \qquad \vdash_{-1}}{\vdash_0 A\supset(B\supset A)}[\supset_R]$$

From the $\mathbb{Z}$Ł calculus we can show that the weakening rules ($[EW]$, $[IW_L]$ and $[IW_R]$) can be *"absorbed"* in the axiom by using an approach similar to the one of [17]. Thus we obtain a new simplified calculus $\mathbb{Z}$Ł$'$ without these rules and with the following axiom: $\dfrac{}{G \mid \Gamma \vdash_n \Delta}\ [Ax](n < 0)$.

Proposition 1. *The $\mathbb{Z}$Ł calculus satisfies the subformula property, namely any formula appearing in a proof of $\mathcal{H}$ in $\mathbb{Z}$Ł is a subformula of a formula in $\mathcal{H}$.*

An important point of these calculi is that they are "merge"-free. It means that the following rule, $\dfrac{G \mid \Gamma_1 \vdash \Delta_1 \qquad G \mid \Gamma_2 \vdash \Delta_2}{G \mid \Gamma_1,\Gamma_2 \vdash \Delta_1,\Delta_2}\ [M]$ called *merge*, is not needed.

In hypersequent calculi for Ł in [4,3] a challenge, in the perspective of proof-search, consists in eliminating this rule that is not appropriate because it is not invertible and context splittings on the left and right sides could be very expensive. The rule has been eliminated in [15] by replacing the existing axioms by the following axiom $G \mid \Gamma, \overbrace{\bot,\ldots,\bot}^{n} \vdash A_1,\ldots,A_n,\Delta$, where $n \geqslant 0$. But our approach based on the labelling of components by integers allows us to eliminate the merge rule without having to complicate the form of axioms.

6 A Terminating Calculus for Ł

Now, we consider an approach based on a focusing technique in [12] in order to provide a terminating calculus for Ł. Thus we consider now so-called *focused hypersequents*.

Definition 6. *A* focused $\mathbb{Z}$-hypersequent *is a structure of the form $[p]\mathcal{H}$ where $\mathcal{H}$ is a $\mathbb{Z}$-hypersequent, p a propositional variable, and $[p]\mathcal{H}$ is valid in Ł iff $\mathcal{H}$ is valid in Ł.*

Let $\mathcal{H} = \Gamma_1 \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2 \mid \ldots \mid \Gamma_k \vdash_{n_k} \Delta_k$ be a $\mathbb{Z}$-hypersequent. We denote by $left(\mathcal{H})$ the multiset $\Gamma_1 \cup \Gamma_2 \cup \ldots \cup \Gamma_k$ and by $right(\mathcal{H})$ the multiset $\Delta_1 \cup \Delta_2 \cup \ldots \cup \Delta_k$. We define a new calculus, called $\mathbb{Z}\text{Ł}\mathbf{T}$, that consists of the logical rules in Figure 1 with the same focus for premises and conclusion, and of these following rules:

$$\frac{}{[p]G \mid \Gamma \vdash_n \Delta}\ [Ax](n<0) \qquad \frac{[p]G \mid \Gamma \vdash_{n-1} \Delta}{[p]G \mid \Gamma, \bot \vdash_n \Delta}\ [\bot_L] \qquad \frac{[p]G \mid \Gamma \vdash_{n-1} \Delta}{[p]G \mid \Gamma, A \vdash_n \Delta, A}\ [SR]$$

$$\frac{[q]\mathcal{H}}{[p]\mathcal{H}}\ [F] \quad \text{where } q \in left(\mathcal{H}) \cap right(\mathcal{H}) \text{ and } p \notin left(G) \cap right(G)$$

$$\frac{[p]G \mid k_2\Gamma_1, k_1\Gamma_2 \vdash_{n'} k_2\Delta_1, k_1\Delta_2 \mid S}{[p]G \mid \Gamma_1, k_1 p \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2, k_2 p}\ [R]$$

where $G, \Gamma_1, \Gamma_2, \Delta_1$ and Δ_2 are atomic and $k_1 > 0, k_2 > 0, p \notin \Gamma_1 \cup \Gamma_2 \cup \Delta_1 \cup \Delta_2$.
S is $\Gamma_1, k_1 p \vdash_{n_1} \Delta_1$ or $\Gamma_2 \vdash_{n_2} \Delta_2, k_2 p$ and $n' = k_2 * n_1 + k_1 * n_2 + k_1 + k_2 - (k_1 * k_2 + 1)$.

Theorem 8. *All the rules of $\mathbb{Z}\text{Ł}\mathbf{T}$ except $[R]$ are strongly invertible.*

Proof. From Theorem 2, the logical rules of $\mathbb{Z}\text{Ł}\mathbf{T}$ are strongly invertible. For the other rules we use similar arguments.

Definition 7. *An* irreducible focused $\mathbb{Z}$-hypersequent $[p]\mathcal{H}$ *is an atomic focused $\mathbb{Z}$-hypersequent where $left(\mathcal{H}) \cap right(\mathcal{H}) = \emptyset$, $\bot \notin left(\mathcal{H})$ and for every component $\Gamma \vdash_n \Delta$ of $\mathcal{H}$, we have $n \geqslant 0$.*

Definition 8. *An* inv-irreducible focused $\mathbb{Z}$-hypersequent $[p]\mathcal{H}$ *is an atomic $\mathbb{Z}$-hypersequent where $p \in left(\mathcal{H}) \cap right(\mathcal{H})$, and for every component $\Gamma \vdash_n \Delta$ of $\mathcal{H}$, we have $\bot \notin \Gamma$, $\Gamma \cap \Delta = \emptyset$ and $n \geqslant 0$.*

Proposition 2. *Any irreducible focused $\mathbb{Z}$-hypersequent has a countermodel.*

Proof. Let $[p]\mathcal{H}$ be an irreducible focused $\mathbb{Z}$-hypersequent. Let $[\![\cdot]\!]$ a valuation defined by: for every $A \in left(\mathcal{H})$ we have $[\![A]\!] = 1$, and for every $B \in right(\mathcal{H})$ we have $[\![B]\!] = 0$. It is easy to prove that $[\![\cdot]\!]$ is a countermodel of $[p]\mathcal{H}$.

Theorem 9. *The application of $\mathbb{Z}\text{Ł}\mathbf{T}$ calculus to every focused $\mathbb{Z}$-hypersequent terminates with axioms or irreducible focused $\mathbb{Z}$-hypersequents.*

Proof. From Theorem 3, we see that the application of the logical rules of $\mathbb{Z}\text{Ł}\mathbf{T}$ to a given focused $\mathbb{Z}$-hypersequent terminates with atomic focused $\mathbb{Z}$-hypersequents. By using the order $>>_m$ defined in the proof of Theorem 3, $G \mid \Gamma, \bot \vdash_n \Delta >>_m G \mid \Gamma \vdash_{n-1} \Delta$ and $G \mid \Gamma, A \vdash_n \Delta, A >>_m G \mid \Gamma \vdash_{n-1} \Delta$ hold. Now considering the rule $[R]$, we can see that its application with the focus p decreases strictly the number of p's. Therefore, in any derivation in $\mathbb{Z}\text{Ł}\mathbf{T}$, the number of applications of the rules $[R]$ and $[F]$ is finite. Thus, the application of $\mathbb{Z}\text{Ł}\mathbf{T}$ calculus to every focused $\mathbb{Z}$-hypersequent terminates.

Since there is always a rule for any $\mathbb{Z}$-hypersequent which is not an axiom or an irreducible $\mathbb{Z}$-hypersequent, we deduce that The application of $\mathbb{Z}\text{Ł}\mathbf{T}$ calculus to every focused $\mathbb{Z}$-hypersequent terminates with axioms or irreducible focused $\mathbb{Z}$-hypersequents.

Theorem 10 (Soundness). *The rules of $\mathbb{Z}\text{Ł}\mathbf{T}$ are sound.*

Proof. The soundness of the logical rules and the rules $[Ax]$, $[\bot_L]$ and $[SR]$ comes from Theorem 6. The soundness of $[F]$ is trivial. For the rule $[R]$ we consider arguments similar to those of proof of Theorem 1.

Proposition 3. *If the atomic $\mathbb{Z}$-hypersequent $G \mid \Gamma_1 \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2$ is valid in Ł then either $G \mid \Gamma_1, \Gamma_2 \vdash_{n_1+n_2+1} \Delta_1, \Delta_2 \mid \Gamma_1 \vdash_{n_1} \Delta_1$ is valid in Ł or $G \mid \Gamma_1, \Gamma_2 \vdash_{n_1+n_2+1} \Delta_1, \Delta_2 \mid \Gamma_2 \vdash_{n_2} \Delta_2$ is valid in Ł.*

Proposition 4. *Let $[p]G \mid \Gamma_1, k_1 p \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2, k_2 p$ be an atomic focused $\mathbb{Z}$-hypersequent. If it is valid in Ł then either $[p]G \mid k_2\Gamma_1, k_1\Gamma_2 \vdash_{n'} k_2\Delta_1, k_1\Delta_2 \mid \Gamma_1, k_1 p \vdash_{n_1} \Delta_1$ is valid in Ł or $[p]G \mid k_2\Gamma_1, k_1\Gamma_2 \vdash_{n'} k_2\Delta_1, k_1\Delta_2 \mid \Gamma_2 \vdash_{n_2} \Delta_2, k_2 p$ is valid in Ł, with $k_1 > 0, k_2 > 0, p \notin \Gamma_1 \cup \Gamma_2 \cup \Delta_1 \cup \Delta_2$ and $n' = k_2 * n_1 + k_1 * n_2 + k_1 + k_2 - (k_1 * k_2 + 1)$.*

Proofs of these propositions are given in appendix A.

Definition 9 (Proof-refutation tree). *A* proof-refutation tree *is a tree where the nodes are labelled by a focused $\mathbb{Z}$-hypersequents and satisfying the following properties:*

- *Every internal node n labelled by $\mathcal{H}$ which is not an inv-irreducible $\mathbb{Z}$-hypersequent has a maximum of two children: if n has two children (resp. a single child) labelled by $\mathcal{H}_1$ and $\mathcal{H}_2$ (resp. $\mathcal{H}'$) then $\dfrac{\mathcal{H}_1 \qquad \mathcal{H}_2}{\mathcal{H}}\ [r]$ (resp. $\dfrac{\mathcal{H}'}{\mathcal{H}}\ [r]$) is an instance of a strongly invertible rule.*
- *Every internal node n labelled by $\mathcal{H}$ which is an inv-irreducible $\mathbb{Z}$-hypersequent, namely $[p]G \mid \Gamma_1, k_1 p \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2, k_2 p$, has two children labelled by $[p]G \mid k_2\Gamma_1, k_1\Gamma_2 \vdash_{n'} k_2\Delta_1, k_1\Delta_2 \mid \Gamma_1, k_1 p \vdash_{n_1} \Delta_1$ and by $[p]G \mid k_2\Gamma_1, k_1\Gamma_2 \vdash_{n'} k_2\Delta_1, k_1\Delta_2 \mid \Gamma_2 \vdash_{n_2} \Delta_2, k_2 p$ where $n' = k_2 * n_1 + k_1 * n_2 + k_1 + k_2 - (k_1 * k_2 + 1)$.*

From Theorem 9, we can see that a proof-refutation tree is finite and its leaf nodes are indexed by axioms and irreducible $\mathbb{Z}$-hypersequents.

Theorem 11 (Completeness). *If $[p]\mathcal{H}$ is valid in Ł then $[p]\mathcal{H}$ is provable in $\mathbb{Z}\text{Ł}\mathbf{T}$.*

Proof. Let $[p]\mathcal{H}$ be a focus $\mathbb{Z}$-hypersequent and $\mathcal{P}$ its proof-refutation tree. We show how to decide if an index of a given node in $\mathcal{P}$ is valid or not. We start by the leaf nodes. From Theorem 9, we know that such leaf nodes are labelled by axioms or irreducible focused $\mathbb{Z}$-hypersequents. Thus, by using Proposition 2, we can decide all the leaf nodes. Now we see how, from the children of a given internal node, we can propagate validity or invalidity. Let $\mathcal{H}$ be an index of internal node. If $\mathcal{H}$ is not an inv-irreducible focused $\mathbb{Z}$-hypersequent then, from Definition 9, this node has a maximum of two children where if these children are labelled by $\mathcal{H}_1$ and $\mathcal{H}_2$ (resp. $\mathcal{H}'$) then $\dfrac{\mathcal{H}_1 \qquad \mathcal{H}_2}{\mathcal{H}}\ [r]$ (resp. $\dfrac{\mathcal{H}'}{\mathcal{H}}\ [r]$) is an instance of a strongly invertible rule. Thus, if $\mathcal{H}_1$ and $\mathcal{H}_2$ (resp. $\mathcal{H}'$) are valid then $\mathcal{H}$ is valid because $[r]$ is sound. Else, from the strong invertibility of $[r]$, $\mathcal{H}$ has the same countermodels of its non-valid premises.

We now deal with the nodes labelled by inv-irreducible focused $\mathbb{Z}$-hypersequent. Let n be an internal node labelled by an inv-irreducible $\mathbb{Z}$-hypersequent $\mathcal{H}$. Thus $\mathcal{H}$ is of the form $[p]G \mid \Gamma_1, k_1 p \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2, k_2 p$. and the children of n are labelled by $[p]G \mid k_2\Gamma_1, k_1\Gamma_2 \vdash_{n'} k_2\Delta_1, k_1\Delta_2 \mid \Gamma_1, k_1 p \vdash_{n_1} \Delta_1$ and $[p]G \mid k_2\Gamma_1, k_1\Gamma_2 \vdash_{n'} k_2\Delta_1, k_1\Delta_2 \mid \Gamma_2 \vdash_{n_2} \Delta_2, k_2 p$ where $n' = k_2 * n_1 + k_1 * n_2 + k_1 + k_2 - (k_1 * k_2 + 1)$. By using Proposition 4, if one of the indexes of the children of n is valid then $\mathcal{H}$ is valid else $\mathcal{H}$ is not valid. Therefore, if a focused $\mathbb{Z}$-hypersequent is valid then it is derivable in $\mathbb{Z}\text{Ł}\mathbf{T}$.

In the completeness proof (Theorem 11) we give a decision procedure for Ł based on the concept of proof-refutation tree. Let $\mathcal{H} = \vdash_0 A \supset B \vee B \supset A$. A proof-refutation tree of $\mathcal{H}$ is given by:

$$\dfrac{\dfrac{\dfrac{\dfrac{[A]\vdash_{-1} \mid A \vdash_0 B}{[A]B \vdash_0 B \mid B \vdash_0 A}[SR] \quad \dfrac{[A]\vdash_{-1} \mid A \vdash_0 B}{[A]B \vdash_0 B \mid A \vdash_0 B}[SR]}{[A]A \vdash_0 B \mid B \vdash_0 A}[R] \qquad [A]\vdash_{-1} \mid B \vdash_0 A}{[A] \vdash_0 A \supset B \mid B \vdash_0 A}[\supset_R] \qquad [A] \vdash_0 A \supset B \mid \vdash_{-1}}{[A] \vdash_0 A \supset B \mid \vdash_0 B \supset A}[\supset_R]}{[A] \vdash_0 A \supset B \vee B \supset A}[\vee_R]$$

From this proof refutation tree, we then deduce that $\mathcal{H}$ is valid.

Our method based on proof-refutation trees cannot be applied to the terminating calculus in [12] because the merge and weakening rules are not invertible. Our terminating calculus that does not contain these rules is then more efficient because all its rules except one are (strongly) invertible: the conclusion of an invertible rule is valid iff its premises are valid.

7 Conclusion and Perspectives

In this work, we provide new decision procedures with countermodel generation for Łukasiewicz logics, using the approach proposed in [2]. A key point is the use of strongly invertible rules and consequently the ability to generate countermodels. An important contribution is the definition of a new terminating calculi for the infinite version Ł. In comparison with the calculi based on hypersequents [3,4] our calculus improves proof-search because it has a single form of axiom and moreover does not contain the merge rule. In further works we will define such labelled terminating calculi for the finite versions of Łukasiewicz logics and also for Bounded Łukasiewicz logics (see preliminary results in appendix C) for which cut-elimination will be studied. We will also study the possible design of labelled systems for other fuzzy logics.

References

1. Aguzzoli, S., Ciabattoni, A.: Finiteness in Infinite-Valued Lukasiewicz Logic. Journal of Logic, Language and Information 9(1), 5–29 (2000)
2. Avron, A.: A Tableau System for Gödel-Dummett Logic based on a Hypersequent Calculus. In: Dyckhoff, R. (ed.) TABLEAUX 2000. LNCS, vol. 1847, pp. 98–111. Springer, Heidelberg (2000)

3. Ciabattoni, A., Fermüller, C., Metcalfe, G.: Uniform Rules and Dialogue Games for Fuzzy Logics. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS (LNAI), vol. 3452, pp. 496–510. Springer, Heidelberg (2005)
4. Ciabattoni, A., Metcalfe, G.: Bounded Lukasiewicz Logics. In: Cialdea Mayer, M., Pirri, F. (eds.) TABLEAUX 2003. LNCS, vol. 2796, pp. 32–47. Springer, Heidelberg (2003)
5. Derschowitz, N., Manna, Z.: Proving termination with multiset ordering. Communications of ACM 22, 465–479 (1979)
6. Dyckhoff, R.: Contraction-free sequent calculi for intuitionistic logic. Journal of Symbolic Logic 57, 795–807 (1992)
7. Galmiche, D., Larchey-Wendling, D., Salhi, Y.: Provability and countermodels in Gödel-Dummett logics. In: Int. Workshop on Disproving: Non-theorems, Non-validity, Non-Provability, DISPROVING 2007, Bremen, Germany, July 2007, pp. 35–52 (2007)
8. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer Academic Publishers, Dordrecht (1998)
9. Larchey-Wendling, D.: Counter-model search in Gödel-Dummett logics. In: Basin, D., Rusinowitch, M. (eds.) IJCAR 2004. LNCS (LNAI), vol. 3097, pp. 274–288. Springer, Heidelberg (2004)
10. Lukasiewicz, J., Tarski, A.: Untersuchungen über den Aussagenkalkül. In: Comptes Rendus des Séances de la Societé des Sciences et des Lettre de Varsovie, Classe III, vol. 23 (1930)
11. Metcalfe, G., Olivetti, N., Gabbay, D.: Lukasiewicz Logic: From Proof Systems To Logic Programming. Logic Journal of the IGPL 13(5), 561–585 (2005)
12. Metcalfe, G., Olivetti, N., Gabbay, D.: Sequent and hypersequent calculi for Abelian and Lukasiewicz logics. ACM Trans. Comput. Log. 6(3), 578–613 (2005)
13. Olivetti, N.: Tableaux for Lukasiewicz Infinite-valued Logic. Studia Logica 73(1), 81–111 (2003)
14. Prijatelj, A.: Bounded contraction and Gentzen-style formulation of Lukasiewicz logics. Studia Logica 57(2/3), 437–456 (1996)
15. Rothenberg, R.: An Hypersequent Calculus for Lukasiewicz Logic without the Merge Rule. In: Automated Reasoning Workshop (2006)
16. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley and Sons, Chichester (1987)
17. Troelstra, A.S., Schwichtenberg, H.: Basic Proof Theory. Cambridge Tracts in Theoretical Computer Science, vol. 43. Cambridge University Press, Cambridge (1996)

A Proofs of Propositions 3 and 4

Proposition 3. *If the atomic $\mathbb{Z}$-hypersequent $G \mid \Gamma_1 \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2$ is valid in Ł then either $G \mid \Gamma_1, \Gamma_2 \vdash_{n_1+n_2+1} \Delta_1, \Delta_2 \mid \Gamma_1 \vdash_{n_1} \Delta_1$ is valid in Ł or $G \mid \Gamma_1, \Gamma_2 \vdash_{n_1+n_2+1} \Delta_1, \Delta_2 \mid \Gamma_2 \vdash_{n_2} \Delta_2$ is valid in Ł.*

Proof. Let $\mathcal{H} = G \mid \Gamma_1 \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2$ be an atomic $\mathbb{Z}$-hypersequent where G being $\Gamma'_1 \vdash_{n'_1} \Delta'_1 \mid \ldots \mid \Gamma'_k \vdash_{n'_k} \Delta'_k$. By using linear programming [16] $\mathcal{H}$ is valid iff there exist $\alpha'_1, \ldots, \alpha'_k, \alpha_1, \alpha_2 \in \mathbb{N}$ where $\alpha'_i > 0$ or $\alpha_j > 0$ for some $1 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant 2$, such that for every valuation $[\![\cdot]\!]$, $\sum_{i=1}^{k}(\alpha'_i * [\![\Gamma'_i]\!]) + \sum_{i=1}^{2}(\alpha_i * [\![\Gamma_i]\!]) \leqslant \sum_{i=1}^{k}(\alpha'_i * [\![\Delta'_i]\!]) + \sum_{i=1}^{2}(\alpha_i * [\![\Delta_i]\!]) - (\sum_{i=1}^{k}(\alpha'_i * n'_i) + \sum_{i=1}^{2}\alpha_i * n_i)$. We suppose that $\alpha_1 \geqslant \alpha_2$. Then for every valuation $[\![\cdot]\!]$ we have $\sum_{i=1}^{k}(\alpha'_i * [\![\Gamma'_i]\!]) + (\alpha_1 - \alpha_2) * [\![\Gamma_1]\!] + \alpha_2 * ([\![\Gamma_1 + \Gamma_2]\!]) \leqslant \sum_{i=1}^{k}(\alpha'_i * [\![\Delta'_i]\!]) +$

$(\alpha_1-\alpha_2)*\lceil\Delta_1\rceil+\alpha_2*(\lceil\Gamma_1+\Gamma_2\rceil)-(\sum_{i=1}^{k}(\alpha'_i*n'_i)+(\alpha_1-\alpha_2)*n_1+\alpha_2*(n_1+n_2+1))$. Then $G \mid \Gamma_1,\Gamma_2 \vdash_{n_1+n_2+1} \Delta_1,\Delta_2 \mid \Gamma_1 \vdash_{n_1} \Delta_1$ is valid in Ł. The case of $\alpha_2 \geqslant \alpha_1$ is symmetrical.

Proposition 4. *Let $[p]G \mid \Gamma_1,k_1p \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2,k_2p$ be an atomic focused $\mathbb{Z}$-hypersequent. If it is valid then one of the following focused $\mathbb{Z}$-hypersequents is valid:*

- $[p]G \mid k_2\Gamma_1,k_1\Gamma_2 \vdash_{n'} k_2\Delta_1,k_1\Delta_2 \mid \Gamma_1,k_1p \vdash_{n_1} \Delta_1$
- $[p]G \mid k_2\Gamma_1,k_1\Gamma_2 \vdash_{n'} k_2\Delta_1,k_1\Delta_2 \mid \Gamma_2 \vdash_{n_2} \Delta_2,k_2p$

where $k_1>0, k_2>0, p \notin \Gamma_1\cup\Gamma_2\cup\Delta_1\cup\Delta_2$ and $n'=k_2*n_1+k_1*n_2+k_1+k_2-(k_1*k_2+1)$.

Proof. We first prove by induction on k that $[p]G \mid \Gamma \vdash_m \Delta$ is valid iff $G \mid k\Gamma \vdash_n k\Delta$ where $n=k*m+(k-1)$. Then, by Proposition 3, if $[p]G \mid \Gamma_1,k_1p \vdash_{n_1} \Delta_1 \mid \Gamma_2 \vdash_{n_2} \Delta_2,k_2p$ is valid then one of the following focused $\mathbb{Z}$-hypersequents is valid:

- $[p]G \mid k_2\Gamma_1,k_1\Gamma_2,(k_1*k_2)p \vdash_{n''} k_2\Delta_1,k_1\Delta_2,(k_1*k_2)p \mid \Gamma_1,k_1p \vdash_{n_1} \Delta_1$
- $[p]G \mid k_2\Gamma_1,k_1\Gamma_2,(k_1*k_2)p \vdash_{n''} k_2\Delta_1,k_1\Delta_2,(k_1*k_2)p \mid \Gamma_2 \vdash_{n_2} \Delta_2,k_2p$

where $n'=k_2*n_1+k_1*n_2+k_1+k_2-1$. Finally we prove the following result by induction: $[p]G \mid \Gamma,kp \vdash_n \Delta,kp$ is valid in Ł iff $[p]G \mid \Gamma \vdash_{n-k} \Delta$ is valid in Ł. Therefore we deduce the result.

B Proofs of Theorems 4, 5 and 7

Theorem 4. *An atomic $\mathbb{Z}$-hypersequent $\mathcal{H}$ has a countermodel in Ł iff $SI_{\mathcal{H}}$ has a solution over $[0,1]$.*

Proof. Let $\mathcal{H}=\Gamma_1 \vdash_{n_1} \Delta_1 \mid \ldots \mid \Gamma_k \vdash_{n_k} \Delta_k$ be an atomic $\mathbb{Z}$-hypersequent. $[\![\cdot]\!]$ is a countermodel of $\mathcal{H}$ in Ł iff for all $i\in\{1,\ldots,k\}$, the inequality $\lfloor\Gamma_i\rfloor > \lceil\Delta_i\rceil - n_i$ holds. Thus, $[\![\cdot]\!]$ is a countermodel of $\mathcal{H}$ iff for all $i\in\{1,\ldots,k\}$, $(x_p=[\![p]\!] \mid p\in\Gamma_i\cup\Delta_i)$ is a solution of $\bigodot\Gamma_1 > \bigoplus\Delta_1 - n_1$. Therefore, $\mathcal{H}$ has a countermodel in Ł iff $SI_{\mathcal{H}}$ has a solution. This solution is over $[0,1]$ because the valuations in Ł are from Var to $[0,1]$.

Theorem 5. *An atomic $\mathbb{Z}$-hypersequent H has a countermodel in $Ł_n$ for $n\neq\infty$ iff $SI^n_{\mathcal{H}}$ has a solution over the set of integers $\{0,\ldots,n-1\}$.*

Proof. Let $\mathcal{H}=\Gamma_1 \vdash_{m_1} \Delta_1 \mid \ldots \mid \Gamma_k \vdash_{m_k} \Delta_k$ be an atomic $\mathbb{Z}$-hypersequent. By using arguments used in the proof of Theorem 4, we show that $\mathcal{H}$ has a countermodel in $Ł_n$ iff the inequality $1+\sum_{A\in\Gamma_i}(x_A-1) > \sum_{A\in\Delta_i} x_A$ has a solution over $[0,1/(n-1),\ldots,(n-2)/(n-1),1]$. Thus $\mathcal{H}$ has a countermodel in $Ł_n$ iff $(n-1)+\sum_{A\in\Gamma_i}(x_A-(n-1)) > \sum_{A\in\Delta_i} x_A$ has a solution over $\{0,\ldots,n-1\}$.

Theorem 7. *If a $\mathbb{Z}$-hypersequent is valid in Ł then it is derivable in $\mathbb{Z}$Ł.*

Proof. From Theorem 3, by applying the logical rules of $\mathbb{Z}Ł$ to every $\mathbb{Z}$-hypersequent $\mathcal{H}$ we obtain a set $\mathcal{S}$ of atomic $\mathbb{Z}$-hypersequents such that $\mathcal{H}$ is valid iff all elements of $\mathcal{S}$ are valid. Let $\mathcal{H} = \Gamma_1 \vdash_{m_1} \Delta_1 \mid \ldots \mid \Gamma_k \vdash_{m_k} \Delta_k$ be an atomic $\mathbb{Z}$-hypersequent. We assume that $\mathcal{H}$ is valid. Hence, the set $SI_{\mathcal{H}}$ of inequalities is not feasible over $[0,1]$. Then, by using linear programming [16], there exists a positive nonnegative combination of the inequalities in $SI_{\mathcal{H}}$ inconsistent over $[0,1]$. Formally, $\exists \alpha_1, \ldots, \alpha_k \in \mathbb{N}$ such that for some $i \in 1, \ldots, K$ we have $\alpha_i > 0$ and the inequality $\alpha_1 * (\bigodot \Gamma_1) + \ldots + \alpha_k * (\bigodot \Gamma_k) > \alpha_1 * (\bigoplus \Delta_1) - \alpha_1 * m_1 + \ldots + \alpha_k * (\bigoplus \Delta_k) - \alpha_k * m_k$ is inconsistent over $[0,1]$. We can easily show, by using Definition 4, that the last inequality is inconsistent over $[0,1]$ iff the $\mathbb{Z}$-hypersequent $\alpha_1\Gamma_1, \ldots, \alpha_k\Gamma_k \vdash_n \alpha_1\Delta_1, \ldots, \alpha_k\Delta_k$ is valid in Ł, where $n = \alpha_1 * (m_1 + 1) + \ldots + \alpha_k * (m_k + 1) - 1$ and for all $i \in 1, \ldots, K$, $\alpha_i\Gamma_i$ (resp. $\alpha_1\Delta_i$) denotes the multiset obtained by the union of α_i copies of the multiset Γ_i (resp. Δ_i). This $\mathbb{Z}$-hypersequent can be obtained from $\mathcal{H}$ by using the external weakening ($[EW]$) and the external contraction rules ($[EC]$).

Let $\Gamma \vdash_n \Delta$ be an atomic $\mathbb{Z}$-hypersequent. We can easily prove that if there is a multiset of formulae Γ_1, subset of Γ and Δ, then $\Gamma \vdash_n \Delta$ is valid iff $\Gamma - \Gamma_1 \vdash_{n-n'} \Delta - \Gamma_1$ is valid, where $n' = \mid \Gamma_1 \mid$ such that $\mid S \mid$ denotes the number of elements in the multiset S. Moreover, if $l\bot \subseteq \Gamma$ such that $l\bot$ denotes the multiset containing l copies of $\bot$, then $\Gamma \vdash_n \Delta$ is valid iff $\Gamma - l\bot \vdash_{n-l} \Delta$ is valid. From these results we obtain $\Gamma \vdash_n \Delta$ is valid iff $\Gamma = \Gamma_1 \cup \Gamma_2 \cup l\bot$ such that $\bot \notin \Gamma_2$; $\Delta = \Delta_1 \cup \Delta_2$; $\Gamma_1 = \Delta_1$; $\Gamma_2 \cap \Delta_2 = \emptyset$; and $\mid \Gamma_2 \mid \leqslant \mid \Gamma \mid -n-1$.Then, $\alpha_1\Gamma_1, \ldots, \alpha_k\Gamma_k \vdash_n \alpha_1\Delta_1, \ldots, \alpha_k\Delta_k$ such that $n = \alpha_1 * (m_1 + 1) + \ldots + \alpha_k * (m_k + 1) - 1$ is derivable in $\mathbb{Z}Ł$ by using $[Ax]$, $[SR]$, $[IW_L]$, $[IW_R]$ and $[\bot_L]$. If a $\mathbb{Z}$-hypersequent is valid in Ł then it is derivable in $\mathbb{Z}Ł$.

C Bounded Łukasiewicz Logic

Bounded Łukasiewicz logic $ŁB_n$ for $n \geqslant 2$ is defined as the intersection of $Ł_k$ for $k = 2, \ldots, n$. A Hilbert axiomatic system for this logic consists of the same axioms and rules than Ł with $nA \supset (n-1)A$. Calculi for $ŁB_n$, called $GŁB_n$ [4], are obtained by adding to the hypersequent calculus $GŁ$ given in [12] the following rule:

$$\frac{G \mid \overbrace{\Gamma, \ldots \Gamma}^{n-1}, \Gamma', \bot \vdash \overbrace{\Delta, \ldots, \Delta}^{n-1}, \Delta'}{G \mid \Gamma \vdash \Delta \mid \Gamma' \vdash \Delta'} \; [nC]$$

It appears that this rule makes proof-search expensive because it duplicates the contexts Γ and Δ n-1 times. Here we introduce new calculi for Bounded Łukasiewicz logics that are simpler than $GŁB_n$. We call $\mathbb{Z}ŁB_n$ the calculus obtained from $\mathbb{Z}Ł'$ by adding:

$$\frac{G \mid \Gamma_1 \vdash_{m_1} \Delta_1 \qquad G \mid \Gamma_2 \vdash_{m_1} \Delta_2}{G \mid \Gamma_1, \Gamma_2 \vdash_{m_1+m_2+1} \Delta_1, \Delta_2} \; [M] \qquad \frac{G \mid \Gamma, A \vdash_{m+1} \Delta, A}{G \mid \Gamma \vdash_m \Delta} \; [GCUT]$$

$$\frac{}{G \mid \Gamma \vdash_0 \Delta, \overbrace{A, \ldots, A}^{n-1} \mid \Gamma', A \vdash_0 \Delta'} \; [Ax_n]$$

Theorem 12 (Soundness). *The rules of* $\mathbb{Z}\text{Ł}\mathsf{B}_\mathsf{n}$ *are sound.*

Proof. By Theorem 1 the logical rules are sound. The rules $[M]$, $[GCUT]$, $[SR]$, $[S]$ and $[\bot_L]$ are proved sound by similar arguments. Let us consider $[AX_n]$. We suppose that $\mathcal{H} = G \mid \Gamma \vdash_0 \Delta, \overbrace{A,\ldots,A}^{n-1} \mid \Gamma', A \vdash_0 \Delta'$ has a countermodel. Thus, for $k \in \{2,\ldots,n\}$ there is a valuation $[\![\cdot]\!]$ countermodel of $\mathcal{H}$ in Ł$_k$. Thus, there exists $i \in \{0,\ldots,k-1\}$ such that $[\![A]\!] = \frac{i}{k-1}$. If $[\![A]\!] = 0$ then $[\![\Gamma', A]\!] \leqslant 0 \leqslant [\![\Delta']\!]$ and we get a contradiction. Now, if $[\![A]\!] = \frac{i}{k-1}$ with $i \neq 0$ then $[\![\Gamma]\!] \leqslant 1 \leqslant (n-1) * \frac{i}{k-1} + [\![\Delta]\!]$ because $n \geqslant k$. This is a contradiction.

Theorem 13 (Completeness). *If A is valid in* ŁB_n *then* $\vdash_0 A$ *is derivable in* $\mathbb{Z}\text{Ł}\mathsf{B}_\mathsf{n}$.

Proof. We have only to prove that (1) the axiom $nA \supset (n-1)A$ is derivable in $\mathbb{Z}\text{Ł}\mathsf{B}_\mathsf{n}$ and (2) the modus ponens rule is admissible in $\mathbb{Z}\text{Ł}\mathsf{B}_\mathsf{n}$. Then we have:

$$\dfrac{\dfrac{\vdash_0 (n-1)A \mid A \vdash_0}{\vdash_0 (n-1)A \mid (n-1)A, A \vdash_1 (n-1)A}\,[SR]}{A \oplus ((n-1)A) \vdash_0 (n-1)A}\,[\oplus_L]$$

and by using $[\oplus_R]$ n-1 times, we obtain the axiom $\vdash_0 \overbrace{A,\ldots,A}^{n-1} \mid A \vdash_0$. A proof of (2) is given by the following derivation:

$$\dfrac{\dfrac{A \vdash_0 B \quad \vdash_0 A}{A \vdash_1 B, A}\,[M]}{\vdash_0 B}\,[GCUT]$$

In next works we will study the cut-elimination problem.

An Infinitely-Often One-Way Function Based on an Average-Case Assumption

Edward A. Hirsch* and Dmitry M. Itsykson**

Steklov Institute of Mathematics at St. Petersburg,
27 Fontanka, St. Petersburg 191023, Russia
http://logic.pdmi.ras.ru/~hirsch/,
http://logic.pdmi.ras.ru/~dmitrits/

Abstract. We assume the existence of a function f that is computable in polynomial time but its inverse function is not computable in randomized average-case polynomial time. The cryptographic setting is, however, different: even for a weak one-way function, every possible adversary should fail on a polynomial fraction of inputs. Nevertheless, we show how to construct an *infinitely-often* one-way function based on f.

1 Introduction

A function is called weakly one-way if for some positive c every randomized polynomial-time algorithm for inverting it fails with probability at least $\frac{1}{n^c}$. One-way functions are one of the main cryptographic primitives. However, no reasonable complexity assumption (such as, say, $\mathbf{P} \neq \mathbf{NP}$ or $\mathbf{AvgP} \neq \mathbf{DistNP}$) is known that would imply the existence of one-way functions.

We say that an algorithm with two parameters x (input) and δ ("give up" probability) runs in a polynomial time on average, if its running time is bounded by a polynomial in $\frac{|x|}{\delta}$ and it "gives up" with probability at most δ (otherwise, it gives the correct answer). This definition is given by Impagliazzo in his influential survey paper on average-case complexity [Imp95]; it is equivalent to Levin's definition of average-case tractability [Lev86].

The three obstacles that prevent using such average-case complexity notions in the cryptographic setting, are

1. A successful cryptographic adversary may err on a polynomial fraction of inputs [Gol01, Definition 2.2.2], while in the average-case setting this is not enough to solve a problem: if one spends exponential time on these inputs, the average-case complexity is not polynomial [Lev86, BT06].

* Partially supported by RFBR grant 08-01-00640, the President of Russia grant for leading scientific schools support NSh-4392.2008.1, and the Dynasty foundation fellowship.

** Partially supported by Russian Science Support Foundation, RFBR grant 08-01-00640, and the President of Russia grant for leading scientific schools support NSh-4392.2008.1.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 208–217, 2008.

2. The (polynomial-time samplable) probability distribution for the cryptographic setting is taken over function inputs, while in the average-case setting it is taken over the outputs (i.e., the instances of the search problem of computing the inverse function): see, e.g., [Imp95, Lev03]. Note that a polynomial-time samplable distribution on the outputs is not necessarily dominated by the distribution induced by a polynomial-time samplable distribution on the inputs.
3. To solve a problem in the average-case, one should solve it on all input lengths, while in the cryptographic case a successful adversary is allowed to solve it just on an infinite number of input lengths. We do not know if this definitional discrepancy can be overcome, and follow the average-case tradition in this work. Thus the function that we obtain is hard to invert only on an infinite number of (rather than on almost all) input lengths.

In this paper we address item 1 of the above list: we prove that average-case hardness of inverting a function implies cryptographic hardness of inverting a related function on an infinite number of input lengths. Namely, we show how to pad any function so that we can use any polynomial-time algorithm that inverts the padded function with any noticeable probability of success for inverting the padded function (as well as the original non-padded function) in polynomial time on the average. The reduction essentially uses the fact that the two concepts use a similarly defined set of input lengths, thus we do not resolve item 3. We do not attempt to resolve item 2 either.

Our method uses the following simple idea of the proof of [Imp95, Proposition 3] of getting an average-case tractable algorithm out of an algorithm that works on a fixed fraction of inputs ($1 - \frac{1}{n^k}$). Bogdanov and Trevisan use this idea to prove that the existence of an algorithm that fails on a $\frac{1}{n}$ fraction of inputs for a version of the bounded halting problem implies the average-case easiness of all **NP** languages with polynomial-time samplable distributions ([BT06, Proposition 3.5]); the trick is: if one pads the input, the failure probability decreases. We adapt this method for the problem of inverting a function. Instead of taking a particular function we show how to modify *any* function to fit it. Note that a straightforward approach of applying the argument to a (**DistNP**-hard) search version of the bounded halting problem fails because for the (cryptographic) problem of inverting a function one needs a (polynomial-time samplable) probability distribution on inputs of this function and not on its outputs.

Our main result is the following theorem:

Theorem 3. *If there is a length-preserving polynomial-time computable function f that cannot be inverted in randomized average-case polynomial time on a polynomial-time samplable distribution on its inputs, then there exists a length-preserving function that is one-way for infinitely many input lengths.*

Organization of the paper. In Sect. 2 we define rigorously the notions we use. In Sect. 3 we prove the main result (Theorem 3).

2 Preliminaries

2.1 Average-Case Complexity

In this subsection we define the core notions of the average-case complexity. We basically follow [BT06] (technically, [BT06] allows distributions that are defined not on every input length, but it does not make any difference for us).

Definition 1. *An ensemble of distributions is a collection* $D = \{D_n\}_{n=1}^{\infty}$ *where* $D_n \colon \{0,1\}^n \to \mathbb{R}$ *is a distribution on inputs of length* n *(i.e.,* $\sum_{a \in \{0,1\}^n} D_n(a) = 1$*).*

Definition 2. *A function* $f \colon \{0,1\}^* \to 2^{\{0,1\}^*}$ *is called polynomial-time verifiable if every string in its output is polynomially bounded in the length of the input and there exists a polynomial-time computable function* v *such that*

$$\forall x, y \in \{0,1\}^* \quad v(x,y) = 1 \iff y \in f(x).$$

Definition 3. *A distributed search problem* (f, D) *consists of a polynomial-time verifiable function* $f \colon \{0,1\}^* \to 2^{\{0,1\}^*}$ *and an ensemble of distributions* D*.*

Remark 1. Bogdanov and Trevisan [BT06] consider search algorithms for **NP** languages instead of formally defining distributed search problems, though these approaches are obviously equivalent.

We follow Impagliazzo [Imp95] and Bogdanov and Trevisan [BT06] in defining average-case polynomial-time algorithms as polynomial-time algorithms that are allowed to "give up" (by outputting a special symbol $\perp$).

Definition 4 (cf. [BT06, Definition 4.2]). *A distributed search problem* (f, D) *can be solved in polynomial time on the average if there exists an algorithm* $A(x, \delta)$ *such that*

- *A runs in time polynomial in* $|x|$ *and* $\frac{1}{\delta}$ *for any* x *in the support of* D *and any positive* δ*;*
- *if* $f(x) \neq \emptyset$*, then* $A(x, \delta) \in f(x) \cup \{\perp\}$*;*
- $\Pr_{x \leftarrow D_n}\{A(x, \delta) = \perp\} \leq \delta$*.*

Definition 5. *Complexity class* **FAvgP** *consists of all distributed search problems that can be solved in polynomial time on the average.*

Definition 6 (cf. [BT06, Definition 4.3]). *A distributed search problem* (f, D) *can be solved in randomized polynomial time on the average if there exists a randomized algorithm* $A(x, \delta)$ *such that*

- *A runs in time polynomial in* $|x|$ *and* $\frac{1}{\delta}$ *for any* x *in the support of* D *and any positive* δ*;*
- *if* $f(x) \neq \emptyset$*, then* $\Pr\{A(x, \delta) \notin f(x) \cup \{\perp\}\} \leq \frac{1}{4}$ *where the probability is taken over the random bits of* A*;*

– $\Pr_{x \leftarrow D_n}\{\Pr\{A(x,\delta) = \perp\} \geq \frac{1}{4}\} \leq \delta$ *where the inner probability is taken over the random bits of* A.

Definition 7. *Complexity class* **FAvgBPP** *consists of all distributed search problems that can be solved in randomized polynomial time on the average.*

The following definition of a (deterministic) reduction is a special case of randomized heuristic search reduction [BT06, 5.1.1]. While **FAvgBPP** might not be closed under these randomized reductions, it is closed under the deterministic ones. In this paper we use only deterministic reductions.

Definition 8. *Consider two distributed search problems* (f, D) *and* (f', D'). *We say that* (f, D) *reduces to* (f', D'), *if there are polynomial-time computable functions* h, g *such that the two following statements hold:*

– $f(x) \neq \emptyset \implies f'(h(x)) \neq \emptyset$;
– $y \in f'(h(x)) \implies g(y) \in f(x)$ *for any* y *and* x *with* $D_{|x|}(x) > 0$;
– *there is a polynomial* $p(n)$ *such that*

$$\sum_{h(x)=x',\ |x|=n} D_n(x) \leq p(n) \cdot D'_{|x'|}(x')$$

for any x'.

(The last condition is called the *domination* condition.) We now formally verify that both **FAvgP** and **FAvgBPP** are closed under such reductions.

Lemma 1. *If a problem* (f, D) *is reducible to a problem* (f', D'), *then* $(f', D') \in$ **FAvgP** *implies* $(f, D) \in$ **FAvgP**.

Proof. Let $A'(y, \delta)$ be an average-case polynomial-time algorithm for the problem (f', D'). Let q be a polynomial such that $\max_{x \in \{0,1\}^n} |h(x)| \leq q(n)$. Define $A(x, \delta) = g(A'(h(x), \frac{\delta}{p(|x|)q(|x|)}))$ (we assume $g(\perp) = \perp$). Clearly, the algorithm A is polynomial in $|x|$ and in $\frac{1}{\delta}$ and does not output wrong answers when $f(x) \neq \emptyset$. The probability of the "give up" answer can be estimated as

$$\Pr_{x \leftarrow D_n}\{A(x,\delta) = \perp\} = \sum_{A'(h(x), \frac{\delta}{p(n)q(n)}) = \perp} D_n(x) \leq \sum_{A'(y, \frac{\delta}{p(n)q(n)}) = \perp} p(n) D'_{|y|}(y) \leq$$

$$p(n)q(n)\frac{\delta}{p(n)q(n)} = \delta.$$

□

Lemma 2. *If a problem* (f, D) *is reducible to a problem* (f', D') *then* $(f', D') \in$ **FAvgBPP** *implies* $(f, D) \in$ **FAvgBPP**.

Proof. The construction of the new algorithm A and the verification of the probability of the "give up" answer is similar to the deterministic case (Lemma 1):

$$\Pr_{x\leftarrow D_n}\{\Pr\{A(x,\delta)=\perp\}\geq\frac{1}{4}\}=\sum_{\Pr\{A'(h(x),\frac{\delta}{p(n)q(n)})=\perp\}\geq\frac{1}{4}} D_n(x)$$

$$\leq\sum_{\Pr\{A'(y,\frac{\delta}{p(n)q(n)})=\perp\}\geq\frac{1}{4}} p(n)D'_{|y|}(y)\leq p(n)q(n)\frac{\delta}{p(n)q(n)}=\delta.$$

The additional condition for randomized algorithms can be also easily verified:

$$\Pr\{A(x,\delta)\notin f(x)\cup\{\perp\}\}\leq\Pr\{A'(h(x),\frac{\delta}{p(n)q(n)})\notin f'(h(x))\cup\{\perp\}\}\leq\frac{1}{4}.$$

□

2.2 Infinitely-Often One-Way Functions

In this section we define infinitely-often one-way functions[1] that differ from "standard" one-way functions in that they are hard only on an infinite number of (rather than on almost all) input lengths. In other words, in a contrast to, e.g., [Gol01, Definition 2.2.1] we require the adversary to invert the function on all sufficiently large input lengths to violate the one-wayness condition, while in the "classical" definition it is enough to invert it on an infinite number of input lengths.

Definition 9. *A polynomial-time computable function $f\colon\{0,1\}^*\to\{0,1\}^*$ is a strong i.o.-one-way function if for any polynomial $p(n)$ and any randomized polynomial-time algorithm B,*

$$\forall N\ \exists n{>}N\ \Pr\{B(f(x))\in f^{-1}(f(x))\}<\frac{1}{p(n)}$$

where the probability is taken over x uniformly distributed on $\{0,1\}^n$ and over the random bits used by B.

We adjust the definition of weak one-way functions similarly (cf. [Gol01, Definition 2.2.2] for "ordinary" one-way functions).

Definition 10. *A polynomial-time computable function $f\colon\{0,1\}^*\to\{0,1\}^*$ is a weak i.o.-one-way function if there exists polynomial $p(n)$ such that for any randomized polynomial-time algorithm B,*

$$\forall N\ \exists n{>}N\ \Pr\{B(f(x))\notin f^{-1}(f(x))\}>\frac{1}{p(n)}$$

where the probability is taken over x uniformly distributed on $\{0,1\}^n$ and over the random bits used by B.

Theorem 1. *The existence of weak i.o.-one-way functions implies the existence of strong i.o.-one-way functions.*

Proof. The proof follows the proof of [Gol01, Theorem 2.3.2] (for "ordinary" one-way functions) literally. □

[1] The term was suggested to us by an anonymous referee of ECCC.

3 Main Result

3.1 Proof Strategy

We assume the existence of a length-preserving function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that the search problem of inverting f on the distribution resulting from the uniform[2] distribution on the inputs of f cannot be solved in randomized polynomial time on the average. Note that f is not necessarily weak i.o.-one-way function: suppose that f is polynomial-time invertible on a set of probability $(1 - \frac{1}{2^{\sqrt{n}}})$, and invertible in time $\Omega(2^n)$ on other inputs. In that case f may be hard to invert in polynomial time on the average but it is not weakly i.o.-one-way. We will show how to modify f so that it becomes weakly i.o.-one-way.

The main idea of the proof is to supply the original function with *padding*. Namely, we define a new function $f_p(x,y)$ on pairs of strings that applies f to its first argument and replaces the second argument by $1^{|y|}$. Note that the probability of an input $f_p(x,y)$ does not depend on the length of padding (the probability is exactly $2^{-|f(x)|}$). By the definition of a randomized average-case polynomial-time algorithm it is required to solve much more instances at higher lengths (the probability of error is a constant), and padding allows us to put the instances of smaller lengths into higher lengths.

We show that the problem of inverting f is average-case reducible to the problem of inverting f_p. Indeed, to invert f on the string y it is sufficient to invert f_p on the pair $(y,1)$. To verify the domination condition we have to specify an economic encoding of the pairs (to satisfy this condition, we are allowed to increase the string length only by a logarithmic number). The details of the encoding are given in the next section.

Suppose that there is a randomized polynomial-time algorithm B that inverts f_p with $1/n$ error: $\Pr\{B(f_p(z)) \in f_p^{-1}(f_p(z))\} \geq 1 - \frac{1}{n}$ where the probability is taken both by z and by the random choices of B. We now define a randomized average-case polynomial-time algorithm $A(z,\delta)$ that inverts f_p. The paradigm is: increase the padding of the input $z \mapsto z1^{\lceil \frac{1}{\delta} \rceil}$ and then use B. Since the probability of the input does not depend on the length of padding, the probability of error of A is at most $\frac{1}{n+\frac{1}{\delta}} \leq \delta$. Thus, inverting f_p, and, by the reduction above, inverting f, is randomized average-case tractable, which contradicts the assumption.

3.2 Proof Details

Throughout this section log denotes the binary logarithm. For a string x, we denote the binary representation of its length $|x|$ (as a string) by $|x|_2$.

Definition 11. *We say that a string $x \in \{0,1\}^*$ is* correct *if*

- *x contains at least one occurrence of 0; let $x_k = 0$ be the first such occurrence;*
- *$|x| \geq 2k - 1$;*
- *the substring $x_{k+1} \ldots x_{2k-1}$ is the binary representation of the number l such that $|x| \geq 2k + l - 1$.*

[2] It is later shown that one can assume any polynomial-time samplable distribution here.

For a correct x, its main part *is the substring $x_{2k}..x_{2k+l-1}$, and its* padding *is the suffix $x_{2k+l}..x_{|x|}$.*

Definition 12. *Let $\pi\colon \{0,1\}^* \to \{0,1\}^*$ be a function that maps a string x to the correct string $\pi(x)$ with the main part x and the empty padding, i.e., $\pi(x) = 1^{\lceil \log |x| \rceil}0|x|_2 x$.*

Remark 2. Note that π is injective.

Definition 13. *Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a length-preserving function. We then define a new (also length-preserving) function f_p as follows: if there are y and z such that $x = 1^{\lceil \log |z| \rceil}0|z|_2 zy$, then $f_p(x) = 1^{\lceil \log |z| \rceil}0|z|_2 f(z)1^{|y|}$, otherwise $f_p(x) = x$.*

Definition 14. *For any length-preserving function g and distribution D, the distribution D^g is generated as the output of the function g whose inputs are sampled according to D. In particular,*

$$U^g(y) = \sum_{g(x)=y} 2^{-|x|} = |g^{-1}(y)| \cdot 2^{-|y|}.$$

Lemma 3. *If $x = 1^{\lceil \log |z| \rceil}0|z|_2 z1^t$, then $U^{f_p}(x) = |f^{-1}(z)| \cdot 2^{-|z|-2\lceil \log |z| \rceil - 1} = U^{f_p}(x1^s)$ for any $s \geq 0$. If x is a correct string whose padding contains zeroes, then $U^{f_p}(x) = 0$. If a string x is incorrect, then $U^{f_p}(x) = 2^{-|x|}$.*

Proof. In the first case one has to sum up the probabilities for different original paddings. The other two cases are trivial. □

Lemma 4. *For any length-preserving function f, the problem (f^{-1}, U^f) is reducible to (f_p^{-1}, U^{f_p}).*

Proof. To satisfy Definition 8, assume

$$h(x) = \pi(x),$$
$$g(y) = \begin{cases} x, & \text{if } y = \pi(x), \\ y, & \text{otherwise,} \end{cases}$$
$$p(n) = 2n^3.$$

If f is invertible on x, then f_p is invertible on $\pi(x)$. If f_p is invertible on $\pi(x)$, then $g(f_p^{-1}(\pi(x))) \in f^{-1}(x)$. If f is not invertible on x, then trivially $U^f(x) = 0$. Let $n = |x|$. Finally, if $x' = \pi(x)$, then by Lemma 3

$$U^{f_p}(x') = |f^{-1}(x)| \cdot 2^{-n-2\lceil \log n \rceil - 1} \geq \frac{1}{2n^3} \cdot |f^{-1}(x)|2^{-n} = \frac{1}{p(n)} U_n^f(x) = \frac{1}{p(n)} \sum_{\pi(y)=x'} U_n^f(y).$$

The last equality holds since π is injective. (If x' cannot be represented as $\pi(x)$, the domination condition is trivially satisfied.) □

Theorem 2. *Let f be a length-preserving polynomial-time computable function. If there exists a randomized polynomial-time algorithm B such that for a constant $c > 0$ and every integer n $\Pr_{x \leftarrow U_n^{f_p}, r \leftarrow U_{s(n)}}\{B(x) \in f_p^{-1}(x)\} \geq 1 - \frac{1}{n^c}$, where r is the string of random bits used by the algorithm B, $s(n)$ is a polynomial, then $(f_p^{-1}, U_n^{f_p}) \in$ **FAvgBPP**.*

Proof. Since one can verify the answer of B, we assume that either B correctly inverts f_p or gives up. We also assume that when B returns an element of $f_p^{-1}(x)$, it chooses one without zeroes in its padding.

We first prove that

$$\Pr_{x \leftarrow U_n^{f_p}} \left\{ \Pr_{r \leftarrow U_{s(n)}} \{B(x) \notin f_p^{-1}(x)\} \geq \frac{1}{4} \right\} \leq \frac{4}{n^c}. \tag{1}$$

Indeed, if this inequality is incorrect, then

$$\Pr_{x \leftarrow U_n^{f_p}, r \leftarrow U_{s(n)}} \{B(x) \notin f_p^{-1}(x)\} = \sum_{x \in \{0,1\}^n} U_n^{f_p}(x) \cdot \Pr_{r \leftarrow U_{s(n)}} \{B(x) \notin f_p^{-1}(x)\} \geq$$

$$\sum_{\substack{x \in \{0,1\}^n \\ \Pr_{r \leftarrow U_{s(n)}}\{B(x) \notin f_p^{-1}(x)\} \geq \frac{1}{4}}} U_n^{f_p}(x) \cdot \Pr_{r \leftarrow U_{s(n)}} \{B(x) \notin f_p^{-1}(x)\} > \frac{1}{4} \cdot \frac{4}{n^c} = \frac{1}{n^c},$$

which contradicts the assumption about B.

The new algorithm $A(x, \delta)$ performs as follows. If x is an incorrect string, then $A(x, \delta) = x$. If x is a correct string with padding without zeros (note that if the padding contains zeros, then $U^{f_p}(x) = 0$), then we pad x and run B. More precisely, let $|x| = n$, $\Delta = \lceil (\frac{4}{\delta})^{1/c} \rceil$, $N = n + \Delta$. Define $\sigma(x) = x1^{\Delta}$. If $B(\sigma(x)) = \perp$ then $A(x, \delta) = \perp$, otherwise $A(x, \delta)1^{\Delta} = B(\sigma(x))$, i.e., A strips Δ trailing 1's of B's answer and outputs the result.

Then

$$\Pr_{x \leftarrow U_n^{f_p}} \left\{ \Pr_{r \leftarrow U_{s(N)}} \{A(x, \delta) \notin f_p^{-1}(x)\} \geq \frac{1}{4} \right\} \leq$$

$$\Pr_{x \leftarrow U_n^{f_p}} \left\{ \Pr_{r \leftarrow U_{s(N)}} \{B(\sigma(x)) \notin f_p^{-1}(\sigma(x))\} \geq \frac{1}{4} \right\} \overset{\text{Lemma 3}}{=}$$

$$\Pr_{y \leftarrow U_N^{f_p}} \left\{ \exists x (x \in \{0,1\}^n \wedge y = \sigma(x)) \wedge \Pr_{r \leftarrow U_{s(N)}} \{B(y) \notin f_p^{-1}(y)\} \geq \frac{1}{4} \right\} \leq$$

$$\Pr_{y \leftarrow U_N^{f_p}} \left\{ \Pr_{r \leftarrow U_{s(N)}} \{B(y) \notin f_p^{-1}(y)\} \geq \frac{1}{4} \right\} \overset{(1)}{\leq} \frac{4}{N^c} < \delta.$$

□

Corollary 1. *Let f be a length-preserving polynomial-time computable function. If the problem $(f^{-1}, U_n^f) \notin$ **FAvgBPP**, then for any randomized polynomial-time algorithm B and for any constant $c > 0$, there exist infinitely many $n \in \mathbb{N}$ such that $\Pr_{x \leftarrow U_n, r \leftarrow U_{s(n)}}\{B(f_p(x)) \in f_p^{-1}(f_p(x))\} < 1 - \frac{1}{n^c}$.*

Proof. By Theorem 2, Lemma 4, and Lemma 2. □

Using Theorem 1 one gets

Corollary 2. *If there exists a length-preserving polynomial-time computable function f that cannot be inverted in randomized average-case polynomial time (i.e., $(f^{-1}, U^f) \notin$ **FAvgBPP**), then there exists a length-preserving strong i.o.-one-way function.*

Corollary 2 formally requires that the function used in the assumption has a uniform distribution on its inputs. However, it is easy to allow any other polynomial-time samplable distribution on the inputs, which is formally proved in the following theorem.

Theorem 3. *If there is a length-preserving polynomial-time computable function f that cannot be inverted in randomized average-case polynomial time on a polynomial-time samplable distribution D (i.e., $(f^{-1}, D^f) \notin$ **FAvgBPP**), then there exists a length-preserving strong i.o.-one-way function.*

Proof. Let $s\colon \{0,1\}^{m(n)} \to \{0,1\}^n$ be a polynomial sampler for D, i.e., $D = U^s$. W.l.o.g. we may assume that $m(n) \geq n$ and $m(n)$ is a strictly increasing polynomial function. We define function f^s as follows:

$$f^s(x) = \begin{cases} f(s(x))1^{m(n)-n}, & \text{if } \exists n(m(n) = |x|), \\ x, & \text{otherwise.} \end{cases}$$

We reduce (f^{-1}, D^f) to $((f^s)^{-1}, U^{f^s})$. To satisfy Definition 8, assume

$$\begin{aligned} h(y) &= y1^{m(n)-n}, \\ g(x) &= s(x), \\ p(n) &= 1. \end{aligned}$$

The domination condition is satisfied since $D^f(y) = U^{f^s}(h(y))$.

Since **FAvgBPP** is closed under reductions, we have $((f^s)^{-1}, U^{f^s}) \notin$ **FAvgBPP**. The theorem now follows from Corollary 2. □

Acknowledgement

The authors are very grateful to Dima Grigoriev, Arist Kojevnikov, Sergey Nikolenko, and Alexander Shen for helpful discussions, and to an anonymous referee of ECCC for valuable comments that improved the presentation of the paper.

References

[BT06] Bogdanov, A., Trevisan, L.: Average-case complexity. Foundation and Trends in Theoretical Computer Science 2(1), 1–106 (2006)

[Gol01] Goldreich, O.: Foundations of Cryptography, vol. 1. Cambridge University Press, Cambridge (2001)

[Imp95] Impagliazzo, R.: A personal view of average-case complexity. In: Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT 1995), pp. 134–147 (1995)

[Lev86] Levin, L.: Average case complete problems. SIAM Journal on Computing 15(1), 285–286 (1986)

[Lev03] Levin, L.A.: The tale of one-way functions. Problems of Information Transmission 39(1), 92–103 (2003)

On Characteristic Constants of Theories Defined by Kolmogorov Complexity

Shingo Ibuka, Makoto Kikuchi, and Hirotaka Kikyo

Dept. of Computer Science and Systems Engineering, Kobe University,
1-1 Rokkodai, Nada, Kobe 657-8501, Japan
ibuka@kurt.scitec.kobe-u.ac.jp, mkikuchi@kobe-u.ac.jp, kikyo@kobe-u.ac.jp

Abstract. Chaitin discovered that for each formal system $\mathbf{T}$, there exists a constant c such that no sentence of the form $K(x) > c$ is provable in $\mathbf{T}$, where $K(x)$ is the Kolmogorov complexity of x. We call the minimum such c the Chaitin characteristic constant of $\mathbf{T}$, or $c_{\mathbf{T}}$. There have been discussions about whether it represents the information content or strength of $\mathbf{T}$. Raatikainen tried to reveal the true source of $c_{\mathbf{T}}$, stating that it is determined by the smallest index of Turing machine which does not halt but we cannot prove this fact in $\mathbf{T}$. We call the index the Raatikainen characteristic constant of $\mathbf{T}$, denoted by $r_{\mathbf{T}}$. We show that $r_{\mathbf{T}}$ does not necessarily coincide with $c_{\mathbf{T}}$; for two arithmetical theories $\mathbf{T}$, $\mathbf{T}'$ with a Π_1-sentence provable in $\mathbf{T}'$ but not in $\mathbf{T}$, there is an enumeration of the Turing machines such that $r_{\mathbf{T}} < r_{\mathbf{T}'}$ and $c_{\mathbf{T}} = c_{\mathbf{T}'}$.

1 Introduction

Algorithmic information theory, originated by Andrey N. Kolmogorov, R. Solomonoff and Gregory J. Chaitin, brought a formalization of the information content of an individual object as Kolmogorov complexity. The Kolmogorov complexity of a number x is defined as the smallest code of Turing machine which outputs x. Chaitin[1] proved the incompleteness theorem in the following form: for a sound, finitely-specified, formal system $\mathbf{T}$ and an enumeration of the Turing machines, there exists a bound c such that $K(x) > c$ is not provable for any number x in $\mathbf{T}$. We call the minimum such c the Chaitin characteristic constant of $\mathbf{T}$, denoted by $c_{\mathbf{T}}$.

The received interpretation of this Chaitin's result is that the Chaitin characteristic constant $c_{\mathbf{T}}$ measures the information content or strength of the formal system $\mathbf{T}$. Michiel van Lambalgen[2] criticized the received interpretation and pointed out that $c_{\mathbf{T}}$ is not determined only by the theory $\mathbf{T}$, but also influenced by the choice of enumeration of the Turing machines. Panu Raatikainen[3] refined Lambalgen's argument and showed that for any formal system $\mathbf{T}$, there exists an enumeration of the Turing machines which makes $c_{\mathbf{T}}$ zero, or arbitrarily large. In the same paper he tried to give a characterization of $c_{\mathbf{T}}$. Let $r_{\mathbf{T}}$ denote the smallest code of Turing machine which does not halt but we cannot prove its non-halting property in $\mathbf{T}$. He stated that $c_{\mathbf{T}}$ is determined by $r_{\mathbf{T}}$. We call $r_{\mathbf{T}}$ *the Raatikainen characteristic constant of* $\mathbf{T}$.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 218–225, 2008.

The purpose of this paper is especially to show that $c_{\mathbf{T}}$ does not coincide with $r_{\mathbf{T}}$ contrary to Raatikainen's claim and to derive some mathematical properties of $c_{\mathbf{T}}$ and $r_{\mathbf{T}}$. First, we show that $r_{\mathbf{T}} \leq c_{\mathbf{T}}$ holds generally, but the converse not. By rearranging the enumeration of the Turing machines, we can make the difference between $r_{\mathbf{T}}$ and $c_{\mathbf{T}}$ arbitrarily large. Moreover, we prove that for two arithmetical theories $\mathbf{T}$, $\mathbf{T}'$ with a Π_1-sentence provable in $\mathbf{T}'$ but not in $\mathbf{T}$, there is an enumeration of the Turing machines such that $r_{\mathbf{T}} < r_{\mathbf{T}'}$ and $c_{\mathbf{T}} = c_{\mathbf{T}'}$.

Since these two characteristic constants are essentially what evaluate some information on Turing machines given by $\mathbf{T}$, the language of $\mathbf{T}$ is assumed to have some fixed way to express outputs of Turing machines. However, it is possible that a formal system with abundant axioms for outputs of Turing machines does not prove any facts on Kolmogorov complexity, because those two ideas may be expressed by different symbols which are not related by the axioms at all. For our purpose we consider formal systems defining Kolmogorov complexity naturally by Turing machines in this paper, while Raatikainen makes use of the recursion theorem and the soundness to dispense with this assumption. We confine our systems to first-order arithmetical theories, or formal systems in which first-order arithmetics can be interpreted, and assume that they are arithmetically sound, finitely-specified, and extending PA.

2 Arithmetizing Computability

Let PA denote the first-order Dedekind-Peano axioms, and $\mathcal{L}_A$ its language. Note that theories extending PA prove any Σ_1-sentence which is true in $\mathbb{N}$.

We consider Turing machines M of the following type. M has one semi-infinite tape, with the left end as the start cell. In one movement, it changes the state, writes a symbol on the tape, and makes the head move left or right, or stable. We assume the tape symbols are $0, 1$ and B, where B represents blank.

Definition 2.1. *Any ordered pair (x, y) of natural numbers can be coded by $\langle x, y\rangle = \frac{1}{2}(x + y)(x + y + 1) + x$.*

Tuples of natural numbers $(x_0, \ldots, x_n)$ can also be coded by a natural number $\langle x_0, x_1, \ldots, x_n\rangle$ defined inductively as follows for $n \geq 2$:

$$\langle x_0, x_1, \ldots, x_n\rangle = \langle x_0, \langle x_1, \ldots, x_n\rangle\rangle.$$

For $a = \langle x, y, n\rangle$, we define *Gödel β-function* as the function defined by

$$\beta(a, i) = x \bmod (y(i + 1) + 1) \text{ for each } i < n.$$

n is called the *length* of a, denoted by $|a|$. We also write $a[i]$ for $\beta(a, i)$.

Lemma 2.2. *Let $f : \mathbb{N} \to \mathbb{N}$ be any definable function in* PA. *Then*

$$\mathrm{PA} \vdash \forall n \exists a\, (\forall i < n \beta(a, i) = f(i) \wedge |a| = n).$$

For any $a_0, \ldots, a_{n-1} \in \mathbb{N}$, a natural number $a \in \mathbb{N}$ is called a *sequence number* for $a_0, \ldots, a_{n-1}$ if $\beta(a, i) = a_i$ for each natural number $i < n$.

Now we formulate the notion of Turing machine in PA.

Definition 2.3. *Suppose each cell in the tape is numbered 0, 1, 2, ... from the start cell to the right.*

(1) We code a transition rule of a Turing machines M by a tuple of the form

$$\langle q, s, q', s', m\rangle,$$

where $m \in \{L, R, S\}$. δ instructs that if the control is in state q over the cell with number s, it transitions into the state q' and writes s' on the cell, and moves or stay.

(2) A Turing machine M is coded by the sequence number

$$\langle N_Q, \delta, q_0, l_F\rangle,$$

where N_Q is the number of states of M, $q_0 < N_Q$ is the initial state of M, l_F is a sequence number coding the set of final states, and δ is a sequence number coding the set of transition rules.

(3) An instantaneous descriptions*, or an* ID*, is coded by the sequence number*

$$\langle q, t, h\rangle$$

such that $q < N_Q$ is a code of a state, t is a sequence number coding the contents of a contiguous finite sequence of cells covering every symbols appearing on the tape other than B, and h is the position of the head.

(4) A process *of M is coded by the sequence number of*

$$\langle ID_0, ID_1, \ldots, ID_l\rangle$$

of instantaneous descriptions such that the state of ID_0 is the initial state, and for each $i < l$, ID_{i+1} is obtained from ID_i by δ.

Next, we define some formulae describing movements of Turing machines.

There is a Δ_0-formula $\Psi_0(x, y)$ such that $\mathbb{N} \models \Psi_0(p, m)$ if and only if p is a number representing a process of the Turing machine coded by number m.

In order to describe the function represented by a Turing machine, we have to describe by a logical formula the relation between a natural number and the corresponding binary string representing it. The following are equivalent:

- $x = a_0 2^n + a_1 2^{n-1} + \ldots + a_{n-1} 2 + a_n$;
- There is a sequence x_0, x_1, ..., x_n such that $x_0 = a_0$, $x_i = 2x_{i-1} + a_i$ for $i = 1, \ldots, n$, and $x = x_n$.

Therefore, there is a Σ_1-formula $bin(x, t)$ which says that "t is a sequence number coding the binary representation of x."

So we have a Σ_1-formula $\Psi_1(m, y, z)$ which states that "z is an ID of the Turing machine coded by m and the contents of the tape is the binary representation of number y.

Using these formulae, we define a Σ_1-formula $\Psi_2(p, m, x)$ which states that "p is a number representing a valid sequence of ID of the Turing machine coded by m with the binary representation of x in the tape at the beginning.

Let φ_m the Turing machine coded by m and identify it with the computable (partial) function defined by φ_m. Then the statement "$\varphi_m(x) = y$" or "$\varphi_m(x) \downarrow y$" can be written by the formula

$$\begin{aligned}\exists p\,(\Psi_0(p,m) \wedge \exists n, t_0, t_2, i < p(n = |p| \wedge p[0] = \langle q_0, t_1, 0\rangle \\ \wedge p[n] = \langle q, t_2, 0\rangle \text{ for some } q \in F_m \\ \wedge bin(x, t_1) \wedge bin(y, t_2))\end{aligned}$$

Definition 2.4. *For two partial functions f and g on $\mathbb{N}$, $f \simeq g$ if and only if for any $x \in \mathbb{N}$,*

$$\begin{gathered}f(x) \text{ is defined} \iff g(x) \text{ is defined and} \\ f(x) = g(x) \text{ if both sides are defined.}\end{gathered}$$

We employ a fundamental result of recursion theory by Stephen C. Kleene.

Fact 2.5 (Kleene's recursion theorem). *If f is a total computable function, then there effectively exists a constant c such that $\varphi_{f(c)} \simeq \varphi_c$.*

The statement "Calculation of $\varphi_m(0)$ does not halt canonically", or "$\varphi_m(0)\uparrow$", can be represented by the formula

$$\forall p\,(\Psi_0(p,m) \wedge p[0] = \langle q_0, 0, 0\rangle \rightarrow p \text{ is not terminating canonically.})$$

Any effective enumeration of the computable functions (or recursive functions) can be represented by a universal Turing machine. Therefore, there is a computable total bijective function f (f is a "compiler" function) such that if m is a code ("program") of a function for given universal Turing machine then the partial function g coded by m with the given universal Turing machine satisfies $g \simeq \varphi_{f(m)}$.

Conversely, any computable bijective function f, $\varphi_{f(m)}$ $(m = 0, 1, \ldots)$ is an effective enumeration of any computable function.

We write φ^f_m for $\varphi_{f(m)}$. We also write $\varphi^f_m(x) \downarrow y$ if $\varphi_{f(m)}(x) \downarrow y$, and $\varphi^f_m(x) \uparrow$ if $\varphi_{f(m)}(x) \uparrow$. We will not specify the input if it is 0. Write $\varphi^f_m \uparrow$ for $\varphi^f_m(0) \uparrow$, $\varphi^f_m \downarrow y$ for $\varphi^f_m(0) \downarrow y$.

Note that any computable function is Σ_1-definable in PA.

Definition 2.6. *Let f be a Σ_1-definable bijective function in PA.*
Let $CT(d,m)$ be a formula saying that d is a code of ID representing a canonical termination of m, $tval(d,x)$ a formula saying that d is a code of ID with a tape value representing x.

We write $\varphi^f_m(x) \downarrow y$ for the formula

$$\exists p(\Psi_2(p, f(m), x) \wedge \exists l\,(|p| = l \wedge CT(p[l], m) \wedge tval(p[l], y))).$$

We write $\varphi^f_m \uparrow$ if the calculation of φ^f_m does not terminate canonically. This notion is equivalent to

$$\forall p(\Psi_2(p, f(m), x) \rightarrow \neg\exists l\,(|p| = l \wedge CT(p[l], m) \wedge tval(p[l], y))).$$

For a natural number n, the Kolmogorov complexity *of n with respect to f, denoted by $K^f(n)$, is the smallest natural number k such that $\varphi^f_k = n$. We can define $K^f(x) = y$ by "y is the smallest m such that $\exists p \Psi_2(p, f(m), 0, x)$."*

3 Characteristic Constants

Definition 3.1. *Let $\mathbf{T}$ be a formal system extending PA. We define two constants $\mathbf{c}_{f,\mathbf{T}}$ and $\mathbf{r}_{f,\mathbf{T}}$ as follows:*

$\mathbf{c}_{f,\mathbf{T}}$ is the smallest number k such that for any natural number n, $\mathbf{T} \nvdash K^f(n) > k$.

$\mathbf{r}_{f,\mathbf{T}}$ is the smallest number e such that $\varphi^f_e \uparrow$ and $\mathbf{T} \nvdash \varphi^f_e \uparrow$.

Theorem 3.2. *$c_{f,\mathbf{T}}$ and $r_{f,\mathbf{T}}$ exist for any sound, finitely-specified formal system $\mathbf{T}$ and a definable permutation f in* PA.

Proof. $c_{f,\mathbf{T}}$ exists, for there is a natural number c such that $\mathbf{T} \nvdash K^f(x) > c$. Define a total computable function i as follows. For each k, we have a Turing machine which searches the proof of $K^f(x) > k$ from $\mathbf{T}$ for some x and, if found, halts with output x. We can effectively obtain the index of this machine, $i(k)$. By recursion theorem, we can effectively find an index c such that $\varphi_c \simeq \varphi_{i(c)}$. If $\mathbf{T} \vdash K^f(x) > c$ for some natural number x, $\varphi_{i(c)} \simeq \varphi_c \downarrow x$, and hence $K^f(x) \leq c$. This contradicts the assumption of soundness of $\mathbf{T}$.

Next, we show that $r_{f,\mathbf{T}}$ exists. Consider a Turing machine $\varphi_{i(k)}$ which halts if $\mathbf{T} \vdash \varphi_k \uparrow$. By recursion theorem we have c such that $\varphi_{i(c)} \simeq \varphi_c$. φ_c does not halt, since if φ_c halts, φ_c does not halt by the soundness of $\mathbf{T}$. Hence, $\varphi_c \uparrow$. If $\mathbf{T} \vdash \varphi_c \uparrow$, then $\varphi_{i(c)}$, thus, φ_c halts. Contradiction. We have $\mathbf{T} \nvdash \varphi_c \uparrow$.

Next we argue the relation between $c_{f,\mathbf{T}}$ and $r_{f,\mathbf{T}}$. We first see $r_{f,\mathbf{T}} \leq c_{f,\mathbf{T}}$.

Lemma 3.3. *Let $\mathbf{T}$ be any formal system extending* PA*, and f any Σ_1-definable permutation, either $\mathbf{T} \vdash \varphi^f_i \uparrow$ or $\mathbf{T} \vdash \varphi^f_i \downarrow$ holds for any $i < r_{f,\mathbf{T}}$.*

Proof. Since $\mathbf{T}$ proves all Σ_1-sentences true in $\mathbb{N}$, if $\varphi^f_i \downarrow n, \mathbf{T} \vdash \varphi^f_i \downarrow n$. If $\varphi^f_i \uparrow$, by the minimality $r_{f,\mathbf{T}}$, $\mathbf{T} \vdash \varphi^f_i \uparrow$.

Theorem 3.4. *For any formal system $\mathbf{T}$ extending* PA *and any definable permutation f, $r_{f,\mathbf{T}} \leq c_{f,\mathbf{T}}$.*

Proof. Suppose $r_{f,\mathbf{T}} > c_{f,\mathbf{T}}$. Let $n \notin \{\varphi^f_0(0), \ldots, \varphi^f_{c_\mathbf{T}}(0)\}$ and $i \leq c_{f,\mathbf{T}}$. If $\mathbf{T} \vdash \varphi^f_i \downarrow k$, $\mathbf{T} \vdash \neg\varphi^f_i \downarrow n$. Otherwise, by lemma 3.3, $\mathbf{T} \vdash \varphi^f_i \uparrow$, $\mathbf{T} \vdash \neg\varphi^f_i \downarrow n$. Then $\mathbf{T} \vdash K^f(n) > c_{f,\mathbf{T}}$, a contradiction to the definition of $c_\mathbf{T}$.

Remark 3.5. For any definable permutation $\sigma : \mathbb{N} \to \mathbb{N}$ in PA, $PA \vdash \varphi^{f\circ\sigma}_m \simeq \varphi^f_{\sigma(m)}$.

However, $r_\mathbf{T} \geq c_\mathbf{T}$ does not necessarily hold. In fact, for a given formal system $\mathbf{T}$ we can enumerate the Turing machines so that $r_\mathbf{T} < c_\mathbf{T}$.

Lemma 3.6. *Let $f : \mathbb{N} \to \mathbb{N}$ be a bijective function Σ_1-definable in PA. There is a $g : \mathbb{N} \to \mathbb{N}$ Σ_1-definable in PA such that for any formal system* $\mathbf{T}$ *extending PA,*

(1) $\mathbf{T} \vdash \varphi^f_m \uparrow$ *if and only if* $\mathbf{T} \vdash \varphi^f_{g(m)} \uparrow$, *and*
(2) $\mathbf{T} \vdash \neg(\varphi^f_{g(m)} \downarrow 0)$.

Proof. We add transition rules to make the machine write 1 on the tape before termination. Let m be a number and $M = (Q, \Gamma, \delta, q_0, F)$ be the Turing machine coded by $f(m)$. Let $M' = (Q \cup \{q_f\}, \Gamma, \delta', q_0, \{q_f\})$ be a Turing machine such that q_f is a new state, and

$$\delta' = \delta \cup \{(q, b, q_f, 1, S) : b \in \{0, 1, B\}, q \in F\}.$$

Let m' be the code of M' and let $g(m) = f^{-1}(m')$.

We claim that g is the desired function.

First, we show (2), i.e., $PA \vdash \neg(\varphi^f_{g(m)} \downarrow 0)$. In case $\varphi^f_{g(m)} \uparrow$, then $\neg(\varphi^f_{g(m)} \downarrow 0)$. Otherwise, $\varphi^f_{g(m)} \downarrow y$ for some y. But $y \neq 0$ by the construction of M'.

Now, we show (1). Suppose $\mathbf{T} \vdash \varphi^f_m \uparrow$. Then $\mathbf{T} \vdash \forall p(\Psi_2(p, f(m), 0) \to \neg\exists l\,(|p| = l \wedge CT(p[l], m) \wedge tval(p[l], y)))$.

The following argument can be done in $\mathbf{T}$. Let p' be any natural number and assume that $\Psi_2(p', f(g(m)), x)$. Suppose that the last ID in p' has final state of M'. Truncate the last ID in p' and name it p. By the definition of M', p represents a valid calculation of "$f(m)$" which is canonically terminating. Therefore, we have $\varphi_{f(m)} \downarrow$, a contradiction. Hence, the last ID in p' does not have a final state of M'.

Conversely, suppose that

$$\mathbf{T} \vdash \forall p'(\Psi_2(p', m', 0) \to \neg\exists l\,(|p'| = l \wedge CT(p'[l], m') \wedge tval(p'[l], y))).$$

where $m' = f(g(m))$ is a code of the Turing machine M' described above.

Let p be any natural number and assume that $\Psi_2(p, f(m), x)$. If p represents a valid calculation of M terminating canonically, we can add an ID of M' to p to get a code p' of a valid calculation of M' terminating canonically. This contradicts with the assumption. Therefore, p does not represent a valid calculation of M terminating canonically.

Theorem 3.7. *Let* $\mathbf{T}$ *be any formal system extending PA. Let $f : \mathbb{N} \to \mathbb{N}$ be a bijective function Σ_1-definable in PA. Then there is a permutation σ on $\mathbb{N}$ definable in* PA *such that*

$$\mathbf{r}_{f,\mathbf{T}} = \mathbf{r}_{f\circ\sigma,\mathbf{T}} < \mathbf{c}_{f\circ\sigma,\mathbf{T}}.$$

Moreover, the difference between $\mathbf{r}_{f\circ\sigma,\mathbf{T}}$ and $\mathbf{c}_{f\circ\sigma,\mathbf{T}}$ can be arbitrarily large.

Proof. Let n be any natural number. We show that there is a permutation σ on $\mathbb{N}$ such that

$$\mathbf{r}_{f,\mathbf{T}} = \mathbf{r}_{f\circ\sigma,\mathbf{T}} < \mathbf{c}_{f\circ\sigma,\mathbf{T}}$$

and the difference of $\mathbf{r}_{f\circ\sigma,\mathbf{T}}$ and $\mathbf{c}_{f\circ\sigma,\mathbf{T}}$ is greater than n.

Let $g : \mathbb{N} \to \mathbb{N}$ be the function obtained in Lemma 3.6 with respect to f. Let σ be a permutation on $\mathbb{N}$ such that $\sigma(i) = g(i)$ for $i \leq \mathbf{r}_{f,\mathbf{T}} + n$. By Lemma 3.3, we have $\mathbf{T} \vdash \varphi_i^{f\circ\sigma} \downarrow$ or $\mathbf{T} \vdash \varphi_i^{f\circ\sigma} \uparrow$ for each $i < \mathbf{r}_{f,\mathbf{T}}$. But $\mathbf{T} \vdash \neg\varphi_i^{f\circ\sigma} \downarrow 0$ for $i \leq \mathbf{r}_{f,\mathbf{T}} + n$. Therefore, $\mathbf{T} \vdash K^{f\circ\sigma}(0) > \mathbf{r}_{f,\mathbf{T}} + n$. Hence, $\mathbf{r}_{f,\mathbf{T}} + n < \mathbf{c}_{f\circ\sigma,\mathbf{T}}$.

It is well-known that the truth definition of a Σ_n-formulas can be expressed by a Σ_{n+1}-formula. With a similar proof to this fact, we can show the following lemma:

Lemma 3.8. *Let $\psi(x)$ be a Δ_0-formula in $\mathcal{L}_A$. Then there is a code m_0 of a Turing machine such that*

$$PA \vdash \psi(x) \leftrightarrow \varphi_{m_0}(x) \downarrow 1,$$
$$PA \vdash \neg\psi(x) \leftrightarrow \varphi_{m_0}(x) \downarrow 0.$$

Lemma 3.9. *Let $\psi(x)$ be any Δ_0-formula in $\mathcal{L}_A$. Then there is a code m of a Turing machine such that*

$$PA \vdash \forall x\psi(x) \leftrightarrow \varphi_m \uparrow .$$

Proof. Let φ_{m_0} be a Turing machine we can get for $\psi(x)$ by Lemma 3.8.

Let φ_m be a Turing machine corresponding to the following C program:

```
while (ψ(x)) x++;
```

We explain φ_m more accurately. The initial state of φ_m is q_0. φ_m "saves" the value of x so that we can retrieve it later. Then φ_m evaluates $\psi(x)$ with the rules of φ_{m_0}. If the value is 0, then it enters the unique final state q_f and halts. If the value is 1, then it retrieves the value of x to an initial segment of the tape. Then it increments the value of x by 1 and enters the initial state q_0.

Now, we show the lemma. We are working in PA.

Suppose $\forall x\psi(x)$. We can show a formula in $\mathcal{L}_A$ expressing the following:

Claim 3.10 *For all n, there is a process of φ_m with input 0 such that the final ID has the initial state and the tape content is n.*

It is clear that the process with single ID $\langle q_0, bin(0), 0\rangle$ is the process for $n = 0$.

Assume $n \geq 1$. By induction hypothesis, there is a process of φ_m with input 0 such that the final ID is $\langle q_0, bin(n-1), 0\rangle$. Since $\forall x\psi(x)$, we have a process p_n for $\varphi_{m_0}(n-1) \downarrow 1$. We can concatenate process p_n and a process to perform $x++$ to the process already obtained. We have a process with the final ID $\langle q_0, bin(n), 0\rangle$. Therefore, we have the claim, and thus, $\varphi_m \uparrow$.

For the converse, suppose that $\varphi_m \uparrow$. We can prove a formula in $\mathcal{L}_A$ expressing the following claim:

Claim 3.11 *For all n, there is a process of φ_m with input 0 such that the final ID is $\langle q_0, bin(n), 0\rangle$, and if $n > 0$ then $\psi(n-1)$.*

We prove this by induction on n. It is obvious for $n = 0$.

Suppose $n \geq 1$. By the induction hypothesis, there is a process of φ_m with input 0 such that the final ID is $\langle q_0, bin(n-1), 0\rangle$.

By Lemma 3.8, there is a process for $\varphi_{m_0}(n-1) \downarrow 0$ or $\varphi_{m_0}(n-1) \downarrow 1$. In case $\varphi_{m_0}(n-1) \downarrow 0$, the control enters q_f and we have $\varphi_m \downarrow$. This contradicts our hypothesis. Therefore, there is a process for $\varphi_{m_0}(n-1) \downarrow 1$. Hence, we have $\psi(n-1)$ by Lemma 3.8. Now, we can increment the tape value x from $n-1$ to n. Therefore, we have the claim.

By the claim, we have $\forall x \psi(x)$.

By Lemma 3.9, we have the following theorem.

Theorem 3.12. *Let* $\mathbf{T}$ *and* $\mathbf{T}'$ *be sound, finitely-specified, formal systems extending* PA *such that* $\mathbf{T} < \mathbf{T}'$. *Then the following are equivalent:*

(1) There is a Π_1*-sentence* θ *such that* $\mathbf{T}' \vdash \theta$ *but* $\mathbf{T} \nvdash \theta$.
(2) There is an enumeration of the Turing machines such that $r_{\mathbf{T}} < r_{\mathbf{T}'}$.

Furthermore, we show that for two theories with a Π_1-gap, there is an enumeration of the Turing machines which makes them different in the Raatikainen constant but the same in the Chaitin characteristic constant.

Theorem 3.13. *Let* $\mathbf{T}, \mathbf{T}'$ *be sound, finitely-specified, formal systems extending* PA *with a* Π_1*-sentence provable in* $\mathbf{T}'$ *but not in* $\mathbf{T}$. *Then there exists an enumeration of the Turing machines such that* $c_{\mathbf{T}} = c_{\mathbf{T}'}$ *and* $r_{\mathbf{T}} < r_{\mathbf{T}'}$.

Proof. By Lemma 3.6 and Theorem 3.12, we can assume that $\mathbf{T} \nvdash \varphi_m \uparrow$, $\mathbf{T}' \vdash \varphi_m \uparrow$, and $\mathbf{T} \vdash \neg\varphi_m \downarrow 0$. By the same argument in Theorem 3.2, we take a c such that $\varphi_c \uparrow$ and $\mathbf{T}' \nvdash \neg\varphi_c \downarrow n$. Let f be a function such that $f(m) = 0$, $f(c) = 1$ and $f(x) = x$ otherwise. Since $\mathbf{T} \nvdash \varphi_0^f \uparrow$, $r_{\mathbf{T}}^f = 0$. By $\mathbf{T}' \vdash \varphi_0^f \uparrow$ and $\mathbf{T}' \nvdash \varphi_1^f \uparrow$, we have $r_{f,\mathbf{T}'} = 1$. Because $\mathbf{T} \vdash K^f(0) > 0$ and $T' \nvdash K^f(n) > 1$ for all n, $c_{f,\mathbf{T}} = c_{f,\mathbf{T}'} = 1$.

Remark 3.14. In the theorem above, we can make $c_{\mathbf{T}}$ arbitrarily large as in Theorem 3.7.

Acknowledgments. We wish to thank Norshasheema Shahidan for her valuable comments in discussions and on our draft.

References

1. Chaitin, G.J.: Information-theoretic limitations of formal systems. Journal of the ACM 21, 403–424 (1974)
2. van Lambalgen, M.: Algorithmic information theory. Journal of Symbolic Logic Logic 54, 1389–1400 (1989)
3. Raatikainen, P.: On interpreting Chaitin's incompleteness theorem. Journal of Philosophical Logic 27(6), 569–586 (1998)
4. Solovay, R.M.: A Version of Ω for Which ZFC Can Not Predict A Single Bit. CDMTCS-104 (1999)
5. Vitanyi, P., Li, M.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn. Springer, Heidelberg (1997)

Adversary Lower Bounds for Nonadaptive Quantum Algorithms

Pascal Koiran[1], Jürgen Landes[2], Natacha Portier[1], and Penghui Yao[3]

[1] LIP*, Ecole Normale Supérieure de Lyon, Université de Lyon
{Pascal.Koiran,Natacha.Portier}@ens-lyon.fr
[2] School of Mathematics, University of Manchester
juergen_landes@yahoo.de
[3] State Key Laboratory of Computer Science, Chinese Academy of Sciences
phyao1985@gmail.com

Abstract. We present general methods for proving lower bounds on the query complexity of nonadaptive quantum algorithms. Our results are based on the adversary method of Ambainis.

1 Introduction

In this paper we present general methods for proving lower bounds on the query complexity of nonadaptive quantum algorithms. A nonadaptive algorithm makes all its queries simultaneously. By contrast, an unrestricted (adaptive) algorithm may choose its next query based on the results of previous queries. In classical computing, classes of problems for which adaptivity does not help have been identified [4, 10] and it is known that this question is connected to a longstanding open problem [15] (see [10] for a more extensive discussion). In quantum computing, the study of nonadaptive algorithms seems especially relevant since some of the best known quantum algorithms (namely, Simon's algorithms and some other hidden subgroup algorithms) are nonadaptive. This is nevertheless a rather understudied subject in quantum computing.

The paper that is most closely related to the present work is [14] (and [8] is another related paper). In [14] the authors use an "algorithmic argument" (this is a kind of Kolmogorov argument) to give lower bounds on the nonadaptive quantum query complexity of ordered search, and of generalizations of this problem. The model of computation that they consider is less general than ours (more on this in section 2).

The two methods that have proved most successful in the quest for quantum lower bounds are the polynomial method (see for instance [5, 2, 11, 12]) and the adversary method of Ambainis. It is not clear how the polynomial method might take the nonadaptivity of algorithms into account. Our results are therefore based on the adversary method, in its weighted version [3]. We provide

* UMR 5668 ENS Lyon, CNRS, UCBL associée à l'INRIA. Work done when Landes and Yao were visiting LIP with financial support from the program MEST-CT-2004-504029 MATHLOGAPS.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 226–237, 2008.

two general lower bounds which yield optimal results for a number of problems: search in an ordered or unordered list, element distinctness, graph connectivity or bipartiteness. To obtain our first lower bound we treat the list of queries performed by a nonadaptive algorithm as one single "super query". We can then apply the adversary method to this 1-query algorithm. Interestingly, the lower bound that we obtain is very closely related to the lower bounds on *adaptive* probabilistic query complexity due to Aaronson [1], and to Laplante and Magniez [13]. Our second lower bound requires a detour through the so-called minimax (dual) method and is based on the fact that in a nonadaptive algorithm, the probability of performing any given query is independent of the input.

2 Definition of the Model

In the black box model, an algorithm accesses its input by querying a function x (the *black box*) from a finite set Γ to a (usually finite) set Σ. At the end of the computation, the algorithm decides to accept or reject x, or more generally produces an output in a (usually finite) set S'. The goal of the algorithm is therefore to compute a (partial) function $F : S \to S'$, where $S = \Sigma^\Gamma$ is the set of black boxes. For example, in the *Unordered Search* problem $\Gamma = [N] = \{1, \ldots, N\}$, $\Sigma = \{0, 1\}$ and F is the OR function: $F(x) = \bigvee_{1 \le i \le N} x(i)$.

Our second example is *Ordered Search*. The sets Γ and Σ are as in the first example, but F is now a partial function: we assume that the black box satisfies the promise that there exists an index i such that $x(j) = 1$ for all $j \ge i$, and $x(j) = 0$ for all $j < i$. Given such an x, the algorithm tries to compute $F(x) = i$.

A quantum algorithm $\mathcal{A}$ that makes T queries can be formally described as a tuple $(U_0, \ldots, U_T)$, where each U_i is a unitary operator. For $x \in S$ we define the unitary operator O_x (the "call to the black box") by $O_x|i\rangle|\varphi\rangle|\psi\rangle = |i\rangle|\varphi \oplus x(i)\rangle|\psi\rangle$. The algorithm $\mathcal{A}$ computes the final state $U_T O_x U_{T-1} \ldots U_1 O_x U_0 |0\rangle$ and makes a measurement of some of its qubits. The result of this measure is by definition the outcome of the computation of $\mathcal{A}$ on input x. For a given ε, the query complexity of a function F, denoted $Q_{2,\varepsilon}$, is the smallest query complexity of a quantum algorithm computing F with probability of error at most ε.

In the sequel, the quantum algorithms as described above will also be called ad*adaptive* to distinguish them from nonadaptive quantum algorithms. Such an algorithm performs all its queries at the same time. A nonadaptive black-box quantum algorithm $\mathcal{A}$ that makes T queries can therefore be defined by a pair (U, V) of unitary operators. For $x \in S$ we define the unitary operator O_x^T by

$$O_x^T |i_1, \ldots, i_T\rangle|\varphi_1, \ldots, \varphi_T\rangle|\psi\rangle = |i_1, \ldots, i_T\rangle|\varphi_1 \oplus x(i_1), \ldots, \varphi_T \oplus x(i_T)\rangle|\psi\rangle.$$

The algorithm $\mathcal{A}$ computes the final state $V O_x^T U |0\rangle$ and makes a measurement of some of its qubits. As in the adaptive case, the result of this measure is by definition the outcome of the computation of $\mathcal{A}$ on input x. For a given ε, the nonadaptive query complexity of a function F, denoted $Q_{2,\varepsilon}^{na}$, is

the smallest query complexity of a nonadaptive quantum algorithm computing F with probability of error at most ε. Our model is more general than the model of [14]. In that model, the $|\varphi\rangle$ register must remain set to 0 after application of U. After application of O_x^T, the content of this register is therefore equal to $|x(i_1), \ldots, x(i_T)\rangle$ rather than $|\varphi_1 \oplus x(i_1), \ldots, \varphi_T \oplus x(i_T)\rangle$.

It is easy to verify that for every nonadaptive quantum algorithm $\mathcal{A}$ of query complexity T there is an adaptive quantum algorithm $\mathcal{A}'$ that makes the same number of queries and computes the same function, so that $Q_{2,\varepsilon} \leq Q_{2,\varepsilon}^{na}$. Indeed, consider for every $k \in [T]$ the unitary operator A_k which maps the state $|i_1, \ldots, i_T\rangle|\varphi_1, \ldots, \varphi_T\rangle$ to

$$|i_k\rangle|\varphi_k\rangle|i_1, \ldots, i_{k-1}, i_{k+1}, \ldots i_T\rangle|\varphi_1, \ldots, \varphi_{k-1}, \varphi_{k+1}, \ldots, \varphi_T\rangle.$$

If the nonadaptive algorithm $\mathcal{A}$ is defined by the pair of unitary operators (U, V), then the adaptive algorithm $\mathcal{A}'$ defined by the tuple of unitary operators

$$(U_0, \ldots, U_T) = (A_1 U, A_2 A_1^{-1}, \ldots, A_T A_{T-1}^{T-1}, V A_T^{-1})$$

computes the same function.

3 A Direct Method

3.1 Lower Bound Theorem and Applications

The main result of this section is Theorem 3. It yields an optimal $\Omega(N)$ lower bound on the nonadaptive quantum query complexity of Unordered Search and Element Distinctness. First we recall the weighted adversary method of Ambainis and some related definitions. The constant $C_\varepsilon = (1-2\sqrt{\varepsilon(1-\varepsilon)})/2$ will be used throughout the paper.

Definition 1. *The function $w : S^2 \to R_+$ is a* **valid weight function** *if every pair $(x, y) \in S^2$ is assigned a non-negative weight $w(x, y) = w(y, x)$ that satisfies $w(x, y) = 0$ whenever $F(x) = F(y)$. We then define for all $x \in S$ and $i \in \Gamma$: $wt(x) = \sum_y w(x, y)$ and $v(x, i) = \sum_{y:\ x(i) \neq y(i)} w(x, y)$.*

Definition 2. *The pair $(w,\ w')$ is a* **valid weight scheme** *if:*

- *Every pair $(x, y) \in S^2$ is assigned a non-negative weight $w(x, y) = w(y, x)$ that satisfies $w(x, y) = 0$ whenever $F(x) = F(y)$.*
- *Every triple $(x, y, i) \in S^2 \times \Gamma$ is assigned a non-negative weight $w'(x, y, i)$ that satisfies $w'(x, y, i) = 0$ whenever $x(i) = y(i)$ or $F(x) = F(y)$, and $w'(x, y, i)w'(y, x, i) \geq w^2(x, y)$ for all x, y, i with $x(i) \neq y(i)$.*

We then define for all $x \in S$ and $i \in \Gamma$ $wt(x) = \sum_y w(x, y)$ and $v(x, i) = \sum_y w'(x, y, i)$.

Of course these definitions are relative to the partial function F.

Remark 1. Let w be a valid weight function and define w' such that if $x(i) \neq y(i)$ then $w'(x,y,i) = w(x,y)$ and $w'(x,y,i) = 0$ otherwise. Then (w,w') is a valid weight scheme and the functions wt and v defined for w in Definition 1 are exactly those defined for (w,w') in Definition 2.

Theorem 1 (weighted adversary method of Ambainis [3]). *Given a probability of error ε and a partial function F, the quantum query complexity $Q_{2,\varepsilon}(F)$ of F as defined in section 2 satisfies:*

$$Q_{2,\varepsilon}(F) \geq C_\varepsilon \max_{(w,w')\ valid} \min_{\substack{x,y,i \\ w(x,y)>0 \\ x(i)\neq y(i)}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,i)}}.$$

A probabilistic version of this lower bound theorem was obtained by Aaronson [1] and by Laplante and Magniez [13].

Theorem 2. *Fix the probability of error to $\varepsilon = 1/3$. The probabilistic query complexity $P_2(F)$ of F satisfies the lower bound $P_2(F) = \Omega(L_P(F))$, where*

$$L_P(F) = \max_w \min_{\substack{x,y,i \\ w(x,y)>0 \\ x(i)\neq y(i)}} \max\left(\frac{wt(x)}{v(x,i)}, \frac{wt(y)}{v(y,i)}\right).$$

Here w ranges over the set of valid weight functions.

We now state the main result of this section.

Theorem 3 (nonadaptive quantum lower bound, direct method). *The nonadaptive query complexity $Q^{na}_{2,\varepsilon}(F)$ of F satisfies the lower bound $Q^{na}_{2,\varepsilon}(F) \geq C^2_\varepsilon L^{na}_Q(F)$, where*

$$L^{na}_Q(F) = \max_w \max_{s\in S'} \min_{\substack{x,i \\ F(x)=s}} \frac{wt(x)}{v(x,i)}.$$

Here w ranges over the set of valid weight functions.

The following theorem, which is an unweighted adversary method for nonadaptive algorithm, is a consequence of Theorem 3.

Theorem 4. *Let $F : \Sigma^\Gamma \to \{0;1\}$, $X \subseteq F^{-1}(0)$, $Y \subseteq F^{-1}(1)$ and let $R \subset X \times Y$ be a relation such that:*

- *for every $x \in X$ there are at least m elements $y \in Y$ such that $(x,y) \in R$,*
- *for every $y \in Y$ there are at least m' elements $x \in X$ such that $(x,y) \in R$,*
- *for every $x \in X$ and every $i \in \Gamma$ there are at most l elements $y \in Y$ such that $(x,y) \in R$ and $x(i) \neq y(i)$,*
- *for every $y \in X$ and every $i \in \Gamma$ there are at most l' elements $x \in X$ such that $(x,y) \in R$ and $x(i) \neq y(i)$.*

Then $Q^{na}_{2,\varepsilon}(F) \geq C^2_\varepsilon \max(\frac{m}{l}, \frac{m'}{l'})$.

Proof. As in [3] and [13] we set $w(x,y) = w(y,x) = 1$ for all $(x,y) \in R$. Then $wt(x) \geq m$ for all $x \in A$, $wt(y) \geq m'$ for all $y \in B$, $v(x,i) \leq l$ and $v(y,i) \leq l'$. □

For the Unordered Search problem defined in Section 2 we have $m = N$ and $l = l' = m' = 1$. Theorem 4 therefore yields an optimal $\Omega(N)$ lower bound. The same bound can be obtained for the Element Distinctness problem. Here the set X of negative instances is made up of all one-to-one functions $x : [N] \to [N]$ and Y contains the functions $y : [N] \to [N]$ that are not one-to-one. We consider the relation R such that $(x,y) \in R$ if and only if there is a unique i such that $x(i) \neq y(i)$. Then $m = 2, l = 1, m' = N(N-1)$ and $l' = N - 1$.

As pointed out in [13], the $\Omega(\max(m/l, m'/l'))$ lower bound from Theorem 4 is also a lower bound on $P_2(F)$. There is a further connection:

Proposition 1. *For any function F we have $L_P(F) \geq L_Q^{na}(F)$. That is, ignoring constant factors, the lower bound on $P_2(F)$ given by Theorem 2 is at least as high as the lower bound on $Q_{2,\varepsilon}^{na}(F)$ given by Theorem 3.*

Proof. Pick a weight function w_Q which is optimal for the "direct method" of Theorem 3. That is, w_Q achieves the lower bound $L_Q^{na}(F)$ defined in this theorem. Let s_Q be the corresponding optimal choice for $s \in S'$. We need to design a weight function w_P which will show that $L_P(F) \geq L_Q^{na}(F)$. One can simply define w_P by: $w_P(x,y) = w_Q(x,y)$ if $F(x) = s_Q$ or $F(y) = s_Q$; $w_P(x,y) = 0$ otherwise. Indeed, for any i and any pair (x,y) such that $w_P(x,y) > 0$ we have $F(x) = s_Q$ or $F(y) = s_Q$, so that $\max(wt(x)/v(x,i), wt(y)/v(y,i)) \geq L_Q^{na}(F)$. □

The nonadaptive quantum lower bound from Theorem 3 is therefore rather closely connected to adaptive probabilistic lower bounds: it is sandwiched between the weighted lower bound of Theorem 2 and its unweighted $\max(m/l, m'/l')$ version. Proposition 1 also implies that Theorem 3 can at best prove an $\Omega(\log N)$ lower bound on the nonadaptive quantum complexity of Ordered Search. Indeed, by binary search the adaptive probabilistic complexity of this problem is $O(\log N)$. In section 4 we shall see that there is in fact a $\Omega(N)$ lower bound on the nonadaptive quantum complexity of this problem.

Remark 2. The connection between nonadaptive quantum complexity and adaptive probabilistic complexity that we have pointed out in the paragraph above is only a connection between the *lower bounds* on these quantities. Indeed, there are problems with a high probabilistic query complexity and a low nonadaptive quantum query complexity (for instance, Simon's problem [16, 10]). Conversely, there are problems with a low probabilistic query complexity and a high nonadaptive quantum query complexity (for instance, Ordered Search).

3.2 Proof of Theorem 3

As mentioned in the introduction, we will treat the tuple $(i_1, \ldots, i_k)$ of queries made by a nonadaptive algorithm as a single "super query" made by an ordinary quantum algorithm (incidentally, this method could be used to obtain lower bounds on quantum algorithm that make several rounds of parallel queries as in [8]). This motivates the following definition.

Definition 3. *Let Σ, Γ and S be as in section 2. Given an integer $k \geq 2$, we define:*

- ${}^k\Sigma = \Sigma^k$, ${}^k\Gamma = \Gamma^k$ *and* ${}^kS = \left(\Sigma^k\right)^{\Gamma^k}$.
- *To the black box $x \in S$ we associate the "super box" ${}^kx \in {}^kS$ such that if $I = (i_1, \ldots, i_k) \in \Gamma^k$ then ${}^kx(I) = (x(i_1), \ldots, x(i_k))$.*
- ${}^kF({}^kx) = F(x)$.
- *If w is a weight function for F we define a weight function W for ${}^kF$ by $W({}^kx, {}^ky) = w(x, y)$.*

Assume for instance that $\Sigma = \{0; 1\}$, $\Gamma = [3]$, $k = 2$, and that x is defined by: $x(1) = 0$, $x(2) = 1$ and $x(3) = 0$. Then we have ${}^2x(1,1) = (0,0)$, ${}^2x(1,2) = (0,1)$, ${}^2x(1,3) = (0,0)$ ...

Lemma 1. *If w is a valid weight function for F then W is a valid weight function for ${}^kF$ and the minimal number of queries of a quantum algorithm computing ${}^kF$ with error probability ε satisfies:*

$$Q_{2,\varepsilon}({}^kF) \geq C_\varepsilon \cdot \min_{\substack{{}^kx, {}^ky, I \\ W({}^kx, {}^ky) > 0 \\ {}^kx(I) \neq {}^ky(I)}} \sqrt{\frac{WT({}^kx)WT({}^ky)}{V({}^kx, I)V({}^ky, I)}}.$$

Proof. Every pair $(x, y) \in S^2$ is assigned a non-negative weight $W({}^kx, {}^ky) = W({}^ky, {}^kx) = w(x, y) = w(y, x)$ that satisfies $W({}^kx, {}^ky) = 0$ whenever $F(x) = F(y)$. Thus we can apply Theorem 1 and we obtain the announced lower bound. □

Lemma 2. *Let x be a black-box and w a weight function. For any integer k and any tuple $I = (i_1, \ldots, i_k)$ we have*

$$\frac{WT({}^kx)}{V({}^kx, I)} \geq \frac{1}{k} \min_{j \in [k]} \frac{wt(x)}{v(x, i_j)}.$$

Proof. Let $m = \min_{j \in [k]} \frac{wt(x)}{v(x, i_j)}$. We have $WT({}^kx) = wt(x)$ and:

$$\begin{aligned} V({}^kx, I) &= \sum_{{}^ky : {}^kx(i) \neq {}^ky(i)} W({}^kx, {}^ky) \\ &\leq \sum_{y : x(i_1) \neq y(i_1)} w(x, y) + \cdots + \sum_{y : x(i_k) \neq y(i_k)} w(x, y) \\ &= v(x, i_1) + \cdots + v(x, i_k) \leq k \max_{j \in [k]} v(x, i_j). \end{aligned}$$

□

Lemma 3. *If w is a valid weight function:*

$$Q^{na}_{2,\varepsilon}(F) \geq C^2_\varepsilon \min_{\substack{x, y \\ F(x) \neq F(y)}} \max\left(\min_i \frac{wt(x)}{v(x, i)}, \min_i \frac{wt(y)}{v(y, i)}\right).$$

Proof. Let w be an arbitrary valid weight function and k be an integer such that

$$k < C_\varepsilon^2 \min_{\substack{x,y \\ F(x)\neq F(y)}} \max\left(\min_i \frac{wt(x)}{v(x,i)}, \min_i \frac{wt(y)}{v(y,i)}\right).$$

We show that an algorithm computing ${}^kF$ with probability of error $\leq \varepsilon$ must make strictly more one than query to the "super box" ${}^kx$. This will prove that for every such k we have $Q_{2,\varepsilon}^{na}(F) > k$ and thus our result.

For every x and I we have

$$\frac{WT({}^kx)}{V({}^kx, I)} \geq 1$$

and thus by lemma 2 for every x, y and $I = (i_1, \ldots, i_k)$:

$$\begin{aligned}
\frac{WT({}^kx)}{V({}^kx,I)}\frac{WT({}^ky)}{V({}^kx,I)} &= \min\left(\frac{WT({}^kx)}{V({}^kx,I)}, \frac{WT({}^ky)}{V({}^kx,I)}\right)\max\left(\frac{WT({}^kx)}{V({}^kx,I)}, \frac{WT({}^ky)}{V({}^kx,I)}\right) \\
&\geq \max\left(\frac{WT({}^kx)}{V({}^kx,I)}, \frac{WT({}^ky)}{V({}^kx,I)}\right) \\
&\geq \frac{1}{k}\max\left(\min_{j\in[k]} \frac{wt(x)}{v(x,i_j)}, \min_{l\in[k]} \frac{wt(y)}{v(x,i_l)}\right).
\end{aligned}$$

In order to apply Lemma 1 we observe that:

$$\begin{aligned}
\min_{\substack{{}^kx,{}^ky,I \\ W({}^kx,{}^ky)>0 \\ {}^kx(I)\neq{}^ky(I)}} \frac{WT({}^kx)WT({}^ky)}{V({}^kx,I)V({}^ky,I)} &\geq \frac{1}{k} \min_{\substack{x,y,i_1,\ldots,i_k \\ w(x,y)>0 \\ \exists m\ x(i_m)\neq y(i_m)}} \max\left(\min_{j\in[k]} \frac{wt(x)}{v(x,i_j)}, \min_{l\in[k]} \frac{wt(y)}{v(x,i_l)}\right) \\
&\geq \frac{1}{k} \min_{\substack{x,y \\ F(x)\neq F(y)}} \max\left(\min_i \frac{wt(x)}{v(x,i)}, \min_i \frac{wt(y)}{v(x,i)}\right)
\end{aligned}$$

By hypothesis on k, this expression is greater than $1/C_\varepsilon^2$. Thus according to Lemma 1 we have $Q_{2,\varepsilon}({}^kF) > 1$, and $Q_{2,\varepsilon}^{na}(F) > k$. □

We can now complete the proof of Theorem 3. Suppose without loss of generality that $F(S) = [m]$ and define for every $l \in [m]$:

$$a_l = C_\varepsilon^2 \min_{\substack{x,i \\ F(x)=l}} \frac{wt(x)}{v(x,i)}.$$

Suppose also without loss of generality that $a_1 \leq \cdots \leq a_m$. It follows immediately from the definition that

$$a_2 = C_\varepsilon^2 \min_{\substack{x,y \\ F(x)\neq F(y)}} \max\left(\min_i \frac{wt(x)}{v(x,i)}, \min_i \frac{wt(y)}{v(x,i)}\right),$$

and

$$a_m = C_\varepsilon^2 \max_{l \in F(S)} \min_{\substack{x,i \\ F(x)=l}} \frac{wt(x)}{v(x,i)}.$$

By Lemma 3 we have $Q_{2,\varepsilon}^{na}(F) \geq a_2$, but we would like to show that $Q_{2,\varepsilon}^{na}(F) \geq a_m$. We proceed by reduction from the case when there are only two classes (i.e., $m = 2$). Let G be defined by

$$G(1) = \cdots = G(m-1) = 1$$

and $G(m) = m$. Applying Lemma 3 to GoF, we obtain that $Q_{2,\varepsilon}^{na}(GoF) \geq a_m$. But because the function GoF is obviously easier to compute than F, we have $Q_{2,\varepsilon}^{na}(F) \geq Q_{2,\varepsilon}^{na}(GoF)$ and thus $Q_{2,\varepsilon}^{na}(F) \geq a_m$ as desired.

4 From the Dual to the Primal

Our starting point in this section is the minimax method of Laplante and Magniez [13, 17] as stated in [9]:

Theorem 5. *Let $p : S \times \Sigma \to \mathbb{R}^+$ be the set of $|S|$ probability distributions such that $p_x(i)$ is the average probability of querying i on input x, where the average is taken over the whole computation of an algorithm $\mathcal{A}$. Then the query complexity of $\mathcal{A}$ is greater or equal to:*

$$C_\varepsilon \max_{\substack{x,y \\ F(x)\neq F(y)}} \frac{1}{\sum\limits_{\substack{i \\ x(i)\neq y(i)}} \sqrt{p_x(i)p_y(i)}}.$$

Theorem 5 is the basis for the following lower bound theorem. It can be shown that up to constant factors, the lower bound given by Theorem 6 is always as good as the lower bound given by Theorem 3.

Theorem 6 (nonadaptive quantum lower bound, primal-dual method). *Let $F : S \to S'$ be a partial function, where as usual $S = \Sigma^\Gamma$ is the set of black-box functions. Let*

$$DL(F) = \min_p \max_{\substack{x,y \\ F(x)\neq F(y)}} \frac{1}{\sum\limits_{\substack{i \\ x(i)\neq y(i)}} p(i)}$$

and

$$PL(F) = \max_w \frac{\sum\limits_{x,y} w(x,y)}{\max\limits_i \sum\limits_{\substack{x,y \\ x_i \neq y_i}} w(x,y)}$$

where the min *in the first formula is taken over all probability distributions p over Γ, and the* max *in the second formula is taken over all valid weight functions w. Then $DL(F) = PL(F)$ and we have the following nonadaptive query complexity lower bound:*

$$Q_{2,\varepsilon}(F) \geq C_\varepsilon DL(F) = C_\varepsilon PL(F).$$

Proof. We first show that $Q_{2,\varepsilon}(F) \geq C_\varepsilon DL(F)$. Let $\mathcal{A}$ be a nonadaptive quantum algorithm for F. Since $\mathcal{A}$ is nonadaptive, the probability $p_x(i)$ of querying i on input x is independent of x. We denote it by $p(i)$. Theorem 5 shows that the query complexity of $\mathcal{A}$ is greater or equal to

$$C_\varepsilon \max_{\substack{x,y \\ F(x)\neq F(y)}} \frac{1}{\sum\limits_{\substack{i \\ x(i)\neq y(i)}} p(i)}.$$

The lower bound $Q_{2,\varepsilon}(F) \geq C_\varepsilon DL(F)$ follows by minimizing over p.

It remains to show that $DL(F) = PL(F)$. Let

$$L(F) = \min_p \max_{\substack{x,y \\ F(x)\neq F(y)}} \sum_{\substack{i \\ x(i)=y(i)}} p(i).$$

We observe that $L(F)$ is the optimal solution of the following linear program: minimize μ subject to the constraints

$$\forall x, y \text{ such that } f(x) \neq f(y) : \ \mu - \sum_{\substack{i \\ x(i)\neq y(i)}} p(i) \geq 0,$$

$$\text{and to the constraints } \sum_{i=1}^{N} p(i) = 1 \text{ and } \forall i \in [N] : \ p(i) \geq 0.$$

Clearly, its solution set is nonempty. Thus $L(f)$ is the optimal solution of the dual linear program: maximize ν subject to the constraints

$$\forall i \in [N] : \ \nu - \sum_{\substack{x,y \\ x_i=y_i}} w(x,y) \leq 0$$

$$\forall x, y : \ w(x,y) \geq 0, \text{ and } w(x,y) = 0 \text{ if } F(x) = F(y)$$

$$\text{and to the constraint} \sum_{x,y} w(x,y) = 1.$$

Hence $L(F) = \max\limits_w \min\limits_i \dfrac{\sum\limits_{x_i=y_i} w(x,y)}{\sum\limits_{x,y} w(x,y)}$ and $DL(F) = \frac{1}{1-L(F)} = PL(F)$. □

4.1 Application to Ordered Search and Connectivity

Proposition 1. *For any error bound $\varepsilon \in [0, \frac{1}{2})$ we have*

$$Q^{na}_{2,\varepsilon}(\textit{Ordered Search}) \geq C_\varepsilon(N-1).$$

Proof. Consider the weight function $w(x,y) = \begin{cases} 1 & \text{if } |F(y) - F(x)| = 1, \\ 0 & \text{otherwise.} \end{cases}$ Thus $w(x,y) = 1$ when the leftmost 1's in x and y are adjacent. Hence $\sum_{x,y} w(x,y) = 2(N-2)+2$. Moreover, if $w(x,y) \neq 0$ and $x_i \neq y_i$ then $\{F(x), F(y)\} = \{i, i+1\}$. Therefore, $\max_i \sum_{\substack{x,y \\ x_i \neq y_i}} w(x,y) = 2$ and the result follows from Theorem 6. □

Our second application of Theorem 6 is to the graph connectivity problem. We consider the adjacency matrix model: $x(i,j) = 1$ if ij is an edge of the graph. We consider undirected, loopless graph so that we can assume $j < i$. For a graph on n vertices, the black box x therefore has $N = n(n-1)/2$ entries. We denote by G_x the graph represented by x.

Theorem 7. *For any error bound $\varepsilon \in [0, \frac{1}{2})$, we have*

$$Q^{na}_{2,\varepsilon}(\textit{Connectivity}) \geq C_\varepsilon n(n-1)/8.$$

Proof. We shall use essentially the same weight function as in ([6], Theorem 8.3). Let X be the set of all adjacency matrices of a unique cycle, and Y the set of all adjacency matrices with exactly two (disjoint) cycles. For $x \in X$ and $y \in Y$, we set $w(x,y) = 1$ if there exist 4 vertices $a, b, c, d \in [n]$ such that the only differences between G_x and G_y are that:

1. ab, cd are edges in G_x but not in G_y.
2. ac, bd are edges in G_y but not in G_x.

We claim that

$$\max_{ij} \sum_{\substack{x \in X, y \in Y \\ x(i,j) \neq y(i,j)}} w(x,y) = \frac{8}{n(n-1)} \sum_{\substack{x \in X, y \in Y \\ x(i,j) \neq y(i,j)}} w(x,y). \tag{1}$$

The conclusion of Theorem 7 will then follow directly from Theorem 6. By symmetry, the function that we are maximizing on the left-hand side of (1) is in fact independent of the edge ij. We can therefore replace the max over ij by an average over ij: the left-hand side is equal to

$$\frac{1}{N} \sum_{x \in X, y \in Y} w(x,y) |\{ij;\ x(i,j) \neq y(i,j)\}|.$$

Now, the condition $x(i,j) \neq y(i,j)$ holds true if and only if ij is one of the 4 edges ab, cd, ac, bd defined at the beginning of the proof. This finishes the proof of (1), and of Theorem 7. □

A similar argument can be used to show that testing whether a graph is bipartite also requires $\Omega(n^2)$ queries.

5 Some Open Problems

For the "1-to-1 versus 2-to-1" problem, one would expect a higher quantum query complexity in the nonadaptive setting than in the adaptive setting. This may be difficult to establish since the adaptive lower bound [2] is based on the polynomial method. Hidden Translation [7] (a problem closely connected to the dihedral hidden subgroup problem) is another problem of interest. No lower bound is known in the adaptive setting, so it would be natural to look first for a nonadaptive lower bound. Finally, one would like to identify some classes of problems for which adaptivity does not help quantum algorithms.

Acknowledgements. This work has benefited from discussions with Sophie Laplante, Troy Lee, Frédéric Magniez and Vincent Nesme.

References

1. Aaronson, S.: Lower bounds for local search by quantum arguments. In: Proc. STOC 2004, pp. 465–474. ACM Press, New York (2004)
2. Aaronson, S., Shi, Y.: Quantum Lower Bounds for the Collision and the Element Distinctness Problems. Journal of the ACM 51(4), 595–605 (2004)
3. Ambainis, A.: Polynomial degree vs. quantum query complexity. J. Comput. Syst. Sci. 72(2), 220–238 (2006)
4. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Sampling Algorithms: lower bounds and applications. In: Proc. STOC 2001, pp. 266–275. ACM Press, New York (2001)
5. Beals, R., Buhrman, H., Cleve, R., Mosca, M., de Wolf, R.: Quantum lower bounds by polynomials. Journal of the ACM 48(4), 778–797 (2001)
6. Dürr, C., Heiligman, M., Høyer, P., Mhalla, M.: Quantum query complexity of some graph problems. SIAM J. Comput. 35(6), 1310–1328 (2006)
7. Friedl, K., Ivanyos, G., Magniez, F., Santha, M., Sen, P.: Hidden translation and orbit coset in quantum computing. In: STOC 2003: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (2003)
8. Grover, L.K., Radhakrishnan, J.: Quantum search for multiple items using parallel queries. arXiv (2004)
9. Hoyer, P., Spalek, R.: Lower bounds on quantum query complexity. EATCS Bulletin 87, 78–103 (2005)
10. Koiran, P., Nesme, V., Portier, N.: On the probabilistic query complexity of transitively symmetric problems, `http://perso.ens-lyon.fr/pascal.koiran`
11. Koiran, P., Nesme, V., Portier, N.: A quantum lower bound for the query complexity of Simon's problem. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1287–1298. Springer, Heidelberg (2005)
12. Koiran, P., Nesme, V., Portier, N.: The quantum query complexity of abelian hidden subgroup problems. Theoretical Computer Science 380, 115–126 (2007)
13. Laplante, S., Magniez, F.: Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. SIAM journal on Computing (to appear)
14. Nishimura, H., Yamakami, T.: An algorithmic argument for nonadaptive query complexity lower bounds on advised quantum computation (extended abstract). In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 827–838. Springer, Heidelberg (2004)

15. Rosenberg, A.L.: On the time required to check properties of graphs: A problem. SIGACT News, 15–16 (1973)
16. Simon, D.R.: On the power of quantum computation. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 116–123 (1994)
17. Spalek, R., Szegedy, M.: All quantum adversary methods are equivalent. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1299–1311. Springer, Heidelberg (2005)

On Second-Order Monadic Groupoidal Quantifiers*

Juha Kontinen[1] and Heribert Vollmer[2]

[1] Department of Mathematics and Statistics, University of Helsinki, P.O. Box 68, FI-00014 University of Helsinki, Finland
[2] Institut für Theoretische Informatik, Universität Hannover, Appelstraße 4, 30167 Hannover, Germany

Abstract. We study logics defined in terms of so-called second-order monadic groupoidal quantifiers. These are generalized quantifiers defined by groupoid word-problems or equivalently by context-free languages. We show that, over strings with built-in arithmetic, the extension of monadic second-order logic by all second-order monadic groupoidal quantifiers collapses to its fragment mon-Q^1_{Grp}FO. We also show a variant of this collapse which holds without built-in arithmetic. Finally, we relate these results to an open question regarding the expressive power of finite leaf automata with context-free leaf languages.

1 Introduction

We study logics defined in terms of so-called second-order monadic groupoidal quantifiers. These are generalized quantifiers defined by groupoid word-problems or equivalently by context-free languages. A *groupoid* is a finite multiplication table with an identity element. For a fixed groupoid G, each $S \subseteq G$ defines a G-word-problem, i.e., a language $\mathcal{W}(S, G)$ composed of all words w, over the alphabet G, that can be bracketed in such a way that w multiplies out to an element of S. Groupoid word-problems relate to context-free languages in the same way as monoid word-problems relate to regular languages: Every such word-problem is context-free, and every context-free language is a homomorphic pre-image of a groupoid word-problem (this result is credited to Valiant in [2]).

Monoidal quantifiers are generalized quantifiers defined by monoid word-problems or equivalently by regular languages. It was known [1] that first-order logic with unnested unary monoidal quantifiers characterizes the class of regular languages. In [6] this was extended to show the following

$$\exists\mathrm{SOM} = \text{mon-}Q^1_{\mathrm{Mon}}\mathrm{FO} = \mathrm{FO}(\text{mon-}Q^1_{\mathrm{Mon}}) = \mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Mon}}) = \mathrm{REG} \ . \quad (1)$$

In (1), $\exists$SOM stands for existential second-order monadic logic and REG denotes the class of regular languages. The class mon-Q^1_{Mon}FO is the class of all languages

* The first author was partially supported by grant 106300 of the Academy of Finland and the Vilho, Yrjö and Kalle Väisälä Foundation. The second author was partially supported by DFG grant VO 630/6-1.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 238–248, 2008.

describable by applying a specific monadic second-order monoidal quantifier Q_L to an appropriate tuple of formulas without further occurrences of second-order quantifiers. On the other hand, in FO(mon-Q^1_{Mon}) arbitrary nestings of monoidal quantifiers is allowed, analogously to SOM(mon-Q^1_{Mon}) in which the base logic is second-order monadic logic.

We see that with monoidal quantifiers the situation is clear-cut, i.e., formulas with monadic second-order monoidal quantifiers cannot define non-regular languages. Note that over strings with built-in arithmetic the classes in (1) are presumably not equal, e.g., $\exists\mathrm{SOM} \subseteq \mathrm{NP}$ and already in FO(mon-Q^1_{Mon}) PSPACE-complete languages can be defined by a similar argument as in Proposition 1. Similarly, the equivalences in (1) do not hold if non-monadic quantifiers are also allowed (under some reasonable complexity-theoretic assumptions).

In [6] it was asked what is the relationship of the corresponding logics if monoidal quantifiers are replaced by groupoidal quantifiers. In this paper we address this question and show the following:

$$\text{mon-}Q^1_{\mathrm{Grp}}\mathrm{FO}(+,\times) = \mathrm{FO}(\text{mon-}Q^1_{\mathrm{Grp}}) = \mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Grp}},+,\times)\ . \qquad (2)$$

It is interesting to note that in the case of groupoidal quantifiers the collapse of the logics happens in the presence of built-in arithmetic.

In Sect. 4 we consider groupoidal quantifiers with a slight change in their semantics (notation $Q^\star_L$). We show that the analogue of (2) also holds in this case. It turns out that (2) remains valid even if we drop the built-in predicates $+$ and $\times$ from mon-$Q^\star_{\mathrm{Grp}}\mathrm{FO}(+,\times)$. Finally, we relate these results to an open question regarding the expressive power of finite leaf automata with context-free leaf languages.

2 Generalized Quantifiers

We follow standard notation for monadic second-order logic with linear order, see, e.g., [14]. We mainly restrict our attention to *string signatures*, i.e., signatures of the form $\langle P_{a_1},\ldots,P_{a_s}\rangle$, where all the predicates P_{a_i} are unary, and in every structure $\mathcal{A}$, $\mathcal{A} \models P_{a_i}(j)$ iff the jth symbol in the input is the letter a_i. Such structures are thus words over the alphabet $\{a_1,\ldots,a_s\}$. We assume that the universe of each structure $\mathcal{A}$ is of the form $\{0,\ldots,n-1\}$ and that the logic's linear order symbol refers to numerical order on $\{0,\ldots,n-1\}$. For technical reasons to be motivated shortly, we also assume that every alphabet has a built-in linear order, and we write alphabets as sequences of symbols to indicate that order, e.g., in the above case we write $(a_1,\ldots,a_s)$.

Our basic formulas are built from first- and second-order variables in the usual way, using the Boolean connectives $\{\wedge,\vee,\neg\}$, the relevant predicates P_{a_i} together with $\{=,<\}$, the constants min and max, the first- and second-order quantifiers $\{\exists,\forall\}$, and parentheses.

SOM is the class of all languages definable using formulas as just described. (The letters SOM stand for second order monadic logic; in the literature, this logic is sometimes denoted by MSO.) FO is the subclass of SOM restricted to

languages definable by first-order formulas. It is known [12] that FO is equal to the class of star-free regular languages and that SOM equals the class REG of regular languages (see [4,3,15]). Sometimes we assume that our structures are also equipped with the built-in predicates + and ×. This assumption is signalled, e.g., by the notation FO(+, ×).

Next, we extend logics in terms of generalized quantifiers. The Lindström quantifiers of Def. 1 are precisely what has been referred to as "Lindström quantifiers on strings" [5]. The original more general definition [11] uses transformations to arbitrary structures, not necessarily of string signature.

Definition 1. *Consider a language L over an alphabet $\Sigma = (a_1, a_2, \ldots, a_s)$. Such a language gives rise to a Lindström quantifier Q_L, that may be applied to any sequence of $s-1$ formulas as follows:*

Let $\overline{x}$ be a k-tuple of variables. We assume the lexical ordering on $\{0,1,\ldots,n-1\}^k$, and we write $\overline{x}^{(1)} < \overline{x}^{(2)} < \cdots < \overline{x}^{(n^k)}$ for the sequence of potential values taken on by $\overline{x}$. The k-ary Lindström quantifier Q_L *binding $\overline{x}$ takes a meaning if $s-1$ formulas, each having as free variables the variables in $\overline{x}$ (and possibly others), are available. Let $\varphi_1(\overline{x})$, $\varphi_2(\overline{x})$, …, $\varphi_{s-1}(\overline{x})$ be these $s-1$ formulas. Then $Q_L\overline{x}\big[\varphi_1(\overline{x}), \varphi_2(\overline{x}), \ldots, \varphi_{s-1}(\overline{x})\big]$ holds on a string $w = w_1 \cdots w_n$, iff the word of length n^k whose ith letter, $1 \le i \le n^k$, is*

$$\begin{cases} a_1 \text{ if } w \models \varphi_1(\overline{x}^{(i)}), \\ a_2 \text{ if } w \models \neg\varphi_1(\overline{x}^{(i)}) \wedge \varphi_2(\overline{x}^{(i)}), \\ \quad \vdots \\ a_s \text{ if } w \models \neg\varphi_1(\overline{x}^{(i)}) \wedge \neg\varphi_2(\overline{x}^{(i)}) \wedge \cdots \wedge \neg\varphi_{s-1}(\overline{x}^{(i)}), \end{cases}$$

belongs to L.

As an example, take $s = 2$ and consider $L_\exists =_{\text{def}} 0^*1(0+1)^*$; then $Q_{L_\exists}$ is the usual first-order existential quantifier. Similarly, the universal quantifier can be expressed using the language $L_\forall =_{\text{def}} 1^*$. The quantifiers $Q_{L_{\text{mod }p}}$ for $p > 1$ are known as modular counting quantifiers [14].

In this paper we are especially interested in quantifiers defined by groupoid word problems. The following definition is due to Bédard, Lemieux, and McKenzie [2]:

Definition 2. *A* groupoidal quantifier *is a Lindström quantifier Q_L where L is a word-problem of some finite groupoid.*

Usage of groupoidal quantifiers in our logical language is signalled by Q_{Grp}. The class Q_{Grp}FO is the class of all languages definable by applying a single groupoidal quantifier to an appropriate tuple of FO formulas. The class FO(Q_{Grp}) is defined analogously, but allowing groupoidal quantifiers to be used as any other quantifier would (i.e., allowing arbitrary nesting).

Second-order Lindström quantifiers on strings were introduced in [5]. Here, we are mainly interested in those binding only set variables, so called *monadic quantifiers*. For each language L we define two monadic quantifiers Q_L and $Q_L^\star$

with slightly different interpretations. It turns out that the interpretation Q_L, which was used in [6], is natural in the context of finite automata. On the other hand, the quantifier $Q_L^\star$ is the exact second-order analogue of the corresponding first-order quantifier Q_L.

Definition 3. *Consider a language L over an alphabet $\Sigma = (a_1, a_2, \ldots, a_s)$. Let $\overline{X} = (X_1, \ldots, X_k)$ be a k-tuple of unary second-order variables, i.e., set variables. There are 2^{nk} different instances (assignments) of $\overline{X}$. We assume the following ordering on those instances: Let each instance of a single X_i be encoded by a bit string $s_0^i \cdots s_{n-1}^i$ with the meaning $s_j^i = 1 \iff j \in X_i$. Then*

i) we encode an instance of $\overline{X}$ by the bit string

$$s_0^1 s_0^2 \cdots s_0^k s_1^1 s_1^2 \cdots s_1^k \cdots s_{n-1}^1 s_{n-1}^2 \cdots s_{n-1}^k$$

and order the instances lexicographically by their codes.

ii) we encode an instance of $\overline{X}$ by the bit string

$$s_0^1 s_1^1 \cdots s_{n-1}^1 s_0^2 s_1^2 \cdots s_{n-1}^2 \cdots s_0^k s_1^k \cdots s_{n-1}^k$$

and order the instances lexicographically by their codes.

The monadic second-order Lindström quantifier Q_L *(respectively $Q_L^\star$) binding $\overline{X}$ takes a meaning if $s-1$ formulas, each having free variables $\overline{X}$, are available. Let $\varphi_1(\overline{X})$, $\varphi_2(\overline{X})$, …, $\varphi_{s-1}(\overline{X})$ be these $s-1$ formulas. Then $\varphi = Q_L\overline{X}\big[\varphi_1(\overline{X}), \varphi_2(\overline{X}), \ldots, \varphi_{s-1}(\overline{X})\big]$ holds on a string $w = w_1 \cdots w_n$, iff the word of length 2^{nk} whose ith letter, $1 \le i \le 2^{nk}$, is*

$$\begin{cases} a_1 \text{ if } w \models \varphi_1(\overline{X}^{(i)}), \\ a_2 \text{ if } w \models \neg\varphi_1(\overline{X}^{(i)}) \wedge \varphi_2(\overline{X}^{(i)}), \\ \quad\vdots \\ a_s \text{ if } w \models \neg\varphi_1(\overline{X}^{(i)}) \wedge \neg\varphi_2(\overline{X}^{(i)}) \wedge \cdots \wedge \neg\varphi_{s-1}(\overline{X}^{(i)}), \end{cases}$$

belongs to L. Above, $\overline{X}^{(1)} < \overline{X}^{(2)} < \cdots < \overline{X}^{(2^{nk})}$ denotes the sequence of all instances ordered as in i). The notation $Q_L^\star$ is used when the ordering of the instances is as in ii).

Again, taking as examples the languages $L_\exists$ and $L_\forall$, we obtain the usual second-order existential and universal quantifiers. Note that for $L \in \{L_\exists, L_\forall\}$ the quantifiers Q_L and $Q_L^\star$ are "equivalent". This is due to the fact that, for the membership in L, the order of letters in a word does not matter.

The class mon-Q_L^1FO is the class of all languages describable by applying a specific monadic second-order groupoidal quantifier Q_L to an appropriate tuple of formulas without further occurrences of second-order quantifiers. The class mon-Q^1_{Grp}FO is defined analogously using arbitrary monadic second-order groupoidal quantifiers. The class SOM(mon-Q^1_{Grp}) is defined analogously, but allowing groupoidal quantifiers to be used as any other quantifier would (i. e., allowing arbitrary nesting). Analogous notations are used for the quantifiers $Q_L^\star$.

3 Groupoidal Quantifiers Q_L

In this section we consider second-order monadic groupoidal quantifiers under the semantics Q_L. We show that the extension of SOM in terms of all second-order monadic groupoidal quantifiers collapses to its fragment mon-Q^1_{Grp}FO over strings with built-in arithmetic.

The following result on first-order groupoidal quantifiers will be central for our argumentation. Below, QFree denotes the set of quantifier-free formulas in which the predicates $+$ and $\times$ do not appear.

Theorem 1 ([10]). $Q_{\mathrm{Grp}}\mathrm{QFree} = \mathrm{FO}(Q_{\mathrm{Grp}}) = \mathrm{FO}(Q_{\mathrm{Grp}}+,\times) = \mathrm{LOGCFL}$ *over string signatures.*

We shall use the following version of Theorem 1.

Lemma 1. *Let $\tau = \{<, c_1, \ldots, c_s\}$, where $c_1, \ldots, c_s$ are constant symbols. Then on τ-structures*

$$Q_{\mathrm{Grp}}\mathrm{QFree} = \mathrm{FO}(Q_{\mathrm{Grp}}) = \mathrm{FO}(Q_{\mathrm{Grp}}+,\times) \ .$$

Proof. The idea is to encode τ-structures into strings and then apply Theorem 1. In order to encode the information about the identities among $c_1, \ldots, c_s$, we introduce a predicate symbol P_A for each non-empty $A \subseteq \{c_1, \ldots, c_s\}$. To simplify notation, let us assume that $\tau = \{<, c_1, c_2\}$. The general case is analogous.

Suppose that K is a class of τ-structures definable by $\varphi \in \mathrm{FO}(Q_{\mathrm{Grp}}+,\times)$. We shall encode K as a class of strings over signature $\langle P_{\{c_1\}}, P_{\{c_2\}}, P_{\{c_1,c_2\}}, P^*\rangle$. The predicate $P_{\{c_1,c_2\}}$ is used when the interpretations of c_1 and c_2 coincide and P^* is interpreted by all the elements different from c_1 and c_2. Denote by $\mathcal{A}'$ the string encoding a τ-structure $\mathcal{A}$. Let φ^* be acquired from φ by replacing atomic subformulas of the form $c_i = d$ by $P_{\{c_i\}}(d) \vee P_{\{c_1,c_2\}}(d)$ and $c_1 = c_2$ by the formula $\exists x P_{\{c_1,c_2\}}(x)$. It is now obvious how to translate atomic formulas using the predicates $+$, $\times$, and $<$, e.g., $c_i < x$ is replaced by $\exists y((P_{\{c_i\}}(y) \vee P_{\{c_1,c_2\}}(y)) \wedge y < x)$. It is easy to verify that for all $\mathcal{A}$, $\mathcal{A} \models \varphi \Leftrightarrow \mathcal{A}' \models \varphi^*$. By Theorem 1 there is a sentence $\theta \in Q_{\mathrm{Grp}}\mathrm{QFree}$ which is equivalent to φ^* over strings. Let θ^* be acquired from θ by the following substitutions: $P_{\{c_i\}}(d)$ is replaced by $c_i = d \wedge c_1 \neq c_2$, $P_{\{c_1,c_2\}}(d)$ by $c_1 = d \wedge c_1 = c_2$, and finally $P^*(d)$ by $c_1 \neq d \wedge c_2 \neq d$. Now $\theta^* \in Q_{\mathrm{Grp}}\mathrm{QFree}$ and θ^* defines K.

We are now ready for the main result of this section.

Theorem 2. $\text{mon-}Q^1_{\mathrm{Grp}}\mathrm{FO}(+,\times) = \mathrm{FO}(\text{mon-}Q^1_{\mathrm{Grp}}) = \mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Grp}},+,\times)$ *over strings.*

Proof. Fix a signature $\tau = \langle P_{a_1}, \ldots, P_{a_s}\rangle$. Suppose that B is a language defined by some sentence $\varphi \in \mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Grp}},+,\times)[\tau]$. We may assume that $\varphi \in \mathrm{FO}(\text{mon-}Q^1_{\mathrm{Grp}})[\tau]$ since the second-order existential quantifier is included in mon-Q^1_{Grp} and already the extension of FO by the quantifier corresponding

to the (context-free) language *majority* can define the predicates + and × on ordered structures [9].

Denote by $\sigma = \{<, +, \times, c_1, \ldots, c_s\}$ the signature where each c_i is a constant symbol. For a τ-structure $\mathcal{A} = \langle\{0, \ldots, n-1\}, <, P_{a_1}^{\mathcal{A}}, \ldots, P_{a_s}^{\mathcal{A}}\rangle$, let $\mathcal{A}^*$ be the following σ-structure

$$\mathcal{A}^* = \langle\{0, \ldots, 2^n - 1\}, <, +, \times, c_1^{\mathcal{A}^*}, \ldots, c_s^{\mathcal{A}^*}\rangle,$$

where $c_i^{\mathcal{A}^*}$ is the unique integer ($< 2^n$) whose length n binary representation corresponds to $P_i^{\mathcal{A}}$.

We shall first show that there is a sentence $\varphi^* \in \mathrm{FO}(Q_{\mathrm{Grp}}, +, \times)[\sigma]$ such that for all τ-structures $\mathcal{A}$,

$$\mathcal{A} \models \varphi \Leftrightarrow \mathcal{A}^* \models \varphi^* \ .$$

We define φ^* via the following transformation:

$$\begin{aligned}
x_1 = x_2 &\rightsquigarrow x_1 = x_2 \\
x_1 < x_2 &\rightsquigarrow x_1 < x_2 \\
P_{a_i}(z) &\rightsquigarrow \mathrm{BIT}(c_i, z) \\
Y(x) &\rightsquigarrow \mathrm{BIT}(y, x) \\
\psi \wedge \phi &\rightsquigarrow \psi^* \wedge \phi^* \\
\neg\psi &\rightsquigarrow \neg\psi^* \\
\exists x\psi &\rightsquigarrow \exists x(x < n \wedge \psi^*(x)) \\
Q_L X_1, \ldots, X_k[\psi_1, \ldots, \psi_{s-1}] &\rightsquigarrow Q_{L'} x_1, \ldots, x_k[\psi_1^*, \ldots, \psi_{s-1}^*]
\end{aligned}$$

Each assignment f over $\mathcal{A}$ is associated with the assignment f^* over $\mathcal{A}^*$ such that if $f(X) = A \subseteq \{0, \ldots, n-1\}$ then $f^*(x)$ is the unique $a < 2^n$ whose binary representation is given by $s_0 \cdots s_{n-1}$ where $s_j = 1 \iff j \in A$. The predicate BIT, which is FO(+, ×)-definable, allows us to recover the set A from the number a. In other words, $\mathrm{BIT}(a, j)$ holds if bit $n - j - 1$ in the binary representation of a is 1 iff $j \in A$. The language L' is defined by

$$L' = \{w \mid s(w) \in L\},$$

where s is defined as follows: s maps a word w to w if $|w| \neq 2^{km}$ for all $m \in \mathbb{N}^*$. Assuming $|w| = 2^{km}$, the position i of each letter in w is determined by a binary string of length km:

$$P_{bin}(i) = r_1^1 \cdots r_m^1 \cdots r_1^k \cdots r_m^k \ .$$

Now, s takes w to the unique string whose ith letter is identical with the letter in position $r_1^1 r_1^2 \cdots r_1^k r_2^1 r_2^2 \cdots r_2^k \cdots r_m^1 r_m^2 \cdots r_m^k$ in w. In other words, s corrects the asymmetry in the semantics of first-order and second-order quantifiers. It is easy to verify that the language L' is FO(+, ×) reducible to L and thus also definable in $\mathrm{FO}(Q_{\mathrm{Grp}}, +, \times)$. Therefore, the logic $\mathrm{FO}(Q_{\mathrm{Grp}}, +, \times)$ is also closed under the quantifier $Q_{L'}$.

By Lemma 1, there is a sentence

$$\theta = Q_L x_1, \ldots, x_l(\chi_1, \ldots, \chi_w),$$

where each χ_i is quantifier-free and does not contain the predicates $+$ and $\times$, equivalent to φ^*. The idea is now to translate θ to the logic mon-Q_L^1FO$(+,\times)$ by changing first-order variables to second-order variables. We shall construct formulas $\delta_i(\overline{X})$ such that for all τ-structures $\mathcal{A}$

$$\mathcal{A} \models Q_L X_1, \ldots, X_l(\delta_1(\overline{X}), \ldots, \delta_w(\overline{X})) \Leftrightarrow \mathcal{A}^* \models \theta \ .$$

The formula $\delta_i(\overline{X})$ should be satisfied by $A_1, \ldots, A_l \subseteq \{0, \ldots, n-1\}$ iff χ_i is satisfied by the tuple $(a_1, \ldots, a_l) \in \{0, \ldots, 2^n - 1\}^l$ corresponding to $A_1, \ldots, A_l$. Again we need to correct the asymmetry caused by the difference in the semantics of first-order and second-order quantifiers. As in Definition 3, each A_i determines the string $s_0^i \cdots s_{n-1}^i$ with the meaning $s_j^i = 1 \iff j \in A_i$. The tuple $\overline{A}$ is now encoded by the string

$$s_0^1 s_0^2 \cdots s_0^l s_1^1 s_1^2 \cdots s_1^l \cdots s_{n-1}^1 s_{n-1}^2 \cdots s_{n-1}^l \ . \tag{3}$$

Therefore, $\overline{A}$ should satisfy $\delta_i(\overline{X})$ iff the tuple $a_1^*, \ldots, a_l^*$ satisfies χ_i, where the concatenation of the length n binary representations of $a_1^*, \ldots, a_l^*$ correspond to the string in (3). In other words, the binary representation of a_i^* is given by $\mathrm{BIT}(a_i^*, j) = 1$ iff the $(n(i-1)+j)$th bit from the right is 1 in (3) iff $c \in A_r$ for the unique r and c for which $n(i-1)+j = cl+r-1$. Since χ_i is quantifier-free and contains only atomic formulas such as $x_1 < c_2$ or $x_l = x_k$, we can construct the formulas $\delta_i(\overline{X})$ using the fact that the binary representations of $a_1^*, \ldots, a_l^*$ can be recovered from $\overline{A}$ in a first-order way with the help of arithmetic. By the above, it is clear that the sentence $Q_L X_1, \ldots, X_l(\delta_1(\overline{X}), \ldots, \delta_w(\overline{X}))$ now defines B.

4 Groupoidal Quantifiers $Q_L^\star$

In [5] the expressive power of generalized second-order quantifiers was characterized in terms complexity classes given by so-called leaf languages. In particular, for every language B that has a neutral letter, i.e., a letter $\epsilon \in \Gamma$ such that, for all $u, v \in \Gamma^*$, we have $uv \in B \iff u\epsilon v \in B$, the following was shown to hold. Let $\mathcal{N}$ be the class of languages that have a neutral letter.

Theorem 3 ([5]). *For any $B \in \mathcal{N}$, $\mathrm{Leaf}^{\mathrm{P}}(B) = Q_B^\star$FO.*

Above, $\mathrm{Leaf}^{\mathrm{P}}(B)$ denotes the class of languages defined in polynomial-time in terms of non-deterministic Turing machines using the leaf language B and $Q_B^\star$FO denotes the class of all languages describable by applying the quantifier $Q_B^\star$ to an appropriate tuple of first-order formulas (the quantifier $Q_B^\star$ is allowed to bind relation variables of arbitrary arity). Note that in this context we could equivalently use the semantics Q_L instead of $Q_L^\star$. This is due to the fact that the difference between Q_L and $Q_L^\star$ only appears if more than one second-order

variable is quantified and this can be avoided by joining relations into a single relation of higher arity.

Since it is known that there are regular languages B, e.g., the word problem for the group S_5, for which $\mathrm{Leaf}^{\mathrm{P}}(B) = \mathrm{PSPACE}$ [8], we conclude that for such B,

$$Q_B^\star \mathrm{FO} = \mathrm{PSPACE} \ . \tag{4}$$

By a simple padding argument, we see that already first-order logic with a monadic second-order quantifier $Q_B^\star$ is sufficient to define a PSPACE-complete language.

Proposition 1. *Let L be a language and suppose that a language A is definable by a sentence $\varphi \in Q_L^\star \mathrm{FO}$. Let k be the maximum of the arities of the relations quantified in φ. Then the language*

$$A^* = \{w^\frown 0^{|w|^k - |w|} \mid w \in A\}$$

is definable in $\mathrm{FO}(\text{mon-}Q_L^\star, +, \times)$.

Proof. The proof using standard techniques will appear in the journal version of the paper.

Proposition 1 shows that logics $\mathrm{FO}(\text{mon-}Q_L^\star, +, \times)$ can be quite powerful. In this section we show that a result analogous to Theorem 2 also holds with respect to the semantics $Q_L^\star$. We also show that in the most general case, i.e., when the logic in question is the extension of SOM by all second-order monadic groupoidal quantifiers, both semantics turn out to be equal in expressive power.

Theorem 4. $\text{mon-}Q_{\mathrm{Grp}}^\star \mathrm{FO} = \mathrm{SOM}(\text{mon-}Q_{\mathrm{Grp}}^\star, +, \times) = \mathrm{SOM}(\text{mon-}Q_{\mathrm{Grp}}^1, +, \times)$ *over strings.*

Proof. Let us first note that by an analogous argument as in the proof of Theorem 2 any sentence $\varphi \in \mathrm{SOM}(\text{mon-}Q_{\mathrm{Grp}}^\star, +, \times)$ can be first translated into $\mathrm{FO}(Q_{\mathrm{Grp}}, +, \times)$ and then to the logic $\text{mon-}Q_{\mathrm{Grp}}^1 \mathrm{FO}(+, \times)$. In fact, the first translation can be even simplified since the quantifier $Q_{L'}$ is not needed. Therefore, it suffices to show the converse inclusion.

Let A be defined by a sentence $\varphi \in \mathrm{SOM}(\text{mon-}Q_{\mathrm{Grp}}^1, +, \times)$. We use the same argument as in the proof of Theorem 2. We only need to modify the last part of the proof and define the translation from a sentence θ of the form

$$Q_L x_1, \ldots, x_k(\chi_1, \ldots, \chi_v),$$

where each χ_i is quantifier-free and does not contain the predicates $+$ and $\times$. We do this in the following way. Denote by $X = Y$ the formula $\forall z(X(z) \leftrightarrow Y(z))$, and by $X < Y$ the first-order formula defining the ordering of subsets when treated as length n binary strings. The transformation is now defined by

$$\begin{gathered} x = y \rightsquigarrow X = Y \\ x < y \rightsquigarrow X < Y \\ \psi \wedge \phi \rightsquigarrow \psi' \wedge \phi' \\ \neg\psi \rightsquigarrow \neg\psi' \\ Q_L x_1, \ldots, x_v[\psi_1, \ldots, \psi_v] \rightsquigarrow Q_L^\star X_1, \ldots, X_v[\psi_1', \ldots, \psi_v'] \end{gathered}$$

The use of $Q_L^\star$ allows us to define the translation simply by changing first-order variables to second-order variables. It is easy to verify that θ' now defines the language A.

By combining Theorems 2 and 4, we get

Corollary 1. $\mathrm{SOM}(\text{mon-}Q^1_{\mathrm{Grp}}, +, \times) = \text{mon-}Q^1_{\mathrm{Grp}}\mathrm{FO}(+, \times) = \text{mon-}Q^\star_{\mathrm{Grp}}\mathrm{FO} = \mathrm{SOM}(\text{mon-}Q^\star_{\mathrm{Grp}}, +, \times)$.

5 Connection to Leaf Automata

A *finite leaf automaton* is a tuple $M = (Q, \Sigma, \delta, s, \Gamma, \beta)$ where Q is the finite set of *states*, Σ is an alphabet, the *input alphabet*, $\delta\colon Q \times \Sigma \to Q^+$ is the *transition function*, $s \in Q$ is the *initial state*, Γ is an alphabet, the *leaf alphabet*, and $\beta\colon Q \to \Gamma$ is a function that associates a state q with its *value* $\beta(q)$. The sequence $\delta(q, a)$, for $q \in Q$ and $a \in \Sigma$, contains all possible successor states of M when reading letter a while in state q, and the order of letters in that sequence defines a *total order on these successor states*. This definition allows the same state to appear more than once as a successor in $\delta(q, a)$.

Let M be as above. The computation tree $T_M(w)$ of M on input w is a labeled directed rooted tree defined as follows:

- The root of $T_M(w)$ is labeled (s, w).
- Let v be a node in $T_M(w)$ labeled by (q, x), where $x \neq \epsilon$ (the empty word), $x = ay$ for $a \in \Sigma$, $y \in \Sigma^*$. Let $\delta(q, a) = q_1 q_2 \cdots q_k$. Then v has k children in $T_M(w)$, and these are labeled by $(q_1, y), (q_2, y), \ldots, (q_k, y)$ in this order.

If we look at the tree $T_M(w)$ and attach the symbol $\beta(q)$ to a leaf in this tree with label (q, ε), then $\mathrm{leafstring}^M(w)$ is defined to be the string of symbols attached to the leaves, read from left to right in the order induced by δ.

Definition 4. *For $A \subseteq \Gamma^*$, the class $\mathrm{Leaf}^{\mathrm{FA}}(A)$ consists of all languages $B \subseteq \Sigma^*$, for which there is a leaf automaton M as just defined, with input alphabet Σ and leaf alphabet Γ such that for all $w \in \Sigma^*$, $w \in B$ iff $\mathrm{leafstring}^M(w) \in A$. If C is a class of languages then $\mathrm{Leaf}^{\mathrm{FA}}(C) = \cup_{A \in C}\mathrm{Leaf}^{\mathrm{FA}}(A)$.*

In [13] the acceptance power of leaf automata with different kinds of leaf languages was examined. It was shown that, with respect to resource-bounded leaf language classes, there is not much difference, e.g., between automata and Turing machines. On the other hand, if the leaf language class is a formal language class then the differences can be huge. In particular it was shown in [13] that $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{REG}) = \mathrm{REG}$ while it is known that $\mathrm{Leaf}^{\mathrm{P}}(\mathrm{REG}) = \mathrm{PSPACE}$. In [13] the power of $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$ was left as an open question. The only upper and lower bounds known are $\mathrm{CFL} \subsetneq \mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL}) \subseteq \mathrm{DSPACE}(n^2) \cap \mathrm{DTIME}\big(2^{O(n)}\big)$.

In [6] the class $\mathrm{Leaf}^{\mathrm{FA}}(L)$ was logically characterized assuming that the language L has a neutral letter.

Theorem 5 ([6]). *For any $L \in \mathcal{N}$, $\mathrm{Leaf}^{\mathrm{FA}}(L) = \text{mon-}Q^1_L\mathrm{FO}$.*

We would like to use either Theorem 2 or Theorem 4 to show that the class $\mathrm{Leaf}^{\mathrm{FA}}(\mathrm{CFL})$ contains PSPACE-complete languages. Unfortunately, Theorem 2 does not apply because it assumes built-in arithmetic which is not allowed in Theorem 5. On the other hand, due to the change in the interpretation of quantifiers in Theorem 4, it is not clear that Theorem 5 holds in this case.

Recall that Greibach's hardest context-free language H is a so-called nondeterministic version of the Dyck language D_2, the language of all syntactically correct sequences consisting of letters for two types of parentheses. It is known that every $L \in \mathrm{CFL}$ reduces to H under some homomorphism [7]. It was shown in [10] that in Theorem 1 the logic $Q_{\mathrm{Grp}}\mathrm{QFree}$ can be even replaced by $Q_{\mathrm{pad}(H)}\mathrm{QFree}$, where $\mathrm{pad}(H)$ is H extended by a neutral symbol. Therefore, we can similarly replace the logics mon-$Q^1_{\mathrm{Grp}}\mathrm{FO}$ and mon-$Q^\star_{\mathrm{Grp}}\mathrm{FO}$, in Theorems 2 and 4, by mon-$Q^1_{\mathrm{pad}(H)}\mathrm{FO}(+,\times)$ and mon-$Q^\star_{\mathrm{pad}(H)}\mathrm{FO}$, respectively.

We call a language symmetric if it is closed under permuting the letters of words. Note that if $\mathrm{pad}(H)$ happened to be symmetric, then we could use the proof of Theorem 4 to show that mon-$Q^1_{\mathrm{pad}(H)}\mathrm{FO}$ = mon-$Q^1_{\mathrm{pad}(H)}\mathrm{FO}(+,\times)$. However, this assumption turns out not to be true, since a symmetric context-free language cannot be complete for all of CFL under homomorphims. It can be even shown that symmetric context-free languages are contained in TC^0.

6 Conclusion

In this paper we have studied several monadic second-order logics with groupoidal quantifiers. Our collapse results partially address an open question in [6]. However, the main open question of that paper remains: What is the power of finite leaf-automata with context-free leaf languages? If one could prove equality between the two variants of semantics for second-order quantifiers, i.e.,

$$\text{mon-}Q^1_{\mathrm{Grp}}\mathrm{FO} = \text{mon-}Q^\star_{\mathrm{Grp}}\mathrm{FO},$$

then it follows immediately from our results that such simple automata can even accept PSPACE-complete problems.

References

1. Barrington, D.A.M., Immerman, N., Straubing, H.: On uniformity within NC^1. Journal of Computer and System Sciences 41, 274–306 (1990)
2. Bédard, F., Lemieux, F., McKenzie, P.: Extensions to Barrington's M-program model. Theoretical Computer Science 107, 31–61 (1993)
3. Büchi, J.R.: On a decision method in restricted second-order arithmetic. In: Proceedings Logic, Methodology and Philosophy of Sciences 1960, Stanford University Press, Stanford (1962)
4. Büchi, J.R., Elgot, C.C.: Decision problems of weak second order arithmetics and finite automata, Part I. Notices of the American Mathematical Society 5, 834 (1958)
5. Burtschick, H.J., Vollmer, H.: Lindström quantifiers and leaf language definability. International Journal of Foundations of Computer Science 9, 277–294 (1998)

6. Galota, M., Vollmer, H.: A generalization of the Büchi-Elgot-Trakhtenbrot theorem. In: Fribourg, L. (ed.) CSL 2001 and EACSL 2001. LNCS, vol. 2142, pp. 355–368. Springer, Heidelberg (2001)
7. Greibach, S.: The hardest context-free language. SIAM Journal on Computing 2, 304–310 (1973)
8. Hertrampf, U., Lautemann, C., Schwentick, T., Vollmer, H., Wagner, K.W.: On the power of polynomial time bit-reductions. In: Proceedings 8th Structure in Complexity Theory, pp. 200–207 (1993)
9. Kontinen, J., Niemistö, H.: Extensions of MSO and the monadic counting hierarchy (2006), http://www.helsinki.fi/~jkontine/
10. Lautemann, C., McKenzie, P., Schwentick, T., Vollmer, H.: The descriptive complexity approach to LOGCFL. Journal of Computer and Systems Sciences 62, 629–652 (2001)
11. Lindström, P.: First order predicate logic with generalized quantifiers. Theoria 32, 186–195 (1966)
12. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge (1971)
13. Peichl, T., Vollmer, H.: Finite automata with generalized acceptance criteria. Discrete Mathematics and Theoretical Computer Science 4, 179–192 (2001)
14. Straubing, H.: Finite Automata, Formal Logic, and Circuit Complexity. Birkhäuser, Boston (1994)
15. Trakhtenbrot, B.A.: Finite automata and logic of monadic predicates (in Russian). Doklady Akademii Nauk SSSR 140, 326–329 (1961)

Inference Processes for Quantified Predicate Knowledge

J.B. Paris and S.R. Rad*

School of Mathematics
University of Manchester
Manchester M13 9PL, UK
jeff.paris@manchester.ac.uk,
soroush.rafiee-rad@postgrad.manchester.ac.uk

Abstract. We describe a method for extending an inference process for propositional probability logic to predicate probability logic in the case where the language in purely unary and show that the method is well defined for the Minimum Distance and CM^{∞} inference processes.

Keywords: Uncertain reasoning, probability logic, inference processes, Renyi Entropies.

1 Motivation

In this paper, which extends [3], we consider the following problem

> *Given that we know only that a probability function w on a predicate language L satisfies the finite set K of constraints*
>
> $$\sum_{i=1}^{q} c_{ij} w(\theta_i) \leq b_j, \quad j = 1, \ldots, m$$
>
> *where the θ_i are sentences of L and the $c_{ij}, b_j \in \mathbb{R}$ what value should be given to $w(\theta)$, for a sentence θ of L?*

in the limited case where L has just finitely many unary predicate symbols $P_1, \ldots, P_n$ and countably many constant symbols $a_1, a_2, \ldots$ (which are intended to exhaust the universe) but no function symbols nor equality.

The relevance of this question for AI is that we imagine an agent whose knowledge consists of just K wishing to nevertheless assign probabilities to other sentences of the language. Indeed if, as seems quite reasonable, we require these assigned values to be consistent as a whole with K and w being a probability function then the question amounts to asking how one should best pick a probability function w on L, that is a function w on the set SL sentences of the language L satisfying that for all $\theta, \phi, \exists x \psi(x) \in SL$

* Supported by a MATHLOGAP Research Studentship - MEST-CT-2004-504029.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 249–259, 2008.

(P1) If $\vDash \theta$ then $w(\theta) = 1$

(P2) If $\vDash \neg(\theta \wedge \phi)$ then $w(\theta \vee \phi) = w(\theta) + w(\phi)$

(P3) $w(\exists x \psi(x)) = \lim_{r\to\infty} w(\bigvee_{j=1}^{r} \psi(a_j))$,

given that w must also satisfy K.

A number of possible answers to such a question have been proposed both for propositional and predicate languages, for example [1], [2], [3], [9], [10], [11], [17], [19], [20], [21], [22], based on various underlying assumptions about the form and origin of the knowledge and the probability function w, see [3] for a discussion. As in that paper we shall assume that w is a subjective probability function corresponding to the beliefs of some agent and that the assigning agent intends to act rationally or logically (though we shall not make any attempt to define these terms here, instead simply leaving it to the reader to decide to what extent our proposals fulfill that intention).

The method we shall describe in this note extends a well developed approach (see [17, Chapter 6]) for the analogous problem in the *propositional* case to the limited *predicate* situation when the language L is *purely unary*. This same path has already been trodden in [3] in a special case (viz. the Maximum Entropy Inference Process). The main novelty in this paper is in giving a general result which applies to a wide range of inference processes.

In the next section we explain, in the specific case of the Minimum Distance Inference Process (see [17, p76]), this method for picking a probability function satisfying K and the key limit result, which we prove in the subsequent section. In the final section we consider how this specific case generalizes.

2 The Method

The idea, as explained in [3], for assigning a probability to a sentence $\theta(a_1, a_2, \ldots, a_m)$ from SL is that this should be the limit as r tends to ∞ of the probability that one would assign to it being true in a finite structure with universe $\{ a_i \,|\, i \leq r \}$. In other words we wish to approximate our beliefs in what holds in a universe with denumerably many individuals $a_1, a_2, \ldots$ with our beliefs of what holds in its large finite substructures.

In more detail let L^k be the sublanguage of L with the same unary predicate symbols $P_1, \ldots, P_n$ but only the constant symbols $a_1, ..., a_k$ and let $Q_1, ..., Q_J$, $J = 2^n$ enumerate all formulas of the form

$$P_1^{\epsilon_1}(x) \wedge P_2^{\epsilon_2}(x) \wedge ... \wedge P_n^{\epsilon_n}(x) \tag{1}$$

where the $\epsilon_1, \epsilon_2, \ldots, \epsilon_n \in \{0, 1\}$ and $P^1 = P, P^0 = \neg P$. Let $\mathfrak{L}^r$ be the propositional language with the propositional variables $P_j(a_i)$, $i = 1, ..., r$ $j = 1, ..., n$. For $k < r$ define $()^{(r)} : SL^k \to S\mathfrak{L}^r$ inductively as follows:

$$P_j(a_i)^{(r)} = P_j(a_i),$$

$$(\neg\phi)^{(r)} = \neg\phi^{(r)},$$

$$(\phi \vee \theta)^{(r)} = \phi^{(r)} \vee \theta^{(r)},$$
$$(\phi \wedge \theta)^{(r)} = \phi^{(r)} \wedge \theta^{(r)},$$
$$(\exists x \psi(x))^{(r)} = \bigvee_{i=1}^{r} \psi(a_i)^{(r)}.$$

Let $K^{(r)}$ be the result of replacing every sentence θ_i in K by $\theta_i^{(r)}$. As indicated above we now wish to make a 'rational' choice $N(K^{(r)})$ of a probability function satisfying $K^{(r)}$ and thence define our 'rational' probability function w satisfying K by

$$w(\theta) = \lim_{r \to \infty} N(K^{(r)})(\theta^{(r)}).$$

Apart from the question of whether this limit even exists (which we will confront in the next section), and even then satisfies K, we need to justify our assignment of probabilities for these finite substructures satisfying the $K^{(r)}$. Fortunately however we are now essentially working in the propositional calculus and a number of such assignment processes, in this context called *Inference Processes, N*, for picking a probability function $N(K)$ (or set of functions, see [22]) satisfying a probabilistic propositional knowledge base K have been studied, and to some extent justified, see for example [17, Chapter 6].

Currently the generally most accepted inference process according to this criterion of rationality is the so called Maximum Entropy Inference Process, and indeed the programme we are advocating in this paper has already been carried out for maximum entropy in [3]. What we plan to do here is to retread this path using the Minimum Distance Inference Process, MD, and then point out how the necessary steps actually hold for a wide range of other inference processes.

Before proceeding with the proof we need to recall the definition of MD. Given a finite proposition language, say with propositional variables $p_1, p_2, \ldots, p_k$, a probability function v on the sentences of this language is determined by (and will be identified with) the vector

$$\langle v(\beta_1), v(\beta_2), \ldots, v(\beta_{2^k}) \rangle \in \mathbb{D}_{2^k} = \{ \langle x_1, x_2, \ldots, x_{2^k} \rangle \mid x_i \geq 0, \ \sum_i x_i = 1 \}$$

where the β_j run through the *atoms*

$$p_1^{\epsilon_1} \wedge p_2^{\epsilon_2} \wedge \ldots \wedge p_k^{\epsilon_k}.$$

Given a non-empty closed and convex subset C of $\mathbb{D}_{2^k}$ we define $MD(C)$ to be that (unique) probability function $\langle x_1, x_2, \ldots, x_{2^k} \rangle \in C$ for which $\sum_i x_i^2$ is minimal. Equivalently that point in C closest in Euclidean distance to the probability function $\langle 2^{-k}, 2^{-k}, \ldots, 2^{-k} \rangle$, which we can think of as representing complete ignorance.

3 The Existence of the Limit

The following results appear in [3] (also there as Lemma 1, Lemma 2 and Theorem 3) and we shall use them in what follows.

Lemma 1. *If $\theta, \phi \in SL^k$ and $k \leq r$ and $\theta \equiv \phi$ then $\theta^{(r)} \equiv \phi^{(r)}$.*

Let α_i for $i = 1, ..., J^k$ enumerate the exhaustive and exclusive set of sentences of the form

$$\bigwedge_{i=1}^{k} Q_{m_i}(a_i)$$

where the Q_i are as in (1).

Lemma 2. *Any sentence $\theta \in SL^k$ is equivalent to a disjunction of consistent sentences $\phi_{i,\boldsymbol{\epsilon}}$ of the form*

$$\alpha_i \wedge \bigwedge_{j=1}^{J} (\exists x Q_j(x))^{\epsilon_j}$$

where the $\epsilon_j \in \{0, 1\}$ and $\vDash \neg(\phi_{i,\boldsymbol{\epsilon}} \wedge \phi_{j,\boldsymbol{\delta}})$ whenever $\langle i, \boldsymbol{\epsilon}\rangle \neq \langle j, \boldsymbol{\delta}\rangle$.

Theorem 3. *$K^{(r)}$ is a satisfiable subset of $S\mathfrak{L}^r$ for large r.*

Theorem 4. *For $\theta \in SL$:*

$$w(\theta) = \lim_{r\to\infty} MD(K^{(r)})(\theta^{(r)})$$

exists and is a probability function on L satisfying K.

Proof. Assume throughout that r is large so that Theorem 3 applies. By Lemma 2 every sentence $\theta(a_1, ..., a_k) \in SL$ is equivalent to a disjunction of consistent sentences of the form

$$\phi_{i,\boldsymbol{\epsilon}} = \alpha_i \wedge \bigwedge_{j=1}^{J} (\exists x Q_j(x))^{\epsilon_j}.$$

If $\alpha_i = \bigwedge_{j=1}^{k} Q_{m_j}(a_j)$ then let

$$A_i = \{\, m_j \,|\, j = 1, ..., k \,\},\ P_{\boldsymbol{\epsilon}} = \{\, j \,|\, \epsilon_j = 1 \,\},\ P_{i,\boldsymbol{\epsilon}} = \{\, j \,|\, j \in P_{\boldsymbol{\epsilon}} \text{ and } j \notin A_i \,\}$$

so

$$\phi^{(r)}_{i,\boldsymbol{\epsilon}} = \alpha_i \wedge \bigwedge_{j=1}^{J} \left(\bigvee_{i=1}^{r} Q_j(a_i) \right)^{\epsilon_j}$$

will be equivalent to

$$\bigvee_{\substack{m_j \in P_{\boldsymbol{\epsilon}} \text{ for } j=k+1,\ldots,r \\ P_{i,\boldsymbol{\epsilon}} \subseteq \{\, m_j \,|\, k+1 \leq j \leq r\}}} \left(\alpha_i \wedge \bigwedge_{j=k+1}^{r} Q_{m_j}(a_j) \right). \tag{2}$$

If we set $p_{\boldsymbol{\epsilon}} = |P_{\boldsymbol{\epsilon}}|$ and $p_{i,\boldsymbol{\epsilon}} = |P_{i,\boldsymbol{\epsilon}}|$ then the number of disjuncts (i.e. atoms of $\mathfrak{L}^r$) in (2) will be

$$\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}} (-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}} - j)^{r-k}.$$

Let $w^{(r)} = MD(K^{(r)})$. Since MD satisfies renaming (see [17, p95]), for every atom ζ in the disjunction in (2)

$$w^{(r)}(\zeta) = \frac{w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}})}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}} - j)^{r-k}}.$$

Hence if the ζ_j enumerate the atoms of $\mathfrak{L}^r$,

$$\sum_{j=1}^{J^r}(w^{(r)}(\zeta_j))^2 = \sum_{i,\boldsymbol{\epsilon}} \left(\frac{w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}})}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}} - j)^{r-k}} \right)^2$$

$$= \sum_{i,\boldsymbol{\epsilon}} \frac{(w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}}))^2}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}} - j)^{r-k}}.$$

From this it follows that w satisfying $K^{(r)}$ is equivalent to some set of linear inequalities

$$\sum_{i,\boldsymbol{\epsilon}} c_{i,\boldsymbol{\epsilon},j} w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}}) \leq b_j, \quad j = 1, \ldots, m \tag{3}$$

where the $c_{i,\boldsymbol{\epsilon},j}$ and b_j *do not depend on* r. Hence the vector of values $w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}})$ (as $i, \boldsymbol{\epsilon}$ vary) is that vector $x_{i,\boldsymbol{\epsilon}} \geq 0$ satisfying (3) for which

$$\sum_{i,\boldsymbol{\epsilon}} \frac{x^2_{i,\boldsymbol{\epsilon}}}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}} - j)^{r-k}} \tag{4}$$

is minimal.

Let $c_1 < c_2 < \ldots < c_k$ be the distinct values for $p_{\boldsymbol{\epsilon}}$ which occur here and define the sets

$$T_0 = \{ \boldsymbol{x} \mid \sum_{i,\boldsymbol{\epsilon}} c_{i,\boldsymbol{\epsilon},j} w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}}) \leq b_j, \quad j = 1, \ldots, m\}$$

and

$$T_{j+1} = \{ \boldsymbol{x} \in T_j \mid \sum_{i, p_{\boldsymbol{\epsilon}} = c_{j+1}} x^2_{i,\boldsymbol{\epsilon}} \text{ is minimal} \}$$

for $0 \leq j < k$. Since these T_j are closed and convex any two points in T_j agree on those coordinates $\langle i, \boldsymbol{\epsilon} \rangle$ with $p_{\boldsymbol{\epsilon}} \leq c_j$. Hence T_k consists of a single point, $\boldsymbol{X}$ say. Notice that this point does not depend on r.

Since $\boldsymbol{X} \in T_0$ by (4)

$$\sum_{i,\boldsymbol{\epsilon}} \frac{(w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}}))^2}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}} - j)^{r-k}} \leq \sum_{i,\boldsymbol{\epsilon}} \frac{X^2_{i,\boldsymbol{\epsilon}}}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}} - j)^{r-k}}. \tag{5}$$

The $w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}})$ have a convergent subsequence (as $r \to \infty$), which for notational convenience we shall assume is the whole sequence, say

$$Y_{i,\boldsymbol{\epsilon}} = \lim_{r\to\infty} w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}}).$$

We shall show that $Y_{i,\boldsymbol{\epsilon}} = X_{i,\boldsymbol{\epsilon}}$ in turn for each of the cases $p_{\boldsymbol{\epsilon}} = c_1, c_2, \ldots, c_k$.

Firstly for the case of c_1, multiplying (5) by c_1^{r-k} and taking the limit as $r \to \infty$ gives

$$\sum_{i,p_{\boldsymbol{\epsilon}}=c_1} Y^2_{i,\boldsymbol{\epsilon}} \leq \sum_{i,p_{\boldsymbol{\epsilon}}=c_1} X^2_{i,\boldsymbol{\epsilon}}$$

and hence for such $\boldsymbol{\epsilon}$, $Y_{i,\boldsymbol{\epsilon}} = X_{i,\boldsymbol{\epsilon}}$ by definition of T_1.

To handle the case $p_{\boldsymbol{\epsilon}} = c_2$ and beyond we will need the following lemma.

Lemma 5. *Let $B \subseteq R^m$ be a convex polyhedron with corners $\boldsymbol{a_1}, \boldsymbol{a_2}, ..., \boldsymbol{a_q}$. Let $\boldsymbol{c} \in B$ and let $f : R^m \to R^n$ be the projection function given by*

$$f\langle x_1, x_2, ..., x_m\rangle = \langle x_1, x_2, ..., x_n\rangle$$

Suppose that $\boldsymbol{y_j} \in R^n$ for $j \in \mathbb{N}$ are such that $f^{-1}(\boldsymbol{y_j}) \cap B \neq \emptyset$ for all j and

$$lim_{j\to\infty}\boldsymbol{y_j} = f(\boldsymbol{c}).$$

Then there is a subsequence $\boldsymbol{z_j} \in B$ converging to $\boldsymbol{c}$ such that the $f(\boldsymbol{z_j})$ form a subsequence of the $\boldsymbol{y_j}$.

Proof. Any point in B can be written as a linear combination

$$\boldsymbol{c} + \sum_{i=1}^{q} \lambda_i \boldsymbol{e_i},$$

where $\boldsymbol{e_i} = \boldsymbol{a_i} - \boldsymbol{c}$ and the $\lambda_i \geq 0$ with sum ≤ 1, so any $\boldsymbol{x} \in f(B)$ can be written as

$$f(\boldsymbol{c}) + \sum_{i=1}^{q} \lambda_i f(\boldsymbol{e}_i)$$

with $\lambda_i > 0$ with sum at most 1, where we drop any terms with $\lambda_i = 0$ (but to avoid messy notation assume there are none, and that this is true also for the $\boldsymbol{y_j}$, otherwise pick a suitable subsequence with the same zero terms throughout). Now for each $\boldsymbol{y_j}$ pick one such presentation:

$$\boldsymbol{y}_j = f(\boldsymbol{c}) + \sum_{i=1}^{q} \lambda_{ij} f(\boldsymbol{e}_i)$$

and set

$$\boldsymbol{z}_j = \boldsymbol{c} + \sum_{i=1}^{q} \lambda_{ij} \boldsymbol{e}_i.$$

It is obvious that the $f(\boldsymbol{z_j})$ form a subsequence of the $\boldsymbol{y_j}$. To show that $\lim_{j\to\infty}$ $\boldsymbol{z_j} = \boldsymbol{c}$ it is enough to show that $\lim_{j\to\infty} \sum_{i=1}^{q} \lambda_{ij}\boldsymbol{e}_i = \boldsymbol{0}$. To this end we will show that $\lim_{j\to\infty} \lambda_{ij} = 0$. We know that $\lim_{j\to\infty} \boldsymbol{y_j} = f(c)$ and so we have

$$\lim_{j\to\infty} \sum_{i=1}^{q} \lambda_{ij} f(\boldsymbol{e}_i) = \mathbf{0}.$$

Let

$$\boldsymbol{t_j} = \sum_{i=1}^{q} \lambda_{ij} f(\boldsymbol{e}_i),$$

so

$$\lim_{j\to\infty} \boldsymbol{t_j} = \mathbf{0}$$

and $\boldsymbol{t_j}$ is in the convex polyhedron with corners $f(\boldsymbol{e_i})$. For each $\boldsymbol{t_j}$ pick a smallest set $f(\boldsymbol{e_{i_1}}), ..., f(\boldsymbol{e_{i_h}})$ such that:

$$\boldsymbol{t_j} = \sum_{k=1}^{h} \lambda_{i_k j} f(\boldsymbol{e_{i_k}}) \tag{6}$$

with $\lambda_{i_k j} \geq 0$ and $\sum_{i_k} \lambda_{i_k j} \leq 1$. By taking a subsequence if necessary we can assume that the $\boldsymbol{t_j}$ all have the same smallest set and that $\lambda_{i_k j} \to \lambda_{i_k}$ as $j \to \infty$. For simplicity of notation we assume that these smallest sets are all the $\boldsymbol{e_i}$, so (6) will become

$$\boldsymbol{t_j} = \sum_{i=1}^{q} \lambda_{ij} f(\boldsymbol{e_i}) \tag{7}$$

and

$$\mathbf{0} = \sum_{i=1}^{q} \lambda_i f(\boldsymbol{e_i}). \tag{8}$$

Now if all the $\lambda_i = 0$ we have the required result, otherwise suppose some of the $\lambda_i > 0$. Then from (7) and (8) we will have:

$$\boldsymbol{t_j} = \sum_{i=1}^{q} (\lambda_{ij} - \nu\lambda_i) f(\boldsymbol{e_i}) \tag{9}$$

Now as we increase ν from 0 one of the coefficients in (9) will become zero while others are still non-negative and this contradicts the choice of smallest set, and so is a contradiction. Hence we must have that all the $\lambda_i = 0$, as required. ■

To continue the proof of Theorem 4 suppose that $Y_{i,\boldsymbol{\epsilon}} \neq X_{i,\boldsymbol{\epsilon}}$ for some $p_{\boldsymbol{\epsilon}} = c_2$. We have already shown that we do have equality when $p_{\boldsymbol{\epsilon}} = c_1$ so by Lemma 5 there is a sequence of vectors $z^{(r)}_{i,\boldsymbol{\epsilon}} \in T_0$ such that for each $i, \boldsymbol{\epsilon}$

$$\lim_{r\to\infty} z^{(r)}_{i,\boldsymbol{\epsilon}} = X_{i,\boldsymbol{\epsilon}}$$

and the $z^{(r)}_{i,\boldsymbol{\epsilon}}$ form a subsequence of the $w^{(r)}(\phi^{(r)}_{i,\boldsymbol{\epsilon}})$ for $p_{\boldsymbol{\epsilon}} = c_1$. For simplicity of notation we will again assume that this subsequent is the whole sequence.

By definition of $w^{(r)}$,

$$\sum_{i,\boldsymbol{\epsilon}} \frac{(w^{(r)}(\phi_{i,\boldsymbol{\epsilon}}^{(r)}))^2}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}}-j)^{r-k}} \leq \sum_{i,\boldsymbol{\epsilon}} \frac{z_{i,\boldsymbol{\epsilon}}^2}{\sum_{j=0}^{p_{i,\boldsymbol{\epsilon}}}(-1)^j \binom{p_{i,\boldsymbol{\epsilon}}}{j} (p_{\boldsymbol{\epsilon}}-j)^{r-k}}.$$

The parts of these sums for $p_{\boldsymbol{\epsilon}} = c_1$ are equal, so can be cancelled out. Multiplying both sides of what remains by c_2^{r-k} and taking the limit as $r \to \infty$ we obtain, just as in the case of c_1 that

$$\sum_{i,p_{\boldsymbol{\epsilon}}=c_2} Y_{i,\boldsymbol{\epsilon}} \leq \sum_{i,p_{\boldsymbol{\epsilon}}=c_2} X_{i,\boldsymbol{\epsilon}}.$$

But since, as we have already shown, the vector $Y_{i,\boldsymbol{\epsilon}}$ is in T_1 this inequality must also go the other way since the vector $X_{i,\boldsymbol{\epsilon}}$ is in T_2. We conclude as required that $Y_{i,\boldsymbol{\epsilon}} = X_{i,\boldsymbol{\epsilon}}$ whenever $p_{\boldsymbol{\epsilon}} = c_2$. Similar arguments give the same result for $c_3, c_4, \ldots, c_k$, as required.

Finally, since any linear identity satisfied by all the $w^{(r)}$ eventually will be satisfied by their limit it is clear that this limit is a probability function satisfying K. ■

4 Some Generalizations

So far we have proved Theorem 4 for a specific inference process, Minimum Distance, but in fact analogous proofs give the the result too for the Maximum Entropy Inference Process (already proved in [3]), the Limiting Centre of Mass Inference Process (see [17, p73-74]) and the spectrum of other inference processes based on generalized Renyi Entropies. [For further results along these lines see [23].]

In our original question we imagined an agent wishing to assign probabilities to all sentences on the basis of qualified knowledge K. A special case of this is when K simply amounts to the assertion that some consistent, finite, set of axioms $\mathcal{T}$ hold categorically, i.e.

$$K = \{\, w(\phi) = 1 \,|\, \phi \in \mathcal{T} \,\}.$$

In this case our question might be reformulated as

> *Given a finite (consistent) set $\mathcal{T}$ of first order axioms what should we take as the default or most normal model of $\mathcal{T}$? More precisely, if we know only that the structure M with universe $\{\, a_i \,|\, i \in \mathbb{N} \,\}$ is a model of $\mathcal{T}$ what probability should we give to a sentence $\theta(a_1, a_2, \ldots, a_n)$ being true in M?*

There are various approaches one might take to this question depending on the interpretation of 'most normal'. For example within a model theory context one might consider a *prime model*, where such exists, to be the 'most normal' in the sense of being the smallest and the canonical example (see for example [5, p96],

[8, p336]). On the other hand one might feel that if possible the default model should be existentially closed (see [24] for a precise definition) in the sense that any quantifier free formula which could be satisfied in a superstructure model of $\mathcal{T}$ was already satisfied in the default model. Alternatively we might consider arguing via the distribution of models, see for example [1], [2], [9], [10], [11], in order to make the default the 'average' model.

Furthermore, at first sight it would appear that there was already a rather well studied approach to this problem via Inductive Logic. In that subject, see for example [4], [7], [13], [16], this same problem with $\mathcal{T} = \emptyset$ is quite central. So it might seem that a solution to our problem here could be had by simply taking a rationally justified probability function w championed within Inductive Logic for the case of a completely empty knowledge base and then conditioning w on $\bigwedge \mathcal{T}$. The first problem with that approach however is that there is currently no clearly favored rational solution to the Inductive Logic problem. But more seriously, those solutions w which have been proposed generally give non-tautologous universal sentences probability 0, see for example [12], [14], [15], [16, p22-23], [17, p196-197], and once $w(\bigwedge \mathcal{T}) = 0$ such conditioning will not be possible.[1,2]

However if we assume that the sentences of $\mathcal{T}$ come from the *purely unary language* of the preceding sections then the method described in this paper, based on any of the above inference processes, indeed in this simple case of *categorical* knowledge, $K = \{ w(\phi) = 1 \mid \phi \in \mathcal{T} \}$, based on *any* inference process just satisfying the Renaming Principle, can be applied, and in fact always yield the same answer. Namely that, in the notation of the proof of Theorem 4, if $\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^s$ are all the vectors $\boldsymbol{\epsilon}$ for which $\bigwedge_{j=1}^{J} (\exists x Q_j(x))^{\epsilon_j}$ is consistent with $\mathcal{T}$ and amongst which $p_{\boldsymbol{\epsilon}}$ takes its largest value then $w(\theta(a_1, \ldots, a_k)) = H/K$ where

$$K = |\{ \phi_{i,\boldsymbol{\epsilon}^r} \mid \phi_{i,\boldsymbol{\epsilon}^r} \text{ is consistent with } \bigwedge \mathcal{T},\ 1 \leq i \leq J^k,\ 1 \leq r \leq s \}|,$$

$$H = |\{ \phi_{i,\boldsymbol{\epsilon}^r} \mid \phi_{i,\boldsymbol{\epsilon}^r} \text{ is consistent with } \theta(a_1, \ldots, a_k) \wedge \bigwedge \mathcal{T},\ 1 \leq i \leq J^k,\ 1 \leq r \leq s \}|.$$

In particular then w gives probability 1 to

$$\bigvee_{i=1}^{s} \bigwedge_{j=1}^{J} (\exists x Q_j(x))^{\epsilon_j^i},$$

[1] It is true that proposals have been made for solutions to the Inductive Logic problem which give some non-tautologous universal sentences non-zero probability, see for example [6], [12], [14], [15], [18]. However they seem (to us) too ad hoc to be seriously considered 'logical'.

[2] This apparent discontinuity between the cases when $\mathcal{T} =/\neq \emptyset$ is intriguing – the method we shall apply in this paper still works when $\mathcal{T} = \emptyset$ but gives an unsatisfactory solution to the inductive logic problem, unsatisfactory in that it corresponds to the so called completely independent solution which entertains no induction i.e. learning by example, see for example [17, p172].

(and probability $1/s$ to each of the disjuncts), thus exclusively favoring those models M of $\mathcal{T}$ in which as many of the Q_j are satisfied as possible, that is the existentially closed models of $\mathcal{T}$.

It would of course be nice to extend this approach (or develop an alternative) to more than just these rather trivial unary languages. For example to the theory saying that the relation $<$ is transitive. In this case what is a 'sensible' probability to even give to $a_1 < a_2$? Certainly the simple method suggested here fails, but whether it can be suitable adapted to make it more applicable, whilst at the same time retaining credibility in relation to the original philosophical question apparently remains to be investigated.

Acknowledgement

We would like to thank the referees for their useful comments.

References

1. Bacchus, F., Grove, A.J., Halpern, J.Y., Koller, D.: Generating new beliefs from old. In: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1994), pp. 37–45 (1994)
2. Bacchus, F., Grove, A.J., Halpern, J.Y., Koller, D.: From statistical knowledge to degrees of belief. Artificial Intelligence 87, 75–143 (1996)
3. Barnett, O.W., Paris, J.B.: Maximum Entropy inference with qualified knowledge. Logic Journal of the IGPL 16(1), 85–98 (2008)
4. Carnap, R.: A basic system of inductive logic. In: Jeffrey, R.C. (ed.) Studies in Inductive Logic and Probability, vol. II, pp. 7–155. University of California Press (1980)
5. Chang, C.C., Keisler, H.J.: Model Theory. Studies in Logic and the Foundations of Mathematics, vol. 73. North Holland, Amsterdam (1973)
6. Dimitracopoulos, C., Paris, J.B., Vencovská, A., Wilmers, G.M.: A multivariate probability distribution based on the propositional calculus, Manchester Centre for Pure Mathematics, University of Manchester, UK, preprint number 1999/6, `http://www.maths.manchester.ac.uk/~jeff/`
7. Fitelson, B.: Inductive Logic, `http://fitelson.org/il.pdf`
8. Hodges, W.: Model Theory. Cambridge University Press, Cambridge (1993)
9. Grove, A.J., Halpern, J.Y., Koller, D.: Random Worlds and Maximum Entropy. Journal of Artificial Intelligence Research 2, 33–88 (1994)
10. Grove, A.J., Halpern, J.Y., Koller, D.: Asymptotic conditional probabilities: the unary case. SIAM J. of Computing 25(1), 1–51 (1996)
11. Grove, A.J., Halpern, J.Y., Koller, D.: Asymptotic conditional probabilities: the non-unary case. J. Symbolic Logic 61(1), 250–276 (1996)
12. Hintikka, J., Niiniluoto, I.: An axiomatic foundation for the logic of inductive generalization. In: Jeffrey, R.C. (ed.) Studies in Inductive Logic and Probability, vol. II, pp. 158–181. University of California Press, Berkeley, Los Angeles (1980)
13. Johnson, W.E.: Probability: The deductive and inductive problems. Mind 41(164), 409–423 (1932)

14. Kuipers, T.A.F.: A survey of inductive systems. In: Jeffrey, R.C. (ed.) Studies in Inductive Logic and Probability, vol. II, pp. 183–192. University of California Press, Berkeley, Los Angeles (1980)
15. Kuipers, T.A.F.: On the generalization of the continuum of inductive methods to universal hypotheses. Synthese 37, 255–284 (1978)
16. Paris, J.B.: A short course on Inductive Logic. In: JAIST 2007 (2007), http://www.maths.manchester.ac.uk/~jeff
17. Paris, J.B.: The Uncertain Reasoner's Companion. Cambridge University Press, Cambridge (1994)
18. Paris, J.B.: On the distribution of probability functions in the natural world. In: Hendricks, V.F., Pedersen, S.A., Jøgensen, K.F. (eds.) Probability Theory: Philosophy, Recent History and Relations to Science. Synthese Library, vol. 297, pp. 125–145 (2001)
19. Paris, J.B.: Vencovská, On the applicability of maximum entropy to inexact reasoning. International Journal of Approximate Reasoning 3(1), 1–34 (1989)
20. Paris, J.B., Vencovská: A note on the inevitability of maximum entropy. International Journal of Approximate Reasoning 4(3), 183–224 (1990)
21. Paris, J.B., Vencovská: In defense of the maximum entropy inference process. International Journal of Approximate Reasoning 17(1), 77–103 (1997)
22. Paris, J.B., Vencovská, A.: Common sense and stochastic independence. In: Corfield, D., Williamson, J. (eds.) Foundations of Bayesianism, pp. 203–240. Kluwer Academic Press, Dordrecht (2001)
23. Rad, S.R.: PhD Thesis, Manchester University, Manchester, UK (to appear)
24. http://en.wikipedia.org/wiki/Existentially_closed_model

Using α-CTL to Specify Complex Planning Goals

Silvio do Lago Pereira and Leliane Nunes de Barros

Institute of Mathematics and Statistics, University of Sao Paulo

Abstract. The temporal logic CTL has been the preferred specification language in the model checking framework. However, when this framework is used for nondeterministic planning, it is not adequate to deal with many useful planning problems with temporally extended goals. This is because the validity of CTL formulas expressing such goals is not evaluated on the planning domain, but on the execution structure of the policy synthesized by the planning algorithm. In previous work we have presented a new variant of CTL, named α-CTL, which semantics can be defined directly on the planning domain. An advantage of this new logic is that plan synthesis can be obtained as a collateral effect of verifying the validity of a formula in the planning domain. In this paper we show how to use α-CTL to express some complex planning goals.

1 Introduction

Traditionally, classical AI Planning assumes that agent's actions have deterministic effects and that its goal is to reach a desired final state [9]. Although this assumption can simplify the planning task [4], it almost never corresponds to the reality. In practical applications of planning, actions have nondeterministic effects and, due to this fact, we often need to impose conditions (*extended goals*) that should be satisfied not only in the final state reached by the agent, but also in all the states visited by it in order to achieve its goal.

In the model checking framework, the temporal logic CTL [7] has been the preferred formalism to deal with nondeterminism; however, when this framework is used for planning, it is not adequate to express many useful planning goals (*e.g.*, "*try its best to achieve φ, while preserving ϕ*" [2,15]). In view of this limitation, a previous work [12] has proposed a planning algorithm to solve some extended goals. However, because this algorithm requires goals being expressed in an extra-logical language, it is difficult to combine it with results on temporally expressed goals from earlier works (*e.g.*, [1,11]). Another recent work [3] proposes a logical language, called P-CTL*, to specify extended goals. Although this language is very expressive, allowing for the specification of extended goals without leaving the temporal logic framework, no planning algorithm has been presented so far [18].

In [15] we have proposed a new logic based on CTL, called α-CTL, that allows us to specify extended goals and also to plan for them by using slightly modified versions of the standard CTL model checking algorithms. With α-CTL, we have not only a logical language to specify goals, but we can also derive algorithms

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 260–271, 2008.

that are capable of planning for many intuitive and useful extended goals. The proposed logic differs from other action logics found in literature, where formulas impose constraints over states and also over *actions* [13,14]. Although actions play an important role in the α-CTL's semantics, they are not used in the formula's composition. Indeed, when we specify extended goals, we want to impose constraints only over the states visited during plan execution and not over the actions used to compose the plan.

In this paper we show how to use α-CTL to express some complex planning goals, such as "*try its best to achieve φ, while preserving ϕ*". Although this type of goal has been described before in P-CTL*, we claim that our approach is simpler and demands less computational effort because the universal quantification over policies in P-CTL* is a costly operation.

The remainder of this paper is organized as follows. In Section 2, we briefly present the background on planning based on model checking techniques and emphasize the frailty of CTL, when it is used to formalize planning algorithms. In Section 3, we present the syntax and semantics of the proposed logic α-CTL and, in Section 4, we show how to specify goals in this new logic. In Section 5, we compare some goal specifications in P-CTL* [3] with corresponding specifications in α-CTL. Finally, in Section 6, we present our conclusions.

2 Planning Based on Model Checking

In this section we present the background on planning based on model checking, where goals are expressed by CTL formulas. With this presentation, we aim (*i*) to emphasize the frailty of the logic CTL, when it is used to formalize planning algorithms, and (*ii*) to justify the need of the new proposed logic α-CTL.

2.1 Domains, Problems and Solutions

The key idea underlying planning based on model checking is to solve planning problems model-theoretically [10]. In this approach, a *planning domain* is a nondeterministic finite state-transition system $\mathcal{D} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the nonempty sets of all possible states and actions in the domain, and $\mathcal{T}$ is the state-transition function defined as $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto 2^{\mathcal{S}}$. A *planning problem* in a planning domain $\mathcal{D}$ is defined by an *initial state* $s_0 \in \mathcal{S}$ and by a set of goal states $\mathcal{G} \subseteq \mathcal{S}$. A *policy* (or plan) $\pi : \mathcal{S} \mapsto \mathcal{A}$ is defined as a partial function from states to actions. The set $\mathcal{S}_\pi$ of states reached by a policy π is $\{s : (s,a) \in \pi\} \cup \{s' : (s,a) \in \pi \text{ and } s' \in \mathcal{T}(s,a)\}$. Given a policy π, the corresponding *execution structure* $\mathcal{D}_\pi$ is the subsystem of $\mathcal{D}$, with $\mathcal{S}_\pi$ as set of states, containing all transitions induced by the actions in policy π. Given a nondeterministic planning problem, we have three kinds of solutions:

- a *weak solution* is a policy that may achieve the goal, but due to nondeterminism, is not guaranteed to do so. A policy π is a weak solution if some path in $\mathcal{D}_\pi$, starting from s_0, reaches a state in $\mathcal{G}$ [5].

- a *strong solution* is a policy that always achieves the goal, in spite of nondeterminism. A policy π is a strong solution if the subsystem $\mathcal{D}_\pi$ is acyclic and all paths starting from s_0 reach a state in $\mathcal{G}$ [6].
- a *strong-cyclic solution* is a policy that always achieves the goal, under the assumption that execution will exit from all cycles. A policy π is a strong-cyclic solution if all paths in $\mathcal{D}_\pi$ starting from s_0 reach a state in $\mathcal{G}$ [8].

2.2 The Branching Time Temporal Logic CTL

The language of CTL [7] is defined over an alphabet $\mathbb{P}$ of atomic propositions and the symbols $\circ$ (*next*), $\Box$ (*invariantly*), $\Diamond$ (*finally*) and $\sqcup$ (*until*), combined with quantifiers $\exists$ and $\forall$, are used to represent temporal operators. The syntax of CTL is inductively defined as $\varphi ::= p \in \mathbb{P} \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \exists \circ \varphi \mid \exists\Box\varphi \mid \exists(\varphi \sqcup \varphi')$, and some useful abbreviations are:

$$
\begin{array}{ll}
\phi \vee \varphi & \doteq \neg(\neg\phi \wedge \neg\varphi) \\
\exists\Diamond\varphi & \doteq \exists(\top \sqcup \varphi) \\
\forall\Diamond\varphi & \doteq \forall(\top \sqcup \varphi) \\
\forall \circ \varphi & \doteq \neg\exists \circ \neg\varphi \\
\forall\Box\varphi & \doteq \neg\exists\Diamond\neg\varphi \\
\forall(\phi \sqcup \varphi) & \doteq \neg(\exists(\neg\varphi \sqcup \neg(\phi \vee \varphi)) \wedge \neg\exists\Box\neg\varphi)
\end{array}
$$

The semantics of CTL is defined over a *Kripke structure* $\mathcal{K} = \langle \mathcal{S}, \mathcal{T}, \mathcal{L} \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$ is the state transition relation and $\mathcal{L} : \mathcal{S} \mapsto 2^{\mathbb{P}}$ is the state labeling function. A *path* in $\mathcal{K}$ is a sequence of states $s_0, s_1, \ldots$ such that $s_i \in \mathcal{S}$ and $(s_i, s_{i+1}) \in \mathcal{T}$, for all $i \geq 0$. Given $\mathcal{K}$ and $s_0 \in \mathcal{S}$, the CTL satisfiability relation is defined as:

$$
\begin{array}{ll}
(\mathcal{K}, s_0) \vDash p & \text{iff } p \in \mathcal{L}(s_0); \\
(\mathcal{K}, s_0) \vDash \neg\varphi & \text{iff } (\mathcal{K}, s_0) \vDash \varphi; \\
(\mathcal{K}, s_0) \vDash (\phi \wedge \varphi) & \text{iff } (\mathcal{K}, s_0) \vDash \phi \text{ and } (\mathcal{K}, s_0) \vDash \varphi; \\
(\mathcal{K}, s_0) \vDash \exists \circ \varphi & \text{iff there exists a path } s_0, s_1, \ldots \text{ in } \mathcal{K} \text{ such that } (\mathcal{K}, s_1) \vDash \varphi; \\
(\mathcal{K}, s_0) \vDash \exists\Box\varphi & \text{iff there exists a path } s_0, s_1, \ldots \text{ in } \mathcal{K} \text{ such that } (\mathcal{K}, s_i) \vDash \varphi, \\
 & \text{for } i \geq 0; \\
(\mathcal{K}, s_0) \vDash \exists(\phi \sqcup \varphi) & \text{iff there exists a path } s_0, s_1, \ldots \text{ in } \mathcal{K} \text{ such that there exists} \\
 & i \geq 0 \text{ for which } (\mathcal{K}, s_i) \vDash \varphi', \text{ and for } 0 \leq j < i,\ (\mathcal{K}, s_j) \vDash \varphi.
\end{array}
$$

2.3 Goal Specification Using CTL

By using CTL it is possible to express goals that take into account nondeterminism [16,17]. In this approach, we consider that states in the planning domain $\mathcal{D}$ are labeled with subsets of $2^{\mathbb{P}}$. Let $g \in \mathbb{P}$ be an atomic proposition holding only on goal states $\mathcal{G} = \{s \in \mathcal{S} : g \in \mathcal{L}(s)\}$. It is possible to specify the following reachability goals:

- $\exists\Diamond g$: requiring a weak solution;
- $\forall\Diamond g$: requiring a strong solution; and
- $\forall\Box\exists\Diamond g$: requiring a strong-cyclic solution.

Let $\mathcal{P} = \langle \mathcal{D}, s_0, \varphi \rangle$ be a planning problem, where $\mathcal{D}$ is a planning domain, s_0 is an initial state and φ is a goal specified in CTL. Let π be a policy in $\mathcal{D}$. Thus, π is a *solution* to $\mathcal{P}$ if and only if $(\mathcal{D}_\pi, s_0) \vDash \varphi$, where the execution structure $\mathcal{D}_\pi$ is interpreted as a Kripke structure. The semantics of CTL can be easily used to formalize algorithms to perform plan validation. However, it is not clear how we can derive the policy π for φ directly from the semantics of CTL.

3 The New Branching Time Temporal Logic α-CTL

To motivate the need of a new temporal logic for planning based on model checking, we start with a simple example.

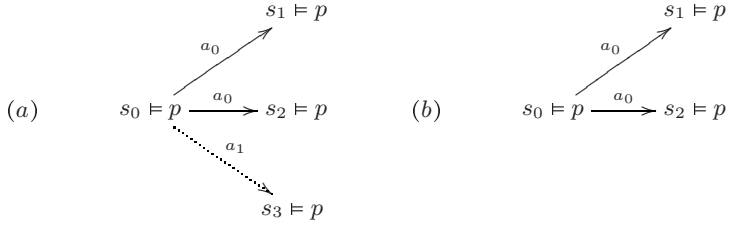


Fig. 1. A planning domain $\mathcal{D}^1$ and the execution structure $\mathcal{D}^1_\pi$, for policy $\pi = \{(s_0, a_0)\}$

Example 1. *Consider the domain $\mathcal{D}^1$ (Fig. 1-a) and suppose that the goal is necessarily to reach a successor of the state s_0 where property p holds. In* CTL*, this goal is expressed by the formula $\forall \bigcirc p$ and a state satisfies this formula only if all its successors satisfy p; therefore, we have that $(\mathcal{D}^1, s_0) \vDash \forall \bigcirc p$. Thus, even though the action a_0 can necessarily lead to a state where p holds, according to the* CTL*'s semantics, the planning goal cannot be achieved. Now, if we consider the policy $\pi = \{(s_0, a_0)\}$ and the corresponding execution structure $\mathcal{D}^1_\pi$ (Fig. 1-b), clearly, we have that $(\mathcal{D}^1_\pi, s_0) \vDash \forall \bigcirc p$.* ◆

With Example 1, we show that CTL is not adequate to formalize plan synthesis algorithms (although it is adequate to formalize plan validation algorithms). We claim that the inability of CTL to formalize synthesis algorithms is mainly due to the fact that its semantics does not take into account the *quality* of transitions (*i.e.*, the actions that produce the transitions) and this is an essential information in plan synthesis algorithms. To overcome this lack of expressiveness in CTL, we propose a new branching time temporal logic, called α-CTL, where the actions play a fundamental role in its formal semantics.

3.1 The Syntax of α-CTL

In CTL, a formula $\forall \bigcirc \varphi$ holds on state s if only if φ holds on *all* successors of s, independently of the actions labeling the transitions from s to its successors. In α-CTL, to enforce that actions play an important role, we use a different set of

"dotted" symbols to represent temporal operators: $\odot$ (*next*), $\boxdot$ (*invariantly*), $\diamondsuit$ (*finally*) and $\sqcup$ (*until*).

Definition 1 (α-CTL's syntax)**.** *Let $p \in \mathbb{P}$ be an atomic proposition. The syntax of α-CTL is inductively defined as:*

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \exists \odot \varphi \mid \forall \odot \varphi \mid \exists \boxdot \varphi \mid \forall \boxdot \varphi \mid \exists(\varphi \sqcup \varphi') \mid \forall(\varphi \sqcup \varphi')$$

◆

According to the α-CTL's syntax, well-formed formulas are in *negative normal form*, *i.e.*, the scope of negation is restricted to the atomic propositions (this allows us to define a fixpoint semantics for the formulas). Furthermore, all temporal operators are prefixed by a path quantifier ($\exists$ or $\forall$). The temporal operators derived from $\diamondsuit$ are $\exists \diamondsuit \varphi \doteq \exists(\top \sqcup \varphi)$ and $\forall \diamondsuit \varphi \doteq \forall(\top \sqcup \varphi)$.

Although actions are essential in the semantics of α-CTL, they are not used to compose the formulas. Indeed, when we specify a planning goal, we wish to impose constraints only over the states visited during the execution of the policy. In general, constraints over the actions that will be used to compose a plan are not relevant when we specify the planning goal. For this reason, actions logics ([13], [14]), which allows constraints over actions, are also inadequate to formalize plan synthesis algorithms.

3.2 The Models in α-CTL

A *model* in α-CTL is a *Kripke Transition System* (KTS). Intuitively, a KTS is a generalization of *Labeled Transitions Systems* (*i.e.*, diagrams with labeled transitions) and *Kripke Structures* (*i.e.*, diagrams with labeled states). Let $\mathbb{P}$ be a finite set o atomic propositions and let $\mathbb{A}$ be a finite set of actions containing the *trivial* action τ. In a KTS with *signature* $(\mathbb{P}, \mathbb{A})$, states are labeled with subsets of $\mathbb{P}$ and transitions are labeled with elements of $\mathbb{A}$.

Definition 2 (α-CTL's model)**.** *A Kripke Transition System (KTS) with signature $(\mathbb{P}, \mathbb{A})$ is a structure $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$, where:*

- *$\mathcal{S} \neq \varnothing$ is a finite set of states;*
- *$\mathcal{L} : \mathcal{S} \mapsto 2^{\mathbb{P}}$ is a state labeling function;*
- *$\mathcal{T} : \mathcal{S} \times \mathbb{A} \mapsto 2^{\mathcal{S}}$ is a transition labeling function.* ◆

We assume that $\top \in \mathcal{L}(s)$, for all state $s \in \mathcal{S}$, and that $\mathcal{T}$ is a total function (all states have a reflexive transition labeled with action τ, such that $\mathcal{T}(s, \tau) = \{s\}$). The justification for these reflexive transitions is that, being $\mathcal{D}$ a temporal model for α-CTL's formulas, terminal states in this model should persist infinitely in time. Given two states $s, s' \in \mathcal{S}$ and an action $\alpha \in \mathbb{A}$, we say that s' is a *α-successor* of s if $s' \in \mathcal{T}(s, \alpha)$. The set of α-successors of s is denoted by $\mathcal{T}(s, \alpha)$.

Definition 3 (preimage)**.** *Let $Y \subseteq \mathcal{S}$ be a set of states. The weak preimage of Y, denoted by $\mathcal{T}_{\exists}^{-}(Y)$, is the set $\{s \in \mathcal{S} : \exists \alpha \in \mathbb{A}. \mathcal{T}(s, \alpha) \cap Y \neq \varnothing\}$ and the strong preimage of Y, denoted by $\mathcal{T}_{\forall}^{-}(Y)$, is the set $\{s \in \mathcal{S} : \exists \alpha \in \mathbb{A}. \varnothing \neq \mathcal{T}(s, \alpha) \subseteq Y\}$.* ◆

3.3 The Semantics of α-CTL

Intuitively, a state s satisfies a formula $\forall \odot \varphi$ (or $\exists \odot \varphi$) if *exists* an action α that, when executed in s, *necessarily* (or *possibly*) reaches an imediate successor of s which satisfies the formula φ. In other words, the modality $\odot$ represents the set of α-successors of s, *for some particular action* α; the quantifier $\forall$ requires that *all* these α-successors satisfy φ; and the quantifier $\exists$ requires that *some* of these α-successors satisfy φ.

For example, consider again the domain $\mathcal{D}^1$ (Fig. 1-a). In this domain, $\mathcal{T}(s_0, a_0) = \{s_1, s_2\}$ and both states s_1 and s_2 satisfy p. Thus, by the α-CTL's semantics it follows that $(\mathcal{D}^1, s_0) \vDash \forall \odot p$ (note that by the CTL's semantics, we would have $(\mathcal{D}^1, s_0) \vDash \forall \bigcirc p$). Furthermore, in α-CTL we can have both $(\mathcal{D}^1, s_0) \vDash \forall \odot p$ and $(\mathcal{D}^1, s_0) \vDash \forall \odot \neg p$, at the same time. This is possible due to the fact that each occurrence of the modality $\odot$ can instantiate a different action α and, consequently, the quantification can be made over different sets of α-successors of the state s_0. However, the fact $(\mathcal{D}^1, s_0) \vDash \forall \odot p \wedge \forall \odot \neg p$ does not mean there exists a policy to achieve both subgoals at the same time; it only means that from state s_0 we can choose to achieve p or $\neg p$.

The semantics of the global temporal operators ($\boxdot$ and $\sqcup$) is derived from the semantics of the local temporal operator $\odot$, by using least fixpoint operations (μ) and greatest fixpoint operations (ν).

Definition 4 (Formula's intension)**.** *Let $\mathcal{D} = \langle \mathcal{S}, \mathcal{L}, \mathcal{T} \rangle$ be a* KTS *with signature $(\mathbb{P}, \mathbb{A})$ and let $p \in \mathbb{P}$ be an atomic proposition. The intension of an α-CTL formula φ in $\mathcal{D}$ (i.e., the set of states satisfying φ in $\mathcal{D}$), denoted by $[\![\varphi]\!]_{\mathcal{D}}$, is defined as:*

- $[\![p]\!]_{\mathcal{D}} = \{s : p \in \mathcal{L}(s)\}$
- $[\![\neg p]\!]_{\mathcal{D}} = \mathcal{S} \setminus [\![p]\!]_{\mathcal{D}}$
- $[\![\varphi \wedge \varphi']\!]_{\mathcal{D}} = [\![\varphi]\!]_{\mathcal{D}} \cap [\![\varphi']\!]_{\mathcal{D}}$
- $[\![\varphi \vee \varphi']\!]_{\mathcal{D}} = [\![\varphi]\!]_{\mathcal{D}} \cup [\![\varphi']\!]_{\mathcal{D}}$
- $[\![\exists \odot \varphi]\!]_{\mathcal{D}} = \mathcal{T}_{\exists}^{-}([\![\varphi]\!]_{\mathcal{D}})$
- $[\![\forall \odot \varphi]\!]_{\mathcal{D}} = \mathcal{T}_{\forall}^{-}([\![\varphi]\!]_{\mathcal{D}})$
- $[\![\exists \boxdot \varphi]\!]_{\mathcal{D}} = \nu Y.([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\exists}^{-}(Y))$
- $[\![\forall \boxdot \varphi]\!]_{\mathcal{D}} = \nu Y.([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\forall}^{-}(Y))$
- $[\![\exists(\varphi \sqcup \varphi')]\!]_{\mathcal{D}} = \mu Y.([\![\varphi']\!]_{\mathcal{D}} \cup ([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\exists}^{-}(Y)))$
- $[\![\forall(\varphi \sqcup \varphi')]\!]_{\mathcal{D}} = \mu Y.([\![\varphi']\!]_{\mathcal{D}} \cup ([\![\varphi]\!]_{\mathcal{D}} \cap \mathcal{T}_{\forall}^{-}(Y)))$ ◆

Definition 5 (α-CTL's semantics)**.** *Let $\mathcal{D} = \langle \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$ be a* KTS *with signature $(\mathbb{P}, \mathbb{A})$, $s \in \mathcal{S}$ be a state in $\mathcal{D}$ and φ be an α-CTL formula. The α-CTL's satisfiability relation is defined as: $(\mathcal{D}, s) \vDash \varphi \Leftrightarrow s \in [\![\varphi]\!]_{\mathcal{D}}$* ◆

The notion of intension of a formula φ can be reformulated such that $[\![\varphi]\!]_{\mathcal{D}}$ turns out to be a subsystem of $\mathcal{D}$ containing all the states satisfying φ, as well as all transitions considered during the selection of these states (we need essentially to redefine preimage functions such that they collect the pair (s, a), whenever the action a is considered to show that s satisfies the property φ). Note that with this reformulatiom we can synthesize plans as a collateral effect of the verification of

property φ in the system $\mathcal{D}$. Therefore, a planning algorithm for extended goals expressed in α-CTL can be formalized through the definition of this new notion of $[\![\varphi]\!]_{\mathcal{D}}$. Indeed, we have implemented a simple prototype of a planning system by codifying α-CTL's semantic rules directly in PROLOG.

4 Goal Specification in α-CTL

As an example, consider the planning domain $\mathcal{D}^2$ (Fig. 2), adapted from [3]. Also, consider the following policies in this domain:

$\pi_1 = \{(s_0, a), (s_1, a), (s_3, b)\}$
$\pi_2 = \{(s_0, a), (s_1, a), (s_3, c)\}$
$\pi_3 = \{(s_0, a), (s_1, b), (s_3, b)\}$
$\pi_4 = \{(s_0, a), (s_1, b), (s_3, c)\}$

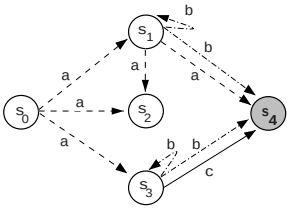


Fig. 2. The planning domain $\mathcal{D}^2$, where s_4 is the only state that satisfies property g

Now, suppose that the agent is initially in state s_0 and that its goal is to try its best to reach a state satisfying the property g, *i.e.*, $\mathcal{G} = [\![g]\!]_{\mathcal{D}^2} = \{s_4\}$. Clearly, if we specify this goal by the formula $\forall \diamond g$ (*strong planning*), it turns out to be unachievable from the state s_0. Due to nondeterminism, after the execution of action a in the state s_0, we cannot guarantee that a goal state still can be reached (*e.g.*, action a can lead to state s_2). The same happens if we specify the goal by the formula $\forall \boxdot \exists \diamond p$ (*strong-cyclic planning*). On the other hand, if the goal is specified by the formula $\exists \diamond g$ (*weak planning*), the policy $\pi_1 = \{(s_0, a), (s_1, a), (s_3, b)\}$ could be considered as solution. However, by following this policy, the agent is not trying its best. As we can see, the formulas $\forall \diamond g$, $\forall \boxdot \exists \diamond p$ and $\exists \diamond g$ are inappropriate to express the planning goal.

4.1 An Agent Who Tries Its Best

Starting from s_0 (Fig. 2), the agent has to choose between to remain in the same state (by selecting action τ) or to move to another state (by selecting action a). Clearly, if the agent is trying its best, it should select action a. With this choice, the agent does not guarantee to achieve the goal; however, at this point, this is the best that it can do (there is no strong nor strong-cyclic solution from this state and the agent should be "happy" with a weak solution). After executing action a in s_0, the agent can reach one of these states:

- s_1: from this state, the best choice is action b, which always maintains the possibility of reaching the goal (*i.e.*, the agent prefer a strong-cyclic solution);
- s_2: from this state, the agent can no longer reach the goal;
- s_3: from this state, the best choice is action c, which will necessarily reach the goal (*i.e.*, from this state, the agent should prefer a strong solution).

In other words, if the agent is trying its best, it should consider the policy $\pi_4 = \{(s_0, a), (s_1, b), (s_3, c)\}$ as the only possible solution to the planning problem of "try its best to achieve g".

Through this example we can see that when an agent is trying its best, it should alter its expectation during planning [3]. The question is how we can express the planning goal so that the agent can alter its expectation during the process of planning trying to do its best. By expressing the goal as $\exists \diamondsuit g$, we do not allow that the agent alters its expectation: it will be always satisfied with a weak solution, even if a strong or a strong-cyclic solution exists. On the other hand, by expressing the goal as $\forall \diamondsuit g$ or $\forall \boxdot \exists \diamondsuit g$, there is no solution from s_0.

4.2 Specifying "Try Its Best to Achieve g" in α-CTL

For now, let's forget about the initial state and consider goal states in system $\mathcal{D}^2$, denoted by $[\![g]\!]_{\mathcal{D}^2}$. From each state in this system, we can try to find a strong policy to reach states in $[\![g]\!]_{\mathcal{D}^2}$. This can be done by considering the subsystem[1] $[\![\forall \diamondsuit g]\!]_{\mathcal{D}^2}$ (Fig. 3-a). Next, from each remaining state in the system (*i.e.*, states that do not appear in $[\![\forall \diamondsuit g]\!]_{\mathcal{D}^2}$), we can try to find strong-cyclic policies. Analogously, this can be done by considering subsystems $[\![\exists \diamondsuit \forall \diamondsuit g]\!]_{\mathcal{D}^2}$ (Fig. 3-b) and $[\![\forall \boxdot \exists \diamondsuit \forall \diamondsuit g]\!]_{\mathcal{D}^2}$ (Fig. 3-c). Finally, we can try to find weak policies by considering the subsystem $[\![\exists \diamondsuit \forall \boxdot \exists \diamondsuit \forall \diamondsuit g]\!]_{\mathcal{D}^2}$ (Fig. 3-d). By proceeding in this way, we allow to the agent to alter its expectation during the planning, given preference to better solutions. An illustration of this policy synthesis process is given in Fig. 4. Now, in order to check if there exists a solution from the initial state is enough to see if $s_0 \in [\![\exists \diamondsuit \forall \boxdot \exists \diamondsuit \forall \diamondsuit g]\!]_{\mathcal{D}^2}$. Note that the subsystem $[\![\exists \diamondsuit \forall \boxdot \exists \diamondsuit \forall \diamondsuit g]\!]_{\mathcal{D}^2}$ is exactly the execution structure corresponding to the policy $\pi_4 = \{(s_0, a), (s_1, b), (s_3, c)\}$, which is the solution of the proposed planning problem.

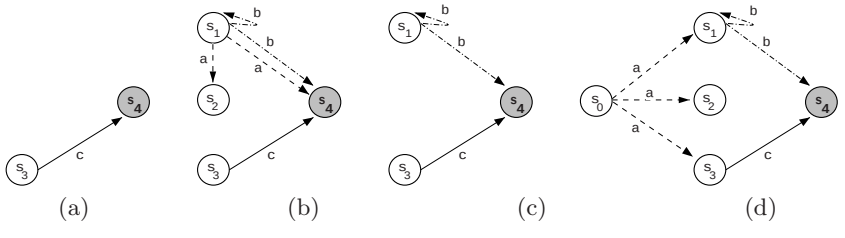


Fig. 3. Subsystems of $\mathcal{D}^2$ satisfying the formulas $\forall \diamondsuit g$, $\exists \diamondsuit \forall \diamondsuit g$, $\forall \boxdot \exists \diamondsuit \forall \diamondsuit g$ and $\exists \diamondsuit \forall \boxdot \exists \diamondsuit \forall \diamondsuit g$, respectively

[1] According to α-CTL's semantics, $[\![g]\!]_{\mathcal{D}^2} \subseteq [\![\forall \diamondsuit g]\!]_{\mathcal{D}^2}$, so even if there is no possible extension of the subsystem $[\![g]\!]_{\mathcal{D}^3}$, we can still consider the subsystem $[\![\forall \diamondsuit g]\!]_{\mathcal{D}^2}$.

5 Comparing Specifications in α-CTL and P-CTL*

In this section, we show how the extended goals discussed in [3] can be expressed in α-CTL, with the advantage that we can also offer algorithms to solve them.

P-CTL* [3] extends π-CTL* [2] with quantification over policies ($\mathcal{EP}$ and $\mathcal{AP}$). Let p denote an atomic proposition, sf denote a state formula, and pf denote a path formula. The syntax of P-CTL* is inductively defined as:

$$sf ::= p \mid sf \wedge sf \mid sf \vee sf \mid \neg sf \mid \mathrm{E}pf \mid \mathrm{A}pf \mid \mathrm{E}_\pi pf \mid \mathrm{A}_\pi pf \mid \mathcal{EP}sf \mid \mathcal{AP}sf$$
$$pf ::= sf \mid pf \wedge pf \mid pf \vee pf \mid \neg pf \mid \bigcirc pf \mid \Diamond pf \mid \Box pf \mid pf \sqcup pf$$

The state formulas' semantics is defined w.r.t. a triple (s_j, Φ, π), where s_j is a state, Φ is the transition function, and π is a policy[2]. A policy π is *consistent* w.r.t. to a transition function Φ if, for all states s, $\Phi(s, \pi(s))$ is a nonempty set.

$(s_j, \Phi, \pi) \vDash p$	iff p is true in s_j
$(s_j, \Phi, \pi) \vDash \neg sf$	iff $(s_j, \Phi, \pi) \vDash sf$
$(s_j, \Phi, \pi) \vDash sf_1 \wedge sf_2$	iff $(s_j, \Phi, \pi) \vDash sf_1$ and $(s_j, \Phi, \pi) \vDash sf_2$
$(s_j, \Phi, \pi) \vDash sf_1 \vee sf_2$	iff $(s_j, \Phi, \pi) \vDash sf_1$ or $(s_j, \Phi, \pi) \vDash sf_2$
$(s_j, \Phi, \pi) \vDash \mathrm{E}pf$	iff there exists a path σ in Φ starting from s_j such that $(s_j, \Phi, \pi, \sigma) \vDash pf$
$(s_j, \Phi, \pi) \vDash \mathrm{A}pf$	iff for all paths σ in Φ starting from s_j we have $(s_j, \Phi, \pi, \sigma) \vDash pf$
$(s_j, \Phi, \pi) \vDash \mathrm{E}_\pi pf$	iff there exists a path σ in Φ starting from s_j consistent with π such that $(s_j, \Phi, \pi, \sigma) \vDash pf$
$(s_j, \Phi, \pi) \vDash \mathrm{A}_\pi pf$	iff for all paths σ in Φ starting from s_j consistent with π we have that $(s_j, \Phi, \pi, \sigma) \vDash pf$
$(s_j, \Phi, \pi) \vDash \mathcal{EP}sf$	iff there exists π' consistent with Φ such that $(s_j, \Phi, \pi') \vDash sf$
$(s_j, \Phi, \pi) \vDash \mathcal{AP}sf$	iff for all π' consistent with Φ we have that $(s_j, \Phi, \pi, \sigma) \vDash sf$

The path formulas' semantics is defined w.r.t. tuple (s_j, Φ, π, σ), where s_j, Φ and π are as before and σ in an infinite sequence of states $s_j, s_{j+1}, \ldots$, called a path.

$(s_j, \Phi, \pi, \sigma) \vDash sf$	iff $(s_j, \Phi, \pi) \vDash sf$
$(s_j, \Phi, \pi, \sigma) \vDash \neg pf$	iff $(s_j, \Phi, \pi, \sigma) \vDash pf$
$(s_j, \Phi, \pi, \sigma) \vDash sf_1 \wedge sf_2$	iff $(s_j, \Phi, \pi, \sigma) \vDash sf_1$ and $(s_j, \Phi, \pi, \sigma) \vDash sf_2$
$(s_j, \Phi, \pi, \sigma) \vDash sf_1 \vee sf_2$	iff $(s_j, \Phi, \pi, \sigma) \vDash sf_1$ or $(s_j, \Phi, \pi, \sigma) \vDash sf_2$
$(s_j, \Phi, \pi, \sigma) \vDash \bigcirc pf$	iff $(s_{j+1}, \Phi, \pi, \sigma) \vDash \bigcirc pf$
$(s_j, \Phi, \pi, \sigma) \vDash \Diamond pf$	iff $(s_k, \Phi, \pi, \sigma) \vDash pf$, for some $k \geq j$
$(s_j, \Phi, \pi, \sigma) \vDash \Box pf$	iff $(s_k, \Phi, \pi, \sigma) \vDash pf$, for all $k \geq j$
$(s_j, \Phi, \pi, \sigma) \vDash (pf_1 \sqcup pf_2)$	iff there exists $k \geq j$ such that $(s_k, \Phi, \pi, \sigma) \vDash pf_2$, and for all $j \leq i < k$, $(s_i, \Phi, \pi, \sigma) \vDash pf_1$

By using P-CTL*, one can express some useful extended goals that cannot be expressed in CTL. Examples of such goals (Fig. 2) are [3]:

- $\varphi_s \doteq \mathrm{A}_\pi\Box(\mathcal{EP}\mathrm{A}_\pi\Diamond g \rightarrow \mathrm{A}_\pi\Diamond g)$: this requires that all along the path following the given policy, if there is a strong policy for g, then the policy chosen must be a strong policy. Only the policies π_2 and π_4 can reach this goal.

[2] As in CTL, the semantics of P-CTL* is defined over a given policy (= *validation*).

- $\varphi_c \doteq \mathrm{A}_\pi\Box(\mathcal{EP}\mathrm{A}_\pi\Box(\mathrm{E}_\pi\Diamond g) \rightarrow \mathrm{A}_\pi\Box(\mathrm{E}_\pi\Diamond g))$: this goal requires that all along the trajectory following the given policy, if there is a strong-cyclic policy for g, then the policy chosen must be a strong-cyclic policy for g. Only the policies π_3 and π_4 can satisfy this goal.
- $\varphi_w \doteq \mathrm{A}_\pi\Box(\mathcal{EP}\mathrm{E}_\pi\Diamond g \rightarrow \mathrm{E}_\pi\Diamond g)$: this requires that all along the path following the given policy, if there is a policy that reaches g then the policy chosen must make it. All the policies π_1, π_2, π_3 and π_4 can satisfy this goal.
- $\varphi_b \doteq \varphi_s \wedge \varphi_c \wedge \varphi_w$: this goal requires that all along the trajectory following the given policy, if there is a strong policy for g, then the policy chosen must be a strong policy; else, if there is a strong-cyclic for g, then the policy chosen must be a strong-cyclic policy; and else, if there is a policy that makes g reachable then the policy chosen must make g reachable. This can be considered as a formal specification of the goal "*trying ones best to reach* g". Only policy π_4 can satisfy this goal.

In Table 1, we present the corresponding specifications of φ_s, φ_c, φ_w and φ_b in α-CTL. Note that the P-CTL*'s semantics requires goals to be defined over given policies, not giving a hint about how to select actions. On the other hand, in α-CTL extended goals are expressed by only imposing constraints over states, leaving the selection of actions for its semantics – a key feature to formalize a planning algorithm. In fact, in right column of Table 1 we can see the policies that have been synthesized by our α-CTL-planner for all the listed extended goals; on that account, they are the same policies predicted by the work on P-CTL* [3].

Table 1. Correspondence between some specifications in P-CTL* and α-CTL

P-CTL*	α-CTL	Solution policies
φ_s	$\exists \diamondsuit \forall \diamondsuit g$	π_2, π_4
φ_c	$\exists \diamondsuit \forall \boxdot \exists \diamondsuit g$	π_3, π_4
φ_w	$\exists \diamondsuit g$	π_1, π_2, π_3, π_4
φ_b	$\exists \diamondsuit \forall \boxdot \exists \diamondsuit \forall \diamondsuit g$	π_4

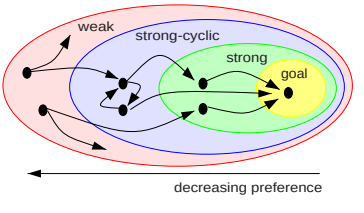


Fig. 4. Policy synthesis process

6 Conclusion

In previous work [15] we have proposed a new branching time temporal logic, called α-CTL, where actions play a fundamental role in its formal semantics

(something that we have not found through the literature so far). We argued that by using α-CTL we can synthesize plans as a collateral effect of the verification of a property φ (*planning goal*) in a system (*planning domain*) $\mathcal{D}$. Thus a planning algorithm for extended goals expressed in α-CTL can be directly formalized by the α-CTL's semantic rules

In this paper we considered how to specify and solve some extended goals, when nondeterministic actions are taken into account [17]. In particular, we have considered goals such as "*try your best to achieve ...*", for which our α-CTL planner[3] has synthesized the same policy predicted for the corresponding goal in P-CTL* [3]. We believe that our approach is simpler and demands less computational effort than that proposed in [3], because the quantification over policies in P-CTL* is a costly operation. Although in this paper we have establish an intuitive comparison between specifications of extended goals in P-CTL* and the corresponding specifications in α-CTL, our claim is that with our implemented planner (*i.e*, the α-CTL-planner) we can synthesize plans for these type of goals, while the P-CTL* we can only be used to verify plans for them.

References

1. Bacchus, F., Kabanza, F.: Planning for temporally extended goals. In: AAAI 1996, pp. 1215–1222. AAAI Press, Menlo Park (1996)
2. Baral, C., Zhao, J.: Goal specification in presence of non-deterministic actions. In: Mántaras, R.L., Saitta, L. (eds.) ECAI, pp. 273–277 (2004)
3. Baral, C., Zhao, J.: Goal specification, non-determinism and quantifying over policies. In: AAAI (2006)
4. Bylander, T.: The computational complexity of propositional STRIPS planning. Artificial Intelligence 69(1-2), 165–204 (1994)
5. Cimatti, A., Giunchiglia, F., Giunchiglia, E., Traverso, P.: Planning via model checking: A decision procedure for AR. In: Steel, S. (ed.) ECP 1997. LNCS, vol. 1348, pp. 130–142. Springer, Heidelberg (1997)
6. Cimatti, A., Roveri, M., Traverso, P.: Strong planning in non-deterministic domains via model checking. In: AIPS, pp. 36–43 (1998)
7. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching-time temporal logic. In: Logic of Programs, Workshop, pp. 52–71. Springer, London (1982)
8. Daniele, M., Traverso, P., Vardi, M.Y.: Strong cyclic planning revisited. In: Biundo, S., Fox, M. (eds.) ECP 1999. LNCS, vol. 1809, pp. 35–48. Springer, Heidelberg (2000)
9. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann Publishers, San Francisco (2004)
10. Giunchiglia, F., Traverso, P.: Planning as model checking. In: Biundo, S., Fox, M. (eds.) ECP 1999. LNCS, vol. 1809, pp. 1–20. Springer, Heidelberg (2000)
11. Kabanza, F., Barbeau, M., St.-Denis, R.: Planning control rules for reactive agents. Artificial Intelligence 95(1), 11–67 (1997)
12. Dal Lago, U., Pistore, M., Traverso, P.: Planning with a language for extended goals. In: Eighteenth national conference on Artificial intelligence, pp. 447–454. AAAI, Menlo Park (2002)

[3] Due to lack of space, we have omitted the algorithm specification.

13. De Nicola, R., Vaandrager, F.: Action versus state based logics for transition systems. In: Proceedings of the LITP spring school on theoretical computer science on Semantics of systems of concurrent processes, pp. 407–419. Springer, New York (1990)
14. Pecheur, C., Raimondi, F.: Symbolic Model Checking of Logics with Actions. In: Edelkamp, S., Lomuscio, A. (eds.) MoChArt IV. LNCS (LNAI), vol. 4428, pp. 113–128. Springer, Heidelberg (2007)
15. Pereira, S.L., Barros, L.N.: A logic-based agent that plans for extended reachability goals. Autonomous Agents and Multi-Agent Systems 9034, 327–344 (2008)
16. Pistore, M., Bettin, R., Traverso, P.: Symbolic techniques for planning with extended goals in non-deterministic domains (2001)
17. Pistore, M., Traverso, P.: Planning as model checking for extended goals in non-deterministic domains. In: IJCAI, pp. 479–486 (2001)
18. Thiébaux, S., Gretton, C., Slaney, J., Price, D., Kabanza, F.: Decision-theoretic planning with non-markovian rewards. JAIR 25(2), 75–118 (2006)

Hyperintensional Questions

Carl Pollard

INRIA-Lorrraine, Universitat Rovira i Virgili, and Ohio State University
pollard@ling.ohio-state.edu

1 Introduction

It has been known for decades that Montague's (1974 [1970]) possible-worlds semantics, which follows Kripke 1963 in treating worlds as unanalyzed primitives and propositions as sets of worlds, does not provide enough meaning distinctions to make correct predictions about a wide range of natural-language entailment patterns.[1] This **granularity** problem, as it has come to be known, has many dimensions, of which the best known is that two declarative sentences which entail each other must express the same proposition. This is because entailment is modelled by the subset inclusion relation on the powerset of the set of propositions, and that relation is irretrievably antisymmetric. The most notorious consequence of this antisymmetry of entailment is the so-called **logical omniscience** problem, that (assuming knowledge is a relation between individuals and propositions) anyone who knows at least one necessary truth (e.g. that s/he is self-identical, or that two is even) must know **every** necessary truth, even an unresolved mathematical conjecture or its denial (whichever is true). Thus, e.g. if Paris Hilton knows that Paris Hilton is Paris Hilton, then she must also know that every nontrivial zero of the zeta-function has real part 1/2, if that is indeed the case, or else she must know that this is *not* the case, if indeed it is not. In short, it seems to be a consequence of MS that a celebrity hotel heiress devoted to parties and shopping knows whether the Riemann Hypothesis is true. This is just one of the unsavory consequences of MS.

From the point of view of MS, the most conservative response to Granularity, and one which remains popular among linguistic semanticists, is to argue, along the lines of Stalnaker (1984), that it is naive to perceive it as a real problem. In effect, Paris Hilton really *does* either know that R or know that not R, she just doesn't *know* that she does. On the other hand, Granularity has been quite widely viewed as a serious foundational problem, and a wide range of ingenious

[1] For much useful discussion, some of it in the distant past, and comments on earlier versions, I wish to thank David Dowty, Jonathan Ginzburg, Howard Gregory, Martin Jansche, Brad Kolb, Tim Leffel, Scott Martin, Drew Moshier, Reinhard Muskens, Andy Plummer, Phil Scott, and Ken Shan. For their help in providing the conditions that made this research possible, I am grateful to Carlos Martin Vide, Philippe de Groote, and the Department of Linguistics and College of Humanities of Ohio State University. The research reported here was supported by grant no. 2006PIV10036 from the Agència de Gestió d'Ajuts Universitaris i de Recerca of the Generalitat de Catalunya.

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 272–285, 2008.

and technically sophisticated replacements for MS have been proposed in response. Among these, to mention just a few of the best known proposals, have been have been Intentional Semantics (Thomason 1980), Situation Semantics (Barwise and Perry 1983), and Property Theory (Chierchia and Turner 1988). For summaries and critical assessment of these and other proposals, together with a more sophisticated version of Property Theory, see Fox and Lappin 2005.

In general, these alternative proposals depart radically from MS, e.g. by embracing impossible worlds in addition to possible ones; by countenancing not just possible worlds but also partial possible worlds; by rejecting possible worlds altogether; by working in a type theory that abandons one or more of the usual structural rules; by moving from typed to untyped lambda calculus, etc. By contrast, the approach adopted here, described in detail in Pollard 2008, holds that MS was close to the mark, and that a relatively minor repair job eliminates the Granularity problem along with certain other of MS's foundational problems.[2] The repair involves five ideas, of which only the last two are original.

First, we must treat propositions as primitives, not as sets of worlds. In connection with natural-language semantics, this was already advocated by Thomason (1980), though the philosophical roots of the idea can be traced back to Wittgenstein or perhaps even Bolzano.

Second, we must treat worlds not as primitives, but rather as ultrafilters over the boolean structure on propositions. In some form or other, this battle-tested idea is present in Adams (1974), Kripke (1959), Jónsson and Tarski (1951), Carnap (1947), and Stone (1936,1937).

Third, the entailment relation on propositions must not be antisymmetric! I am indebted to Howard Gregory[3] for making me explicitly conscious of the central and nonnegotiable character of this requirement.

Fourth, the boolean structure on propositions, in terms of which worlds are defined as ultrafilters, has to be induced by the entailment relation on propositions. That is, rather than *define* entailment as subset inclusion of sets of worlds, we *axiomatize* entailment to be a boolean preorder on the set of propositions. Then, since the type theory we will work in has Choice[4], it will follow from the (internal) Stone Representation Theorem that this preorder really deserves to be called entailment, in the sense that, for any two propositions p and q, p entails q iff every ulrafilter with p as a member also has q as a member.

And fifth, the type theory must countenance a Separation-like notion of subtyping, along the lines of Lambek and Scott 1986. This will ensure that the worlds, in spite of having been defined as ulrtafilters of propositions, constitute a type of the underlying type theory. That is because the property of of being an ultrafilter is an internally definable property of propositions. More specifically, we can write a formula (boolean term) $u[S]$ (where S is a variable of the

[2] Another approach that is similar in spirit, though not in technical detail, is that of Muskens (2005). Unfortunately Muskens and I separately adopted the term "hyperintensional" for our approaches.

[3] Personal communication to Shalom Lappin and me, 1999.

[4] Actually, the Boolean Prime Ideal Theorem would be enough.

type of sets of propositions) which says of S that it is an ultrafilter, and then use that formula to form a subtype—call it World—of the type Prop $\supset$ Bool. This type will play a role in the semantic theory analogous in crucial respects to that played by the world-type s in MS. However, unlike MS, meanings (including propositions) will *not* be modelled as intensions (functions from worlds to extensions), but rather as the considerably more fine-grained *hyperintensions*.

The purpose of this paper is to show how a semantic theory along these lines sheds light on the understanding of *questions*, the meanings expressed by interrogative sentences. Our questions will turn out to be closely related to the two standard modellings of questions in the linguistic semantics literature, due respectively to Karttunen (1977) (hereafter, K) and Groenendijk and Stokhof (1984) (hereafter, G). The gist of the relationship is as follows. First, the hyperintensional theory of meanings has the property that it contains an 'isomorphic copy of MS', together with a type-parametrized family of functions ext_A that maps each each hyperintension to the corresponding Montagovian intension. In general this function is many-to-one, but it becomes a bijection if we (perversely!) add to our meaning theory an axiom that makes entailment antisymmetric. In the special case of the type Prop of propositions, this function is just the Stone function that maps each member of the preboolean algebra of propositions to a clopen subset of the corresponding Stone space (the one whose members are the ultrafilters of which that proposition is a member.) Under this mapping, each hyperintensional question is mapped to a K-question intension. And the corresponding G-question is in turn obtained from that, by taking the induced equivalence relation on worlds.

Thus, our theory of meanings is a natural generalization of MS (dropping the antisymmetry of entailment), and our theory of questions within it is a natural generalization of the standard MS modellings of questions. The greater generality will enable us to make finer-grained distinctions among questions than the standard theories can make. For example, on K's account (and therefore, on G's as well) these two complementized denote the same questions:

(1) a. whether Paris Hilton is Paris Hilton
    b. whether Britney Spears is Britney Spears

But on our account (hereafter, H), they do not. And on G's account, these two questions are the same:

(2) a. Which students are vegetarians?
    b. Which vegetarians are students?

but on our account they are distinct. The remainder of the paper is organized as follows. In section 2, we describe the version of higher order logic (our counterpart of Montague's IL or Gallin's Ty2) in which the theory of meanings is written. Section 3 sketches the meaning theory itself, while section 4 deals with worlds and extensions at them. Section 5 extends the meaning thory to include questions. And in section 6, we conclude by showing how our theory of questions relates to those of Karttunen and Groenendijk-Stokhof.

2 Higher Order Logic (HOL) with Subtypes

We work in a classical HOL broadly similar to that of Henkin 1950 or Gallin 1975, but augmented with machinery for handling subtyping along the lines of Lambek and Scott 1986. This logic is built on top of positive typed lambda calculus (TLC) (positive in the sense that the underlying type theory is positive intuitionstic propositional logic, so that we have the nullary type constructor T (unit type) and product $\wedge$ as well as the usual exponential $\supset$). For the term constructors, we write $*$ for the constant of type T, $(\_\,,\_)$ for pairing, π and π' for the projections, and $f(a)$ for $\mathsf{eval}(f,a)$. As usual, we have the basic types Ent (entities, corresponding to Montague's e) and Bool (truth values, corresponding to Montague's t[5]). In addition we will have two basic meaning types Prop (propositions, which will serve as the meanings of declarative sentences) and Ind (individual concepts, which will serve as the meanings of names and other expressions that refer to entities). We also have plenty of constants (see (20) below).

To get from positive TLC to HOL, we follow Lambek and Scott and add an equality symbol $=_A : (A \wedge A) \supset \text{Bool}$ for each type A, and then define the usual logical connectives and quantifiers in terms of λ and equality, as shown in the Appendix (13). There we also list some of the logical axioms (or theorems, depending on the choice of axiomatization) of the HOL (14-18).

The crucial feature of our HOL, which will make it possible to internally define a type for worlds (and later, as we will see, also for questions), is the machinery for handling subtypes, adapted from Lambek and Scott 1986. The motivation for this machinery is that the familiar HOLs employed in linguistic semantics provide no way to say that A is a *subtype* of B. In a set-theoretic intepretation I of the logic, this should mean $I(A) \subseteq I(B)$.

(3) **Subtypes (after Lambek and Scott 1986)**
If A is a type and a an A-predicate (i.e. a closed term of type $A \supset \text{Bool}$), then

a. A_a is a type

b. embed_a is a term of type $A_a \supset A$; and

c. Axioms:

i. $\vdash \forall_{y,z \in A_a}[(\mathsf{embed}_a(y) = \mathsf{embed}_a(z)) \supset y = z)]$

ii. $\vdash \forall_{x \in A}[a(x) \leftrightarrow \exists_{y \in A_a} x = \mathsf{embed}_a(y)]$

In a set-theoretic interpretation I of the logic, $I(\mathsf{embed}_a)$ is the function that embeds into $I(A)$ the subset whose characteristic function is $I(a)$.[6]

[5] Or to Lambek and Scott's Ω (subobject classifier). The name Bool will be justified because the HOL of truth-value-typed terms will be classical, and correspondingly the categorical models will be boolean toposes.

[6] More generally, in a topos model, $I(\mathsf{embed}_a)$ is the char of $I(a)$.

3 The Theory of Meanings

We now use our HOL to express a theory about the things that will serve as meanings of (utterances of) natural language expressions. We call this a theory of meanings, not a semantic theory, because we reserve the latter term for a theory that connects linguistic expressions with their meanings. (Some aspects of such a theory are developed in Pollard (2007, submitted, in preparation 1, in preparation 2).

We start by recalling that our basic types are the two extensional types Ent (entities) and Bool (truth values), and the two meaning types Ind (individual concepts) and Prop (propositions). As we'll see, entities are the kinds of things that can be the extensions (at worlds) of individual concepts, and truth values are the kinds of things that can be the extensions of propositions. Note that World is not a basic type! Instead, World will be *defined* as the subtype of Prop $\supset$ Bool consisting precisely of the ultrafilters. This works because ultrafilterhood is an internally definable property (see Appendix, (19).

We now define a certain set of types, the **hyperintensional** types, that will serve as meaning types. This is the set obtained from the basic meaning types Ind and Prop by closing under the TLC type constructors $(\mathrm{T}, \wedge, \supset)$ and subtype formation. Intuitively, the things of these types are the things that have the potential to be meanings. They play a role in our semantic theory analogous to the role played by intensions in MS: they are mathematical models of Fregean senses. (Unavoidably, we will have intensions as well; but we won't use them to model meanings.) Some illustrative examples of constants of various hyperintensional types (corresponding to word meanings) are given in the Appendix (20).

Next, roughly following Montague, we definine a function Ext that assigns to each meaning type the type for the corresponding extensions, i.e. any meaning of type A will have, at each world, an extension of type $\mathrm{Ext}(A)$.

(4) **Extensional Types**

a. $\mathrm{Ext}(\mathrm{Prop}) =_{\mathrm{def}} \mathrm{Bool}$
b. $\mathrm{Ext}(\mathrm{Ind}) =_{\mathrm{def}} \mathrm{Ent}$
c. $\mathrm{Ext}(\mathrm{T}) =_{\mathrm{def}} \mathrm{T}$
d. $\mathrm{Ext}(A \wedge B) =_{\mathrm{def}} \mathrm{Ext}(A) \wedge \mathrm{Ext}(B)$
e. $\mathrm{Ext}(A \supset B) =_{\mathrm{def}} A \supset \mathrm{Ext}(B)$
f. $\mathrm{Ext}(A_a) =_{\mathrm{def}} \mathrm{Ext}(A)$

(5) **Linguistic Consequences**

At any world:

a. Declarative sentences denote truth values.
b. Names denote entities.
c. Dummy pronouns have vacuous reference.

d. The list of complements of a verb denotes the ordered tuple of the denotations the complements.
e. A verb that expresses a function from A's to propositions denotes (the characteristic function of) a set of A's.

We now turn to the axiomatization of *entailment*, the binary relation between propositions that semantics is centrally concerned with. We represent this relation by the constant $\models$ of type (Prop $\wedge$ Prop) $\supset$ Bool. Loosely speaking, we also say that one (utterance of a) declarative sentence entails another if the the proposition it expresses entails the one expressed by the other. Of course every sentence entails itself, and entailment is transitive, so we start with these (nonlogical!) axioms:

(6) **Preorder Axioms for Entailment**
a. $\vdash \forall_p(p \models p)$
b. $\vdash \forall_{p,q,r}(((p \models q) \wedge (q \models r)) \supset (p \models r))$

We use the constant $\equiv$ for mutual entailment:

(7) **Mutual Entailment**
a. $\vdash \forall_{p,q}[(p \equiv q) = (p \models q \wedge q \models p)]$
b. Nothing in our theory will let us prove
$\vdash \forall_{p,q}[(p \equiv q) \supset (p = q)$

That is, entailment is not antisymmetric.

We now introduce the constants Truth : Prop, Falsity : Prop, not' : Prop $\supset$ Prop, and' : (Prop $\wedge$ Prop) $\supset$ Prop, or' : (Prop $\wedge$ Prop) $\supset$ Prop, and implies' : (Prop $\wedge$ Prop) $\supset$ Prop. These will be interpreted as the operations in the preboolean algebraic structure induced by entailment on the set of propositions. As the spelling suggests, some of these operations will also serve as the meanings of the English "logic words" (see Appendix (21) for details). These are axiomatized (Appendix (22)) so that, in an interpretation, the propositions preordered by entailment form a preboolean algebra (roughly, a boolean algebra without antisymmetry, i.e. the usual boolean facts obtain, but with equality replaced by mutual entailment). These axioms essentially say that the usual natural deduction rules of classical propositional logic are valid for natural language argumentation.

4 Worlds and Extensions at Them

So far we have not brought worlds into our theory of meanings. In fact, our view is that worlds are not relevant to meaning: meanings are "out there" (in Frege's Heaven, if you will) and are independent of contingent fact. On the other hand, we must get involved with worlds to deal with reference; since the reference of a linguistic expression is the extension of its meaning, and what that extension is *does* depend on how things are. The most obvious example of this is

that the reference of a declarative sentence is the extension—a truth value—of the proposition it expresses. Fortunately, for us, a world will just be a set of propositions (more specifically, an ultrafilter of the preboolean algebra induced by entailment), and so for that proposition to be true at a given world is simply to be a set-theoretic member of it. This turns MS on its head, since there a proposition is true at a world if the world is a member of *it*.

Now pretheoretically, for p to entail q is supposed to mean that, no matter how things are, if p is true when things are that way, then so is q. So for this to be the case in our framework, it must be the case that p entails q iff every ultrafilter with p as a member also has q as a member. Fortunately for us, this is a theorem of ZFC, not about entailment specifically of course, but about any preboolean algebra (in our setting, the preorder is entailment). In fact this is just a slight generalization (dropping the antisymmetry requirement) of the Boolean Prime Ideal (BPI) Theorem, the key ingredient of Stone's Representation Theorem for boolean algebras. Of course, we are not working in ZFC; but there are topos-theoretic versions of Choice appropriate to the kind of HOL we are working in, and once we add one, we are home free. To summarize, once we add a suitable form of Choice to our axioms, we can prove an internal version of BPI, and this will guarantee that entailment, by virtue of being a boolean preorder, really does behave the way that, pretheoretically, we expect entailment to behave.

We express the connection between hyperintensions, worlds, and extensions using a family of constants ext_A of type $A \supset (\text{World} \supset \text{Ext}(A))$, where A ranges over the hyperintensional types. A full account is given in Pollard 2008, but the three key cases are these:

(8) **Extensions at Worlds**

$$\vdash \forall_{p,w}[\mathsf{ext}_{\text{Prop}}(p)(w) = p@w]$$
$$\vdash \forall_{w,z}[\mathsf{ext}_{A\wedge B}(z)(w) = (\mathsf{ext}_A(\pi(z))(w), \mathsf{ext}_B(\pi'(z))(w))]$$
$$\vdash \forall_{w,f}[\mathsf{ext}_{A\supset B}(f)(w) = \lambda_{x\in A}\mathsf{ext}_B(f(x))(w)]$$

Here $p@w$ abbreviates $\mathsf{emb}(w)(p)$, where emb is the term constructor (from the subtyping schema) that denotes the subtype embedding of World into Prop $\supset$ Bool.[7] Note that $\mathsf{ext}_{\text{Prop}}$ denotes the Stone embedding (the function that maps each element of a preboolean algebra to the set of ultrafilters of which it is a member); that is, ext extends Stone duality from just propositions to all meaning types.

5 Questions in a Hyperintensional Setting

Our modelling of questions is inspired by Kartunnen's (1977) idea that the extension of a question at a world is the set of its true answers there. That is, whatever the type of questions is—for now let's just call it Que—the corresponding extensional type should be sets of propositions, i.e. Ext(Que) should

[7] This is how to say in this kind of HOL that p is a member of w.

be Prop ⊃ Bool. Correspondingly, Que should be Prop ⊃ Prop or some subtype thereof. In short, questions are (certain) propositional operators.

To start with the easy case, how should we analyze yes/no questions, or embedded *whether*-questions? (We will analyze root and embedded interrogatives the same way, for both polar and consituent questions.) On Kartunnen's account, the extension at w of $\mathsf{whether'}(p)$ is the singleton set whose only member is either p or its denial, whichever is true at w. Thus, e.g. *Is Bush crazy*, or equivalently *whether Bush is crazy*, denotes the singleton of the proposition that Bush is crazy in worlds where he is, and the singleton of the proposition that he isn't in worlds where he isn't. We accept this analysis, except that for us the question itself has to be a hyperintension rather than an intension. The key ingredient in our analysis is the meaning $\mathsf{whether'}$ of the interrogative complementizer *whether*, for which we give the following meaning postulate:

(9) **Meaning Postulate for Whether**
$\vdash \mathsf{whether'} = \lambda_{p'}\lambda_p[p\ \mathsf{and'}\ ((p\ \mathsf{equals'}\ p')\ \mathsf{or'}\ (p\ \mathsf{equals'}\ \mathsf{not'}(p')))]$

Here $\mathsf{equals'}$ is the constant whose interpretation is the meaning of the verb *equals*, which refers to true equality (Appendix (23).

(10) **Example (Polar Question)**

a. Is Bush crazy?/whether Bush is crazy

b. Meaning:
$\lambda_p[p\ \mathsf{and'}\ ((p\ \mathsf{equals'}\ \mathsf{crazy'}(\mathsf{Bush'}))\ \mathsf{or'}\ (p\ \mathsf{equals'}\ \mathsf{not'}(\mathsf{crazy'}(\mathsf{Bush'}))))]$

c. Extension at w:
$\lambda_p[p@w \wedge ((p = \mathsf{crazy'}(\mathsf{Bush'})) \vee (p = \mathsf{not'}(\mathsf{crazy'}(\mathsf{Bush'}))))]$

What about constituent questions, such as *which dog barked?* For Karttunen, the extension at w is the set of all propositions true at w of the form $\mathsf{barked'}(x)$ for x an individual such that $\mathsf{dog'}(x)$ is true at w. That is, Kartunnen-extensions for constituent questions contain only *positive* true answers. As discussed in Pollard ms. 2, we propose to include also the *negative* true answers; that is, in a world where Fido barks and Spot doesn't, the extension of the 'plus-or-minus' K- (hereafter, $\mathrm{K}_{\pm}$) question will have as members, possibly *inter alia*, both the proposition that Fido barks *and* the proposition that Spot *doesn't* bark. The essence of this analysis is captured in our semantics for the interrogative determiner *which*:

(11) **The Meaning of *which***
$\vdash \mathsf{which'} = \lambda_P\lambda_Q\lambda_p\mathsf{exists'}(P)(\lambda_x((\mathsf{whether'}(Q(x))(p))))$
Here $x : A$ for A a hyperintensional type; $P, Q : A \supset \mathrm{Prop};$, and $p : \mathrm{Prop}$.

Then our analysis of a simple consituent question looks like this;

(12) **Example (Constituent Question)**

a. Which dogs bark?

b. Meaning: $\mathsf{which}'(\mathsf{dog}')(\mathsf{bark}')$

c. Expanding the abbreviation: $\lambda_p[\mathsf{exists}'(\mathsf{dog}')(\lambda_x(p\ \mathsf{and}'\ ((p\ \mathsf{equals}'\ \mathsf{bark}'(x))\mathsf{or}'\ (p\ \mathsf{equals}'\ \mathsf{not}'(\mathsf{bark}'(x))))))]$

d. Extension at w: $\lambda_p[\exists_x[\mathsf{dog}'(x)@w \wedge (p@w \wedge ((p = \mathsf{bark}'(x)) \vee (p = \mathsf{not}'(\mathsf{bark}'(x)))))]$

Here the constant exists' denotes the hyperintensional counterpart of the usual existential generalized determiner. (Meaning postulates for some representative hyperintensional generalized determiners are given in the Appendix (24).) The upshot is that, at each world w, the reference of the interrogative sentence *which dogs bark* is the range of the function that maps each w-dog to the proposition which correctly answers, for w, the question of whether or not s/he barks.

6 What Have We Gained? What Have We Lost?

We have developed a new modelling of questions within a fine-grained theory of linguistic meanings, a theory which was independently motivated by a number of long-standing problems not specifically connected with questions. And this modelling in turn permits us to distinguish questions from each other in an appropriately more fine grained way.

For example, suppose S_1 and S_2 are English sentences that express mutually entailing but nonidentical propositions. It seems clear that one might wonder whether S_1 without wondering whether S_2. This is problematic for MS, but not for us, because it is easy to show that the function denoted by whether' is injective.

Next, consider a constituent interrogative embedded under a "question-resolving" verb, e.g. *Mary knows (in precise detail) which students in Linguistics 680 are vegetarians*. For this to be true, Mary needs to know, at minimum, for each Linguistics 680 student, whether or not s/he is a vegetarian. But she does not need to know, for each vegetarian, whether or not s/he is taking Linguistics 680. And it would seem that a minimum requirement for predicting this fact must be that the two sentences in (2) express different questions. The G-semantics does not meet this requirement, but the $K_{\pm}$-semantics (both in its original form and in the hyperintensional generalization) does.

To summarize, the hyperintensional semantics of questions generalizes the $K_{\pm}$-semantics, in the sense that for any hyperintentensional question, the corresponding $K_{\pm}$-question (should it be needed) is its Stone dual, directly obtainable by application of ext. Moreover, recalling that for any function f, the induced equivalence relation on the domain is the relation of being mapped to the same value by f, it is not hard to see (Pollard ms. 2) that the corresponding G-question is exactly the equivalence relation on worlds induced by the $K_{\pm}$-question. So it too is easily recovered if needed. Thus, nothing is lost in comparison with the traditional accounts.

References

Adams, R.: Theories of Actuality. Noûs 8, 211–231 (1974)

Barwise, J., Perry, J.: Situations and Attitudes. Bradford Books, Cambridge (1983)

Carnap, R.: Meaning and Necessity. University of Chicago Press, Chicago (1947)

Chierchia, G., Turner, R.: Semantics and property theory. Linguistics and Philosophy 11, 261–302 (1988)

Fox, C., Lappin, S.: Foundations of Intensional Semantics. Blackwell, Oxford (2005)

Gallin, D.: Intensional and Higher Order Modal Logic. North-Holland, Amsterdam (1975)

Groenendijk, J., Stokhof, M.: Studies on the Semantics of Questions and the Pragmatics of Answers. Ph. D. dissertation, University of Amsterdam (1984)

Hamblin, C.: Questions in Montague English. Foundations of Language 10, 41–53 (1973)

Henkin, L.: Completeness in the Theory of Types. Journal of Symbolic Logic 15, 81–91 (1950)

Johnstone, P.: Stone Spaces. Cambridge University Press, Cambridge (1982)

Jónsson, B., Tarski, A.: Boolean algebras with operators, part 1. American Journal of Mathematics 73(4), 891–939 (1951)

Karttunen, L.: Syntax and semantics of questions. Linguistics and Philosophy 1, 3–44 (1977)

Kripke, S.: A completeness theorem in modal logic. Journal of Symbolic Logic 24, 1–14 (1959)

Kripke, S.: Semantic analysis of modal logic I: normal modal propositional calculi. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9, 67–96 (1963)

Lambek, J., Scott, P.: Introduction to Higher-Order Categorical Logic. Cambridge University Press, Cambridge (1986)

Montague, R.: The proper treatment of quantification in ordinary English. In: Thomason, R. (ed.) Formal Philosophy: Selected Papers of Richard Montague, pp. 247–270. Yale University Press, New Haven (1974)

Muskens, R.: Sense and the computation of reference. Linguistics and Philosophy 28(4), 473–504 (2005)

Pollard, C.: Nonlocal dependencies via variable contexts. In: Muskens, R. (ed.) Workshop on New Directions in Type-Theoretic Grammar. ESSLLI 2007, Dublin (2007); Revised and extended version under review for a special issue of Journal of Logic, Language, and Information

Pollard, C.: Hyperintensions. Journal of Logic and Computation 18(2), 257–282 (2008)

Pollard, C.: Covert movement in logical grammar. In: ESSLLI 2008 Workshop on Ludics and Symmetric Calculi, Hamburg, August 2008 (submitted)

Pollard, C.: Ms. 1. Stone dual semantics for natural language. Ohio State University and Universitat Rovira i Virgili. In: Semantics in Paris 2: Semantics beyond Set Theory, CNRS/ENS, Paris, October 2007 (unpublished paper)

Pollard, C.: Ms. 2. What do interrogative sentences refer to? Ohio State University, Universitat Rovira i Virgili, and INRIA-Lorraine. In: Workshop on Reference to Abstract Objects, Universitat Pompeu Fabra, Barcelona, March 2008 (unpublished paper)

Pollard, C.: 1. The *whether* underground. In: Workshop on Ludics, Dialogue, Game Theory, and Questions, Autrans, France, May 2008 (in preparation)

Pollard, C.: 2. Continuations for comparatives. Semantics and Linguistic Theory 19, Columbus, April 2009 (submitted)
Stalnaker, R.: Inquiry. Bradford Books/MIT Press, Cambridge (1984)
Stone, M.: The theory of representation for boolean algebras. Transactions of the American Mathematical Society 40, 37–111 (1936)
Stone, M.: Topological representation of distributive lattices and Brouwerian logics. Časopis pešt. mat. fys. 67, 1–25 (1937)
Thomason, R.: A model theory for propositional attitudes. Linguistics and Philosophy 4, 47–70 (1980)

Appendix

(13) **Classical Connectives and Quantifiers are Definable**
Here ϕ and ψ are metavariables over formulas, x is a variable of type A, and t is a variable of type Bool:

a. $\mathsf{true} =_{\mathrm{def}} * = *$;
b. $\forall_x \phi =_{\mathrm{def}} \lambda_x \phi = \lambda_x \mathsf{true}$;
c. $\mathsf{false} =_{\mathrm{def}} \forall_t t$
d. $\phi \wedge \psi =_{\mathrm{def}} (\phi, \psi) = (\mathsf{true}, \mathsf{true})$;
e. $\phi \supset \psi =_{\mathrm{def}} \phi = (\phi \wedge \psi)$;
f. $\phi \leftrightarrow \psi =_{\mathrm{def}} [(\phi \supset \psi) \wedge (\psi \supset \phi)]$;
g. $\sim \phi =_{\mathrm{def}} \phi \supset \mathsf{false}$;
h. $\phi \vee \psi =_{\mathrm{def}} \sim [(\sim \phi) \wedge (\sim \psi)]$; and
i. $\exists_x \phi =_{\mathrm{def}} \sim \forall_x \sim \phi$.

(14) **Equality is an Equivalence Relation**
In the following, $\alpha, \beta, \gamma, \delta$ are metavariables over terms, and ϕ, ψ are metavariables over formulas,

a. $\vdash \alpha = \alpha$ (reflexivity)
b. $\vdash (\alpha = \beta) \leftrightarrow (\beta = \alpha)$ (symmetry)
c. $\vdash [(\alpha = \beta) \wedge (\beta = \gamma)] \supset (\alpha = \gamma)$ (transitivity)

(15) **Substitution of Equals**

a. $\vdash [(\alpha = \gamma) \wedge (\beta = \delta)] \supset ((\alpha, \beta) = (\gamma, \delta))$
b. $\vdash [(\alpha = \gamma) \wedge (\beta = \delta)] \supset (\alpha(\beta) = \gamma(\delta))$
c. $\vdash (\alpha = \beta) \supset (\pi(\alpha) = \pi(\beta)$
d. $\vdash (\alpha = \beta) \supset (\pi'(\alpha) = \pi'(\beta)$
e. $\vdash (\alpha = \beta) \supset (\lambda_x \alpha = \lambda_x \beta)$

(16) **Axioms for Cartesian Products**

a. $\vdash \alpha = *$ (α a term of type T)
b. $\vdash \pi(\alpha, \beta) = \alpha$
c. $\vdash \pi'(\alpha, \beta) = \beta$
d. $\vdash (\pi(\gamma), \pi'(\gamma)) = \gamma$

(17) **Axioms for Lambda Conversion**

a. $\vdash \lambda_{x \in A}\gamma[x] = \lambda_{y \in A}\gamma[y]$ if y is substitutable for x in γ (Rule α)
b. $\vdash [\lambda_{x \in A}\gamma[x]](a) = \gamma[a]$ if a is substitutable for x in γ (Rule β)
c. $\vdash \lambda_x(\alpha(x)) = \alpha$ if x is not free in α (Rule η)

(18) **Axioms for Boolean Equality**

a. $\vdash \phi = (\phi = \mathsf{true})$
b. If $\vdash \phi$ and $\vdash \phi = \psi$, then $\vdash \psi$
c. $\vdash \phi$ iff $\vdash \phi = \mathsf{true}$
d. $\vdash \forall_{s,t}[(s \leftrightarrow t) \supset (s = t)]$ (Boolean Extensionality)

(19) **Ultrafilterhood is Definable**

a. u is $\lambda_S[a(S) \wedge b(S) \wedge c(S)$ where
   i. $a(S)$ says S is closed under entailment;
   ii. $b(S)$ says s is closed under $\mathsf{and'}$; and
   iii. $c(S)$ says that for each proposition p, exactly one of p and $(\mathsf{not'}p)$ is in S.
b. To be explicit:
   i. $a(S)$ is $\forall_{(p,q)}[(S(p) \wedge p \models q) \supset S(q)]$;
   ii. $b(S)$ is $\forall_{(p,q)}[(S(p) \wedge S(q)) \supset S(p\ \mathsf{and'}\ q)]$; and
   iii. $c(S)$ is $\neg S(\mathsf{falsity'}) \wedge \forall_p(S(p) \vee S(\mathsf{not'}\ p))$.

(20) **Some Constants for Word Meaning**

a. Ind: names, e.g. Fido'
b. T: dummy pronouns. Up to provable equality, the only closed term of this type is $*$.
c. T $\supset$ Prop: intransitive verbs with dummy subjects, e.g. rain'
d. Ind $\supset$ Prop: ordinary intransitive verbs and common nouns, e.g. dog', bark'
e. (Ind $\wedge$ Ind) $\supset$ Prop: transitive verbs, e.g. bite'
f. (Ind $\wedge$ Ind $\wedge$ Ind) $\supset$ Prop: ditransitive verbs, e.g. give'
g. (Ind $\wedge$ Prop) $\supset$ Prop: verbs with declarative sentential complements, e.g. believe'
h. ((Ind $\supset$ Prop) $\wedge$ (Ind $\supset$ Prop)) $\supset$ Prop : (individual) determiners, e.g. every'.

(21) **Boolean Operations on Propositions**

a. Truth : Prop will be interpreted as $\top$, the designated top.

b. $\mathsf{Falsity}$: Prop will be interpreted as $\bot$, the designated bottom.

c. $\mathsf{not'}$: Prop $\supset$ Prop will be interpreted as $\neg$, the designated complement operation.

d. $\mathsf{and'}$: (Prop $\wedge$ Prop) $\supset$ Prop will be interpreted as $\sqcap$, the designated glb operation.

e. $\mathsf{or'}$: (Prop $\wedge$ Prop) $\supset$ Prop will be interpreted as $\sqcup$, the designated lub operation.

f. $\mathsf{implies'}$: (Prop $\wedge$ Prop) $\supset$ Prop will be interpreted as $\Rightarrow$, the designated relative complement operation.

(22) **Preboolean Axioms for Entailment**

a. $\vdash \forall_p(p \models \mathsf{Truth})$

b. $\vdash \forall_p(\mathsf{Falsity} \models p)$

c. $\vdash \forall_{p,q}((p\ \mathsf{and'}\ q) \models p)$

d. $\vdash \forall_{p,q}((p\ \mathsf{and'}\ q) \models q)$

e. $\vdash \forall_{p,q,r}[((p \models q) \wedge (p \models r)) \supset (p \models (q\ \mathsf{and'}\ r))]$

f. $\vdash \forall_{p,q}(p \models (p\ \mathsf{or'}\ q))$

g. $\vdash \forall_{p,q}(q \models (p\ \mathsf{or'}\ q))$

h. $\vdash \forall_{p,q,r}[((p \models r) \wedge (q \models r)) \supset ((p\ \mathsf{or'}\ q) \models r)]$

i. $\vdash \forall_{p,q}[(p\ \mathsf{implies'}\ q)\ \mathsf{and'}\ p) \models q]$

j. $\vdash \forall_{p,q,r}[((r\ \mathsf{and'}\ p) \models q) \supset (r \models (p\ \mathsf{implies'}\ q))]$

k. $\vdash \forall_p((\mathsf{not'}\ p) \equiv (p\ \mathsf{implies'}\ \mathsf{Falsity}))$

l. $\vdash \forall_p[(\mathsf{not'}\ (\mathsf{not'}\ p)) \models p]$

(23) **Three Grades of Equality**

Three families of constants of type $(A \wedge A) \supset$ Prop (for $A \in$ HYPER):

a. equals_A is interpreted as the meaning of the verb *equals*. This has 'true equality' as its extension, as expressed in this meaning postulate:

$\vdash \forall_{w,x,y}[(x\ \mathsf{equals}\ y)@w = (x = y)]$

b. equiv_A is interpreted as the meaning of the term-of-art *is hyperintensionally equivalent to*, subject to the meaning postulate.

$\vdash \forall_{w,x,y}[(x\ \mathsf{equiv}\ y)@w = \forall_{w'}(\mathsf{ext}(x)(w') = \mathsf{ext}(y)(w'))]$

c. coext_A is interpreted as the meaning of the term-of-art *is coextensive with*, subject to the meaning postulate:

$\vdash \forall_{w,x,y}[(x\ \mathsf{coext}\ y)@w = (\mathsf{ext}(x)(w) = \mathsf{ext}(y)(w))]$

(24) **Hyperintensional Generalized Determiners**

a. Constants:
$\vdash \mathsf{every'}_A, \mathsf{exists'}_A, \mathsf{no'}_A : ((A \supset \text{Prop}) \wedge (A \supset \text{Prop})) \supset \text{Prop}$

b. Meaning postulates:

$\vdash \forall_{w,P,Q}[\mathsf{every'}(P,Q)@w = \forall_x(P(x)@w \supset Q(x)@w)]$

$\vdash \forall_{w,P,Q}[\mathsf{exists'}(P,Q)@w = \exists_x(P(x)@w \wedge Q(x)@w)]$

$\vdash \forall_{w,P,Q}[\mathsf{no'}(P,Q)@w = \sim\exists_x(P(x)@w \wedge Q(x)@w)]$

Skolem Theory and Generalized Quantifiers

Livio Robaldo

Department of Computer Science, University of Turin, Italy

Abstract. Several authors proposed to import in Natural Language (NL) the ideas lying behind the well-known Skolem Theorem, defined in standard logic. In these proposals, logical forms aiming at capturing the meaning of NL sentences include referential (functional) terms. Nevertheless, whether referentiality is allowed for non-indefinites NPs, two main problems arise: the need to refer to the maximal sets of entities involved in the predications, and the need to cope with readings where two or more sets of entities are introduced at the same level of scope and have to be evaluated in parallel. Particularly problematic is the representation of sentences featuring nested quantifications. The paper shows how it is possible to incorporate referential terms in the standard Generalized Quantifier account in order to properly deal with nested quantifications.

1 Introduction

The existence of quantifier scope ambiguities in Natural Language (NL) is accepted as a standard by the scientific community. An example is shown in (1)

(1) a. Every man heard a mysterious sound.
b. $\exists_y(mystSound'(y), \forall_x(man'(x), heard'(x, y)))$
c. $\forall_x(man'(x), \exists_y(mystSound'(y), heard'(x, y)))$

(1.a) is ambiguous between two (distributive) readings, represented, in the standard Generalized Quantifier (GQ) account ([16], [1]), as in (1.b-c). $\exists$ and $\forall$ are 2-place GQs, i.e. functions of type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$. In a model M with domain E, $\|\exists\|^M$ and $\|\forall\|^M$ respectively assign, to each $A \subseteq E$, the family of all subsets of E including at least one element of A, and the the family of all subsets of E including all elements of A. (1.b) is true iff a particular sound was heard by all men, while (1.c) is true iff each man heard a (potentially different) sound.

A different approach relies on the assumption that some\all NPs are referential in nature, i.e. they do not denote GQs of type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$ but functional terms of type e, $\langle e, e\rangle$, $\langle e, \langle e, e\rangle\rangle$, etc. This approach is clearly reminiscent of the well-known Skolem theorem, whose application substitutes all existential quantifications by functional terms, which are existentially quantified. For example, the two readings of (1) are represented as (2.a) and (2.b) respectively

(2) a. $\exists_{f_0}\forall_x[mystSound'(f_0) \wedge [man'(x) \rightarrow heard'(x, f_0)]]$
b. $\exists_{f_1}\forall_x[man'(x) \rightarrow (mystSound'(f_1(x)) \wedge heard'(x, f_1(x)))]$

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 286–297, 2008.

In (2.a), f_0 refers to a particular mysterious sound that was heard by every man. Conversely, in (2.b) f_1 is 1-ary function that denotes a functional dependency from men to sounds: given a man x, $f_1(x)$ refers to the sound heard by x.

Several attempts have been made to incorporate Skolem theory in the standard GQ account. [6], [24], [5], and [25] propose the use of Skolem function for properly dealing with the semantics of indefinitives. Conversely, [20], [2], [18], [21], [22], and [19] argue in favour of a view where (almost) all NPs are represented via a referential term. Nevertheless, by allowing referential interpretations of non-indefinites NPs, two main problems arise. First, a standard model theory basically predicts the correct truth conditions only if all involved quantifiers are upwards monotone ($M\uparrow$). Conversely, the truth values may be wrong whenever referential terms feature different monotonicities[1]. Second, this move enables the representation of readings where two or more sets of entities are introduced at the same level of scope; an example, taken from [3], is reported in (3)

(3) Two examiners marked six scripts.

If we allow both NPs to receive wide scope, we get a reading where there is a set of two examiners and a set of six scripts and each of the two examiners marked each of the six scripts. Readings like this are known with the term of "Branching Quantifier" readings. Branching Quantification was introduced by [10] in the context of FOL; afterwards, [11] showed that it can also occurs in NL, and [13], [23], [20], [9], [21], [4] provided further evidence for this claim.

The term "Branching Quantifier readings" is misleading, in that it refers to a formalization that is independent of the meaning it aims to capture. In [21], those reading are handled by referential terms rather than by branching quantifiers. Analogously, [2] represents the reading under exam as in (4)

(4) $(\exists E{:}2(examiner))(\exists S{:}6(script))(\forall e{:}E)(\forall s{:}S)\ marked(e, s)$

Without wanting to touch upon the debate about which terminology should be used, in order to avoid confusion I will refer to the readings where two or more sets of entities are introduced at the same level with the expression "Independent Set" (IS) readings. The existence of IS readings in NL has been, by and large, neglected in the recent literature. Section 2 briefly fleshes out the motivations for considering them as autonomous readings, worth of an explicit representation. Afterwards, I will present an extention of [21] that acts as a uniform device for all those theories where some or all NPs are referentially interpreted.

2 Are Independent Set Readings Really Available in NL?

It may be argued that IS readings can be pragmatically inferred from other available readings. For example, it is easy to see that all models satisfying (4) also satisfy the two linear readings in (5): in (5.a) we may choose the same six scripts for each examiner in E or, in (5.b) the same two examiners for each script in S, obtaining, in both cases, the reading in (4).

[1] See [1] for a survey on possible monotonicities featured by GQs.

(5) a. $(\exists E{:}2(examiner))(\forall e{:}E)(\exists S{:}6(script))(\forall s{:}S)\ marked(e,s)$
b. $(\exists S{:}6(script))(\forall s{:}S)(\exists E{:}2(examiner))(\forall e{:}E)\ marked(e,s)$

I disagree with this supposition and I believe it is logically, linguistically and empirically non-motivated. Concerning its logical inadequacy, it can be shown (see [19]) that the implication does not hold when non-$M\uparrow$ quantifiers are involved. From a linguistic point of view, I believe that not only IS readings do actually occur in NL, but they are rather widespread in everyday human life. Consider:

(6) a. Two students of mine have seen three drug-dealers in front of the school.
b. Most men noticed that few children left the room.
c. The director suggested the failure of two students to most of the teachers.

In (6.a), world knowledge seems to render the reading where the two underlined quantifiers are independent of one another the most salient. Analogously, (6.b) refers to a set of most men and a set of few children such that each man in the former notices that each child in the latter left the room. Finally, (6.c) is saying that the director has two specific students in mind and suggested their respective failure to each individual in a specific set of most teachers.

The existence of IS readings seems also to be empirically confirmed; the reader is addressed to [7] and, in particular, to [8] whose results seem hard to reconcile with approaches that do not take them into account. In his cross-linguistic experiment, Gil shows that IS readings are actually the most highly preferred interpretations by native speakers of Dutch, Hebrew, and Bengali. Furthermore, he provides evidence in favour of the hypothesis that the speakers were performing a semantic rather than a pragmatic task. Similarly, [22] argues that NP referentiality strongly depends on the combinatorics of the grammar. He proposes a logical framework for English only, where all NPs but the universally quantified ones (like *Every man*, *All cats*, etc.) yield a Skolem term. Conversely, in other languages, like Japanese (see [17], [14]), it seems that all NPs, including the universally quantified ones, are referentially interpreted (Steedman, p.c.).

From all these considerations, then, it seems that the fact that IS readings may be special cases of other non-IS ones does not appear to be a sufficient justification for disregarding them. The logic proposed here can be uniformly used to represent the semantics of all natural languages regardless of their scopal preferences, in that IS readings are not seen as special cases of the non-IS ones.

3 The Need of Maximality Conditions

Until we referentially interpret indefinites only, the truth values of the formulae may be easily handled, while it is not so when referentiality is also enabled on other NPs. In particular, we need to introduce special clauses, termed *Maximality Conditions*, when the NPs involves downward monotone (M$\downarrow$) or non-monotone (non-M) quantifiers. Consider (7a-c), respectively involving an M$\uparrow$ (*At least two*), an M$\downarrow$ (*At most two*), an a non-M quantifier (*Exactly two*) quantifier

(7) a. At least two men walk.
b. At most two men walk.
c. Exactly two men walk.

by using Skolem terms, it seems that the meaning of (7.a-c) has to be represented via the formulae in (8.a-c) respectively.

(8) a. $\exists_{f_{m0}}[\,|f_{m0}| \geq 2 \wedge \forall_x[\,x \in f_{m0} \rightarrow (man'(x) \wedge walk'(x))]\,]$
b. $\exists_{f_{m0}}[\,|f_{m0}| \leq 2 \wedge \forall_x[\,x \in f_{m0} \rightarrow (man'(x) \wedge walk'(x))]\,]$
c. $\exists_{f_{m0}}[\,|f_{m0}| = 2 \wedge \forall_x[\,x \in f_{m0} \rightarrow (man'(x) \wedge walk'(x))]\,]$

Nevertheless, only (8.a) correctly yields the truth values of the corresponding sentence. To see why, consider a model in which three men walk. In such a model, (7.a) is true, while (7.b) and (7.c) are false. Conversely, all formulae in (8) evaluate to true, in that all of them allow to choose a Skolem term f_{m0} whose denotation is a plural individual referring to a set of *two* walking men. Therefore, we cannot allow a free choice of the Skolem terms. Instead, they have to denote the *maximal* set of individuals satisfying the predicates, i.e. the maximal set of walking men, in (8). This is achieved by changing (8.b-c) to (9.a-b) respectively

(9) a. $\exists_{f_{m0}}[\,|f_{m0}| \leq 2 \wedge \forall_x[\,x \in f_{m0} \rightarrow (man'(x) \wedge walk'(x))] \wedge$
$\neg\exists_{\boldsymbol{f'_{m0}}}[\,\boldsymbol{f_{m0} \subseteq f'_{m0} \wedge \forall_y[\,y \in f'_{m0} \rightarrow (man'(y) \wedge walk'(y))]]]}$

b. $\exists_{f_{m0}}[\,|f_{m0}| = 2 \wedge \forall_x[\,x \in f_{m0} \rightarrow (man'(x) \wedge walk'(x))] \wedge$
$\neg\exists_{\boldsymbol{f'_{m0}}}[\,\boldsymbol{f_{m0} \subseteq f'_{m0} \wedge \forall_y[\,y \in f'_{m0} \rightarrow (man'(y) \wedge walk'(y))]]]}$

The clauses in boldface are maximality conditions asserting the nonexistence of a superset of f_{m0} whose elements also satisfy man' and $walk'$. Therefore, (9.a-b) are true iff f_{m0} denotes the set of *all* walking men, and this contains exactly two and at most two individuals. In a model with three walking men, (9.a) and (9.b) correctly turn out to be false[2]. [21] proposed a framework grounded on maximality conditions and 1-place GQs, i.e. functions of type $\langle\langle e, t\rangle, t\rangle$. Furthermore, her formulae include FOL predicates whose denotations are relations over the domain, rather than Skolem terms whose denotations are plural individuals. In order to represent IS readings, Sher proposed to impose a maximality condition on the *cartesian product* of the independent sets. (10) shows an example where Two_y and Two_z are independent of one another, and they both depend on Two_x.

(10) Two_x boys have two_y toys and two_z friends who don't like their toys.

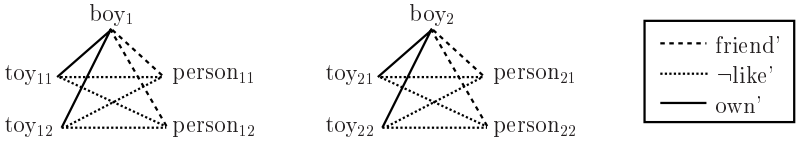
$$\exists P_x P_y P_z[C_x\colon 2!_x(P_x(x)) \wedge C_{yz}\colon \forall_x[P_x(x) \rightarrow 2!_y(P_y(x,y)) \wedge 2!_z(P_z(x,z))] \wedge$$
$$IN\colon \forall_{xyz}[(P_y(x,y) \wedge P_z(x,z)) \rightarrow (boy'(x) \wedge toy\text{-}of'(y,x) \wedge$$
$$friend\text{-}of'(z,x) \wedge \neg\, likes'(z,y))] \wedge$$
$$Max(\langle P_x\rangle, \langle C_{yz}\rangle) \wedge Max(\langle P_y, P_z\rangle, \langle IN\rangle)]$$

[2] Note that, for the sake of uniformity, the maximality condition in boldface can be inserted in the first formula as well: it does not affect the truth values.

The multiple existential quantification has scope on a conjunction of labelled clauses (labels are 'C_x:', 'C_{yz}:' and 'IN:') and a maximality condition in the form $Max(S_1, S_2)$, where S_1 and S_2 are a set of existentially quantified predicates and a set of labels respectively[3]. The labelled clauses require all the individuals in the extention of a predicate to satisfy a certain subformula, while the maximality conditions constraint the cartesian product of the predicates in S_1 to be a *maximal* cartesian product satisfying the clauses whose labels are in S_2. For example, for $Max(\langle P_y, P_z\rangle, \langle IN\rangle)$ the following equivalence holds

$$\begin{aligned}&\mathrm{Max}(\langle P_y, P_z\rangle, IN) \Leftrightarrow\\ &\quad \forall P'_y P'_z[\ \forall_{xyz}[\ (P_y(x,y) \wedge P_z(x,z)) \rightarrow (P'_y(x,y) \wedge P'_z(x,z)) \wedge\\ &\qquad\qquad (P'_y(x,y) \wedge P'_z(x,z)) \rightarrow (boy'(x) \wedge \ldots \wedge \neg\, likes'(z,y))\ ] \rightarrow\\ &\qquad \forall_{xyz}[\ (P'_y(x,y) \wedge P'_z(x,z)) \rightarrow (P_y(x,y) \wedge P_z(x,z))\ ]\ ]\end{aligned}$$

A typical situation in which (10) comes out as true is one where the extension of the 'IN' condition includes two maximal substructures of the form ([21], p.38):



4 Extending Sher's Theory to All NL Determiners

The main peculiarity of Sher's proposal is that it allows to devise formulae with any partial order on any set of quantifiers, regardless of their monotonicities. Nevertheless, it involves 1-place GQs only; this amount to saying that her logical framework allows to represent NL numerals only. In fact, 1-place GQs have a single argument, i.e. they do not distinguish between restriction and body. Conversely, the semantics of NL quantifiers is grounded precisely on this distinction. Therefore, it is necessary to introduce *2-place* GQs, i.e. functions of type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$. Some examples of formulae involving 2-place GQs are

$$\|Few_x(P_1(x), P_2(x))\|^M{=}1 \text{ iff } |(\|P_1(x) \wedge P_2(x)\|^M)| < \varepsilon * |(\|P_1(x)\|)|$$

$$\|n!_x(P_1(x), P_2(x))\|^M{=}1 \text{ iff } |(\|P_1(x) \wedge P_2(x)\|^M)| = n$$

By using 2-place GQs, we can represent the meaning of sentences like (11)

(11) Few$_x$ men inserted a$_y$ coin in every$_z$ coffee machine.

$$\begin{aligned}&\exists P_x P_y P_z[\ \mathrm{C}_x\text{: } Few_x(\text{man'}(x),\ P_x(x)) \wedge \mathrm{C}_y\text{: } \forall_y(\text{CoffeeMach'}(y),\ P_y(y)) \wedge\\ &\qquad \mathrm{C}_z\text{: } \forall_{xy}[(P_x(x) \wedge P_y(y)) \rightarrow \exists_z(\text{coin'}(z),\ P_z(x,\ y))] \wedge\\ &\qquad \mathrm{IN}\text{: } \forall_{xyz}[P_z(x,\ y,\ z) \rightarrow \text{inserted'}(x,\ y,\ z)] \wedge\\ &\qquad \mathrm{Max}(\langle P_x,\ P_y\rangle,\ \langle C_z\rangle) \wedge \mathrm{Max}(\langle P_z\rangle,\ \langle IN\rangle)\ ]\end{aligned}$$

[3] It must be stressed that labels are just a notational convenience used to refer to subformulas in the Max predicate.

The formula in (11) expresses the preferred reading of the sentence, where there is a set including few men and a set including all coffee machines and each man in the former inserted a coin in each machine in the latter (clearly, the coin is different for each pair $\langle Man,\ Coffee\ machine\rangle$). The formula in (11) requires $\|P_z\|^M$ to be the maximal set of triples satisfying $inserted'$ (i.e. it has to coincide with $\|inserted'\|^M$) and $\|P_x\|^M \times \|P_y\|^M$ a maximal cartesian product such that for each pair $\langle m, c\rangle$ in it, $\|\lambda_z.P_z(m, c, z)\|^M$ contains at least one coin. Finally, iff $\|P_x\|^M$ and $\|P_y\|^M$ contain few men and all coffee machines respectively, (11) will be true. Although the introduction of 2-place GQs allows to represent sentences like (11), it does not suffice yet to cope with nested quantifications. Consider:

(12) Exactly two [representatives [of exactly three African countries]] arrive.

The semantics of the IS reading of (12), where two representatives of the same three African countries arrive, is not easy to represent with a Sher-like formula. In fact, at first glance, it seems that the nested quantification in the sentence has to be mirrored in the formula via a logical nested quantification. In other words, it seems that the formula yielding the meaning of (12) has to be[4]:

$$(13)\quad \exists P_x P_y[\ C_x\colon 2!_x(\Psi_x(x),\ P_x(x)) \wedge IN\colon \forall_x[P_x(x) \rightarrow arrive'(x)] \wedge Max(\langle P_x\rangle, \langle IN\rangle)\ ],$$

$$\Psi_x =_{def} \lambda_x.[\ C_y\colon 3!_y(af\_c'(y),\ P_y(y)) \wedge R_y\colon \forall_y[P_y(y) \rightarrow (rep\_of'(x, y))] \wedge Max(\langle P_y\rangle, \langle R_y\rangle)\ ]$$

(13) does not properly capture the truth values of the IS reading of (12) since Ψ_x is true only for those individuals that represent all *and only* the individuals in P_y. On the contrary, in an IS reading, a representative may represent other African countries besides the ones in P_y. Consider a model where it holds that:

- r_1 represents $Egypt$, $Morocco$, $Tunisia$, and $Libya$.
- r_2 represents $Egypt$, $Morocco$, $Tunisia$, and $Algeria$.

and suppose that both r_1 and r_2 arrive. Clearly the IS readings of (13) have to evaluate to true in this model: the model actually includes a set of exactly three African countries ($Egypt$, $Morocco$, and $Tunisia$) such that exactly two of their common representatives arrive. Conversely, (13) evaluates to false. In fact, by taking $\|P_y\|^M$ as the set $\{Egypt, Morocco, Tunisia\}$, the subformula

$$R_y\colon \forall_y[P_y(y) \rightarrow (rep\_of'(x, y))] \wedge Max(\langle P_y\rangle, \langle R_y\rangle)$$

is false both when $x = r_1$ and $x = r_2$, because $\|P_y\|^M$ is not the maximal set of individuals represented either by r_1 or by r_2. Alternatively, we could take $\|P_y\|^M \equiv \{Egypt, Morocco, Tunisia, Libya\}$; however, in such a case, $\|\Psi_x\|^M$ will contain r_1 only, and the formula is again false since $\|\Psi_x\|^M$ has to contain *exactly two* arriving representatives. The same considerations hold by taking $\|P_y\|^M \equiv \{Egypt, Morocco, Tunisia, Algeria\}$: $\|\Psi_x\|^M$ will contain r_2 only.

[4] To increase readability, the restriction of $2!_x$ is externally defined as a predicate Ψ_x.

In order to capture the correct truth conditions of IS readings, we need to avoid nested quantifications in the formulae. Further evidence for this claim is provided by analizing (14) in the logical framework proposed in [22].

(14) a. Exactly two boys visited exactly one museum.
b. $\forall_x[x \in sk_{boy';\lambda_s.\|s\|=2} \rightarrow visited'(x, sk_{museum';\lambda_s.\|s\|=1})]$

In Steedman's framework, the formula corresponding to (14.a), shown in (14.b), contains a Skolem term for each NP; Skolem terms are in the form $sk_{p;c}$ where p is the restriction, e.g. the predicates boy' and $museum'$ in (14.a), and c a cardinality condition. In (14), the two cardinality conditions require the denoted plural individual to include exactly two and exactly one element respectively. The maximality conditions are not explicitly asserted; rather, they are a component of the model theory. In particular, the model theory asserts that a formula $\forall_x[\Phi(x)]$ is true iff, for each individual i in the domain, each Skolem term $sk_{p;c}$ occurring in $\Phi[x\backslash i]$ denotes a *maximal* set of individuals satisfying $p(sk_{p;c})$, $c(sk_{p;c})$, and all atomic predicates in $\Phi[x\backslash i]$ where it is involved. Note that (14.a) does not contain a nested quantification while (14.b) does: the Skolem term $sk_{museum';\lambda_s.\|s\|=1}$ is embedded in the scope of the universal quantifier ranging over the elements of $sk_{boy';\lambda_s.\|s\|\geq 2}$. As in the example discussed above in Sher's framework, the evaluation of the maximality conditions on this embedding prevents the formula to predict the right truth conditions. Consider a model where it holds that

- A boy b_1 visited the museum m_1.
- A boy b_2 visited the museums m_1 and m_2.

In this model, the IS reading of (14.a) is true, while (14.b) is false. In fact, the only reasonable denotation of $sk_{boy';\lambda_s.\|s\|=2}$ is the set $\{b_1, b_2\}$; however, in such a case, $sk_{museum';\lambda_s.\|s\|=1}$ has to refer to a maximal set of museums that independently renders true each of the two subformulae:

- $visited'(b_1, sk_{museum';\lambda_s.\|s\|=1})$
- $visited'(b_2, sk_{museum';\lambda_s.\|s\|=1})$

Nevertheless, by taking $\|sk_{museum';\lambda_s.\|s\|=1}\|^M \equiv \{m_1\}$, the first subformula evaluates to true, while the second to false: $\{m_1\}$ is the set of all museums visited by b_1, but not by b_2. By taking $\|sk_{museum';\lambda_s.\|s\|=1}\|^M \equiv \{m_1, m_2\}$, the opposite holds: $\{m_1, m_2\}$ is the set of all museums visited by b_2, but not by b_1. To solve the problem arising from Nested Quantification, two different solutions are possible: the first one aims at identifying the set of all possibile readings that an NL sentence can receive, and at adding further suitable constraints to the logic so that it will guarantee to assign the correct truth conditions to those readings only. For instance, it can be argued that (13) and (14) never receive a reading where two persons represent the same three African countries and where two boys visited the same single museum. The second solution, which will be

proposed here, aims at defining a logic that works for any set of quantifiers and any partial order on them, regardless of their actual existence in NL.

5 A Uniform Logical Framework for Referential NPs

In this section, I propose an extention of [21] where formulae do not contain embedded subformulae. The key idea is to avoid nestings in the formulae by introducing two relational variables P_x^R and P_x^B for each quantifier Q_x. The first one will identify the set of entities in Q_x's restriction, while the second one the set of entities satisfying Q_x's body. The formulae will assert maximality conditions on both P_x^R and P_x^B. Contrary to Sher's approach, where all predicates are inserted in a single inclusion condition, in the proposal made here several inclusion conditions need to be distinguished. One of them is the main inclusion condition (I will continue to label it by IN); this will maximize the body of one or more quantifiers with respect to the predicates constituting the main assertion. Then, the formulae include another inclusion condition, with label R_x, for each quantifier Q_x; this will be asserted on the predicates in Q_x's restriction. For instance (15) shows the formula corresponding to the IS reading of (12).

$$
\begin{aligned}
(15)\quad & \exists P_x^R P_x^B P_y^R P_y^B [\ 2!_x(P_x^R(x), P_x^B(x)) \wedge 3!_y(P_y^R(y),\ P_y^B(y)) \wedge \\
& IN\colon \forall_x[P_x^B(x) \rightarrow (arrive'(x))] \wedge Max(\langle P_x^B\rangle, \langle IN\rangle) \wedge \\
& R_y\colon \forall_y[P_y^R(y) \rightarrow (af\_c'(y))] \wedge Max(\langle P_y^R\rangle, \langle R_y\rangle) \wedge \\
& R_x\colon \forall_{xy}[(P_x^R(x) \wedge P_y^B(y)) \rightarrow (rep\_of'(x,y))] \wedge Max(\langle P_x^R, P_y^B\rangle, \langle R_x\rangle)]
\end{aligned}
$$

The formula in (15) identifies four sets of entities. P_y^R is the restriction of *Exactly three* and it is required to be the set of all African countries. Analogously, P_x^B, the body of *Exactly two*, is required to be the set of all arrivers. P_x^R, and P_y^B, the restriction of *Exactly two* and the body of *Exactly three* respectively, need to be maximized together, in that the two quantifiers are independent to each other. Therefore, $\|P_x^R\|^M \times \|P_y^B\|^M$ is a maximal cartesian product included in the extension of rep_of' such that each individual in $\|P_x^R\|^M$ represents each individual in $\|P_x^R\|^M$. Finally, iff $\|P_x^R\|^M$ contains exactly two elements of $\|P_x^B\|^M$ and $\|P_x^B\|^M$ contains exactly three elements of $\|P_y^R\|^M$ the formula will be true. In the model above, (15) predicts the right truth values: $\|P_x^R\|^M$ can be taken as the set made up by r_1 and r_2 and $\|P_y^B\|^M$ as the set of all their *commonly* represented African countries, i.e. $\{Egypt, Morocco, Tunisia\}$.

The two standard linear readings of the sentence are allowed in a uniform way; for instance, (16) shows the reading where *Exactly two* outscopes *Exactly three*.

$$
\begin{aligned}
(16)\quad & \exists P_x^R P_x^B P_y^R P_y^B [\ C_x\colon 2!_x(P_x^R(x),\ P_x^B(x)) \wedge \\
& C_y\colon \forall_x[P_x^R(x) \rightarrow 3!_y(P_y^R(x,y),\ P_y^B(x,y))] \wedge \\
& R_x\colon \forall_{xy}[P_y^B(x,y) \rightarrow rep\_of'(x,y)] \wedge R_y\colon \forall_{xy}[P_y^R(x,y) \rightarrow af\_c'(y)] \wedge \\
& IN\colon \forall_x[P_x^B(x) \rightarrow arrive'(x)] \wedge Max(\langle P_y^R\rangle, \langle R_y\rangle) \wedge \\
& Max(\langle P_y^B\rangle, \langle R_x\rangle) \wedge Max(\langle P_x^R\rangle, \langle C_y\rangle) \wedge Max(\langle P_x^B\rangle, \langle IN\rangle)\ ]
\end{aligned}
$$

In this formula, $\|P_y^R\|^M$ is required to be a binary relation such that, for each individual i, $\|\lambda_y.P_y^R(i,y)\|^M$ is the set of all African countries[5]. Analogously, $\|P_y^B\|^M$ is required to be a binary relation such that, for each individual i, $\|\lambda_y.P_y^R(i,y)\|^M$ is the set of all individuals represented by i. Then, the cardinality condition with label C_y constraints $\|P_x^R\|^M$ to be the set of each individual such that all it represents includes exactly three African countries (potentially different from individual to individual). Finally, iff exactly two elements in $\|P_x^R\|^M$ arrive, the formula evaluates to true. Another example is shown in (17)

(17) Few$_x$ boys ate [a$_y$ portion of [every$_z$ course]]

$$\begin{aligned}
&\exists P_x^R P_x^B P_y^R P_y^B P_z^R P_z^B [C_x\colon Few!_x(P_x^R(x),\, P_x^B(x)) \wedge C_z\colon \forall_z(P_z^R(z),\, P_z^B(z)) \wedge \\
&\quad R_x\colon \forall_x[P_x^R(x) \rightarrow boy'(x)] \wedge R_z\colon \forall_z[P_z^R(z) \rightarrow course'(z)] \wedge \\
&\quad C_y\colon \forall_{xz}[(P_x^B(x) \wedge P_z^B(z)) \rightarrow \exists_y(P_y^R(x,z,y),\, P_y^B(x,z,y))] \wedge \\
&\quad R_y\colon \forall_{xzy}[P_y^R(x,z,y) \rightarrow portion_of'(y,z)] \wedge Max(\langle P_y^R\rangle, \langle R_y\rangle) \wedge \\
&\quad IN\colon \forall_{xzy}[P_y^B(x,z,y) \rightarrow ate'(x,y)] \wedge Max(\langle P_y^B\rangle, \langle IN\rangle) \wedge \\
&\quad Max(\langle P_z^R\rangle, \langle R_z\rangle) \wedge Max(\langle P_x^R\rangle, \langle R_x\rangle) \wedge Max(\langle P_x^B, P_z^B\rangle, \langle C_y\rangle)]
\end{aligned}$$

The meaning of (17) is that there is a set of few boys and a set of every course such that each boy in the former ate a portion of each course in the latter. Therefore, in this reading, Few$_x$ and Every$_z$ are independent of one another and they both outscope A$_y$, because the portion varies for each pair $\langle boy, course\rangle$. In the formula, $\|P_y^B\|^M$ is a ternary relation such that for each individual i, $\|\lambda_{xy}.P_y^B(x,i,y)\|^M$ coincides with the extension of ate', while $\|P_y^R\|^M$ is a unary relation such that for each individual i, $\|\lambda_{zy}.P_y^B(i,z,y)\|^M$ coincides with the extension of $portion\_of'$. Then, the cardinality condition with label C_y constraints $\|P_x^B\|^M \times \|P_z^B\|^M$ to be a maximal cartesian product such that each individual in $\|P_x^B\|^M$ ate at least one portion of each individual in $\|P_z^B\|^M$. Finally, iff $\|P_x^B\|^M$ contains few boys and $\|P_z^B\|^M$ every course, the formula will be true. As final complex example, (18) shows a formula involving four quantifiers.

(18) describes a situation where there is a different pair of students and a different program for each teacher, while the topics vary for each pair $\langle student, program\rangle$. The condition with label C_z constraints $\|P_y^R\|^M \times \|P_w^B\|^M$ to be a maximal cartesian product including pairs of pairs such that for each individual i, $\|\lambda_y.P_y^R(i,y)\|^M$ is the set of all students who studied less than half of the topics of all programs in $\|\lambda_w.P_w^B(i,w)\|^M$. Then, the conditions with label C_w and C_y constraints $\|P_x^B\|^M$ to be a maximal set of individuals i such that $\|\lambda_y.P_y^R(i,y)\|^M$

[5] The truth values do not change if $\|P_y^R\|^M$ would be a unary relation made up the set of all African countries, as in (15); however, for the sake of uniformity, two relational variables P_x^R and P_x^B will always have the same arity.

contains at least two students failed by i and $\|\lambda_w.P^B_w(i,w)\|^M$ the single program of i. Finally, iff $\|P^B_x\|^M$ contains all teachers, the formula will be true.

(18) Every$_x$ teacher failed at least two$_y$ students who studied less than half$_z$ of the topics in his$_w$ program.

$$\exists P^R_x P^B_x P^R_y P^B_y P^R_z P^B_z P^R_w P^B_w [\ C_x\colon \forall_x(P^R_x(x), P^B_x(x)) \wedge$$
$$R_x\colon \forall_x[P^R_x(x) \rightarrow teacher'(x)] \wedge Max(\langle P^R_x\rangle, \langle R_x\rangle) \wedge$$
$$C_w\colon \forall_x[P^B_x(x) \rightarrow 1!_w(P^R_w(x,w),\ P^B_w(x,w))] \wedge$$
$$C_y\colon \forall_x[P^B_x(x) \rightarrow Two_y(P^R_y(x,y),\ P^B_y(x,y))] \wedge$$
$$R_w\colon \forall_{xw}[P^R_w(x,w) \rightarrow program\_of'(w,x)] \wedge Max(\langle P^R_w\rangle, \langle R_w\rangle) \wedge$$
$$IN\colon \forall_{xy}[P^B_y(x,y) \rightarrow failed'(x,y)]\ \&\ Max(\langle P^B_y\rangle, \langle IN\rangle) \wedge$$
$$C_z\colon \forall_{xyw}[(P^R_y(x,y) \wedge P^B_w(x,w)) \rightarrow <\tfrac{1}{2}_z(P^R_z(x,w,y,z), P^B_z(x,w,y,z))) \wedge$$
$$R_z\colon \forall_{xwyz}[P^R_z(x,w,y,z) \rightarrow topic'\_of'(z,w)]\ \&\ Max(\langle P^R_z\rangle, \langle R_z\rangle) \wedge$$
$$R_y\colon \forall_{xwyz}[P^B_z(x,w,y,z) \rightarrow (stud'(y) \wedge study'(y,z))]\ \&\ Max(\langle P^B_z\rangle, \langle R_y\rangle) \wedge$$
$$Max(\langle P^B_x\rangle, \langle C_w, C_y\rangle) \wedge Max(\langle P^R_y, P^B_w\rangle, \langle C_z\rangle)\ ]$$

6 Conclusions

In this article, the work proposed in [21] was extended to 2-place GQs. The result is a new logical framework able to represent NL sentences where all NPs are assumed to be referential in nature. The main problem that was solved in order to achieve this result was the occurrence of logical quantifier embeddings, which, if evaluated with respect to a maximality condition and an IS reading, lead to wrong truth conditions. Nested quantifications were forbidden by introducing two referential variables for each quantifier Q, one referring to the set of entities in Q's restriction and one to the set of entities in Q's body.

In this framework, IS readings are considered as natural as the standard linear ones, in that it is possible to set up formulae with any set of quantifiers and any partial order on them, regardless of their actual existence in NL. Whenever the framework is incorporated in an NL theory where some\all NPs receive a referential meaning, the task of determining which partial orders are really available in NL, i.e. which formulae have to be built, is left to the syntax-semantic interface and the disambiguation process. A syntax-semantic interface with respect to a Dependency Syntactic Structure (see [15], [12]) is proposed in [19].

I believe this approach to be more uniform, modular and less risky than proposals aiming at defining logics whose expressivity is restricted to NL Semantics. In fact, it is rather difficult to identify the set of all possible readings that an NL sentence can receive, and there is not a clear agreement yet in the linguistic community about them. These proposals would require to add new constructs\constraints in the logic, in order to extend\restrict its expressivity, once a new non-covered reading is found and considered acceptable or once a covered reading is rejected.

Acknowledgments

First at all, I would like to thank Mark Steedman for all fruitful discussions on our respective works during my five-months visit at UPenn (Philadelphia, USA). This has clearly to be extended to all people of the XTAG group, especially to Aravind Joshi, Maribel Romero, Lucas Champollion, Prashanth Mannem, Tatjana Scheffler, Joan Chen-Main, Joshua Tauberer, and Neville Ryant. Thank you very much for your hospitality, the great vibe, the feedback to my work, and the patience to understand my very complicated formulae.

References

1. Barwise, J., Cooper, R.: Generalized Quantifiers and Natural Language. Linguistics and Philosophy 4(2), 159–219 (1981)
2. Carpenter, B.: Distribution, Collection and Quantification: A Type-Logical Account. Technical Report (1994)
3. Davies, M.: Two examiners marked six scripts. Interpretations of Numerically Quantified Sentences. Linguistics and Philosophy 12(3), 293–323 (1989)
4. Eklund, M., Kolak, D.: Is Hintikka's Logic First Order? Synthese 131(5), 371–388 (2002)
5. Farkas, D.: Dependent Indefinites and Direct Scope. In: Condoravdi, C., Renardel, G. (eds.) Logical Perspectives on Language and Information, pp. 41–72. CSLI Lecture Notes, Stanford (2001)
6. Fodor, J., Sag, I.: Referential and quantificational indefinites. Linguistics and Philosophy 5, 355–398 (1982)
7. Gierasimczuk, N., Szymanik, J.: Hintikka's Thesis Revisited, abstract from Logic Colloquium 2006. The Bulletin of Symbolic Logic 13 (2007)
8. Gil, D.: Quantifier scope, linguistic variation, and natural language semantics. Linguistics and Philosophy 5(4), 421–472 (1982)
9. Hand, M.: A Defense of Branching Quantification. Synthese 95, 419–432 (1993)
10. Henkin, L.: Some remarks on infinitely long formulas. In: Finitistic methods, Proceedings. Symphosium of Foundations Math., pp. 167–183 (1961)
11. Hintikka, J.: Quantifiers vs. Quantification Theory. Dialectica 27, 329–358 (1973)
12. Hudson, R.: English word grammar. Basil Blackwell, Oxford, Cambridge (1990)
13. Kempson, R.M., Cormak, A.: Ambiguity and Quantification. Linguistics and Philosophy 4(2), 259–309 (1981)
14. Miya, S.: Against Optional Scrambling. Linguistic Inquiry 28, 1–25 (1997)
15. Mel'cuk, I.: Dependency syntax: theory and practice. SUNY University Press, New York (1988)
16. Mostowski, A.: On a generalization of quantifiers. Fundamenta Mathematicae 44, 12–36 (1957)
17. Naka, M.: Scrambling and Scope in Japanese. In: Clancy, P. (ed.) Japanese/Korean Linguistics, pp. 283–298. CSLI Lecture Notes, Stanford (1993)
18. Park, J.: A Lexical Theory of Quantification in Ambiguous Query Interpretation. Ph.D. thesis, University of Pennsylvania, USA (1996)
19. Robaldo, L.: Dependency Tree Semantics. Ph.D. thesis, University of Turin, Italy, 1996 (2007)
20. Sher, G.: Ways of branching quantifiers. Linguistics and Philosophy 13, 393–422 (1990)

21. Sher, G.: Partially-ordered (branching) generalized quantifiers: a general definition. The Journal of Philosophical Logic 26, 1–43 (1997)
22. Steedman, M.: The Grammar of Scope. See Surface-Compositional Scope-Alternation Without Existential Quantifiers. Draft 5.2 (September 2007) (forthcoming), ftp://ftp.cogsci.ed.ac.uk/pub/steedman/quantifiers/journal6.pdf
23. Westerståhl, D.: Branching Generalized Quantifiers and Natural Language. In: Gärdenfors, P. (ed.) Generalized Quantifiers: Linguistic and Logical Approaches, pp. 269–298. Reidel, Dordrecht (1987)
24. Winter, Y.: Distributivity and Dependency. Natural Language Semantics 8, 27–69 (2000)
25. Winter, Y.: Functional Quantification. Research on Language and Computation 2, 331–363 (2004)

On a Graph Calculus for Algebras of Relations*

R. de Freitas[1], P.A.S. Veloso[2], S.R.M. Veloso[3], and P. Viana[1]

[1] Institute of Mathematics, UFF: Universidade Federal Fluminense; Niterói, Brazil
[2] Systems and Computer Engin. Program, COPPE, UFRJ: Universidade Federal do Rio de Janeiro; Rio de Janeiro, Brazil
[3] Systems and Computer Engin. Dept., Faculty of Engineering, UERJ: Universidade do Estado do Rio de Janeiro; Rio de Janeiro, Brazil

Abstract. We present a sound and complete logical system for deriving inclusions between graphs from inclusions between graphs, taken as hypotheses. Graphs provide a natural tool for expressing relations and reasoning about them. Here we extend this system to a sound and complete one to cope with proofs from hypotheses. This leads to a system dealing with complementation. Other approaches using pictures for relations use as bases the theory of allegories or rewriting systems. Our formalism is more widely applicable and provides a common denominator of these approaches.

Keywords: Relational language, reasoning from hypotheses, graph calculus, completeness, complementation.

1 Introduction

This paper presents a sound and complete logical system for deriving inclusions between graphs from a set of inclusions between graphs, taken as hypotheses. Traditionally, formulas are written on a single line. S. Curtis and G. Lowe [4] suggest a more visually appealing alternative: using graphs for expressing relations and reasoning about them in a natural way. Although Curtis and Lowe give motivation, present the ideas of using graphs to prove inclusions between relations and illustrate how to apply such a calculus to justify the inference of an inclusion from a set of hypotheses, no proper treatment of these ideas as a logical system seems to have been presented. A proper formulation of the logical system $+\mathsf{RG}$ was presented [6]: a playful logical calculus to derive graphs from graphs, shown to be sound, complete and decidable, for the valid inclusions without the empty relation and complementation.

The main motivation of the present work is to obtain a graph calculus that can be applied to algebras of relations. As complementation can be defined in terms of intersection, union, the universal and the empty relations, here we extend the system $+\mathsf{RG}$ to cope with the empty relation and derivations from hypotheses.

* Research partly sponsored by CNPq (Brazilian National Research Council), FAPERJ (Rio de Janeiro State Research Foundation) and FAPESP (São Paulo State Research Foundation).

W. Hodges and R. de Queiroz (Eds.): WoLLIC 2008, LNAI 5110, pp. 298–312, 2008.

This is more than a simple and natural extension. The proof of completeness for this non-decidable system involves much more elaborated work.

Pictures have been proposed by some authors as a tool to help investigating and applying relational formalisms. Here, we mention two main lines.

The approach based on the *theory of allegories* [1,2,3,4,11] views pictures as arrows in a (unitary pretabular) allegory [8] and uses laws directly associated to the valid allegorical identities for transforming pictures. Results of the theory of allegories are used to show that two pictures can be proved equal by using the laws on pictures iff they represent the same relation.

The approach based on the *rewriting systems* [12,13,14] endows pictures with a relational semantics, which allows them to be interpreted as terms of an algebraic language. A rewriting mechanism for pictures is built as a variant of the algebraic approach to graph rewriting. The way one can use rewriting sequences as proofs leads to a general and flexible tool for the proof of relational algebraic identities.

Our approach [3,6,7] may be called the *logic systematic* approach: pictures are considered as ordinary formulas of a (non-orthodox) logical system and a set of inference rules is provided for deriving pictures from pictures. This approach emphasizes notions of normal form and homomorphism for pictures, which are used to prove the inclusions and equalities.

Each one of these approaches has its own flavor, techniques of investigations and line of results. Nevertheless, they are not completely disjoint, sharing characteristics whose interactions deserve further investigation. The work reported here may also be viewed as a contribution in this direction. We thus provide a new formalism, which is more widely applicable and provides a common denominator of the above three lines of investigation.

The structure of this paper is as follows. In Section 2, we briefly review the relational framework. In Section 3, we introduce our non-negative graph relational framework. In Section 4, we present a derivation system for our graph relational framework and examine some of its aspects: strategy for derivations, soundness and completeness. In Section 5, we indicate how to extend it to handle complementation. In Section 6, we show an application of the graph calculus: proving the main result in [9] as a corollary of our completeness result. In the concluding section, we comment on some on-going work and perspectives. The Appendix complements the main text with figures and proofs of the results.

2 Relational Framework

We now briefly review the relational framework.

Abstractly, relation algebras can be defined by a set of identities specifying the behavior of the Boolean and Peircean operators as follows. The former operators behave as in Boolean algebras. The latter operators behave as in involuted monoid theory. One also adds an identity expressing a geometric aspect of the interaction of Boolean and Peircean operators [10,16].

The non-negative relational language RL^- is the fragment of the relational algebra language with no occurrences of complementation. The RL^- *terms* or

simply *terms*, typically denoted by R, S, T, are generated from the set of relational variables $\textsc{Rvar} = \{r_i : i \in \omega\}$ by applying the relational operators E, O, I, $^\intercal$, $\sqcap$, $\sqcup$, and $\circ$, as usual. We use $\mathcal{T}$ for the set of all (RL^-) terms. The *non-negative relational inclusions* and *equalities* are the expressions of the forms $R \sqsubseteq S$ and $R \equiv S$, respectively.

A *model* is a pair $\mathcal{M} = (M, r_i^{\mathcal{M}})_{i\in\omega}$, where M is a nonempty universe and $r_i^{\mathcal{M}} \subseteq M \times M$ for every $i \in \omega$. The *meaning* $[\![R]\!]_{\mathcal{M}}$ of a term R in a model $\mathcal{M}$ is defined as in the relational case (excluding all references to complementation). Formally, given a model $\mathcal{M}$ with universe M, we interpret the symbols as follows: symbols E, O and I as the relations $M^2 := M \times M$, the empty relation and $\{(a, a) : a \in M\}$, respectively, and symbols $\sqcap$, $\sqcup$, $^\intercal$ and $\circ$ as intersection, union, conversion and composition of relations, respectively. Satisfaction, consequence and validity are defined as usual: $\mathcal{M} \models R \sqsubseteq S$ iff $[\![R]\!]_{\mathcal{M}} \subseteq [\![S]\!]_{\mathcal{M}}$ and $\mathcal{M} \models R \equiv S$ iff $[\![R]\!]_{\mathcal{M}} = [\![S]\!]_{\mathcal{M}}$. Given a set of inclusions Δ, we use $\mathsf{Mod}\,\Delta$ for the class of models satisfying every inclusion in Δ and we define consequence by $\Delta \models R \sqsubseteq S$ iff $\mathsf{Mod}\,\Delta \subseteq \mathsf{Mod}\,\{R \sqsubseteq S\}$. The validities are the consequences of the empty set.

3 Non-negative Graph Relational Framework

We now introduce syntax and semantics of our graph relational framework.

In the non-negative graph relational framework RG^- relations are represented by (directed pseudo multi) graphs having two distinguished nodes and arcs labeled by non-negative relational terms. We consider a fixed set of *nodes* $\textsc{Inod} = \{x_n : n \in \omega\}$, typically denoted by x, y, z, u, v, w.

A *slice* is a structure $S = (N, A, x, y)$, where N is a finite nonempty set of nodes; $A \subseteq N \times \mathcal{T} \times N$ is a set of labeled *arcs* ($\mathcal{T}$ is the set of all terms), with x, y being, not necessarily distinct, distinguished nodes in N. A *non-negative relational graph*, or simply a *graph*, is a finite set of slices. We often identify a single-slice graph with its slice.

We call a slice *basic* iff the labels of its arcs are relational variables or I and a *basic* graph is one whose slices are basic. The RG^- *inclusions* and *equalities* are expressions of the forms $G \sqsubseteq H$ and $G \equiv H$, respectively.

We now present semantics: slices and graphs denote binary relations.

Given a slice $S = (N, A, x, y)$ and a model $\mathcal{M}$ with universe M, an $\mathcal{M}$-*assignment* for S, denoted by $g : S \to \mathcal{M}$, is a function $g : N \to M$ such that $(gu, gv) \in [\![R]\!]_{\mathcal{M}}$ for every arc uRv in A. Now, the *meaning* of a slice S in a model $\mathcal{M}$ is the binary relation $[\![S]\!]_{\mathcal{M}}$ on M defined by $(a, b) \in [\![S]\!]_{\mathcal{M}}$ iff $gx = a$ and $gy = b$, for some assignment $g : S \to \mathcal{M}$. The *meaning* of a graph G in a model $\mathcal{M}$, $[\![G]\!]_{\mathcal{M}}$, is the union of the meanings of its slices.

We now extend some notions for terms to graphs. *Satisfaction* is as expected: $\mathcal{M} \models G \sqsubseteq H$ iff $[\![G]\!]_{\mathcal{M}} \subseteq [\![H]\!]_{\mathcal{M}}$ and $\mathcal{M} \models G \equiv H$ iff $[\![G]\!]_{\mathcal{M}} = [\![H]\!]_{\mathcal{M}}$. Given a set of inclusions Γ, we use $\mathsf{Mod}\,\Gamma$ for the class of models satisfying every inclusion in Γ, and we define *consequence* by $\Gamma \models G \sqsubseteq H$ iff $\mathsf{Mod}\,\Gamma \subseteq \mathsf{Mod}\,\{G \sqsubseteq H\}$. The *validities* are the consequences of the empty set. Also, graphs G and H are *equivalent* iff $G \equiv H$ is valid.

We connect the relational and graphical frameworks by associating single-slice graphs to terms. Given a term R, its graph is $G_R := \{(\{x, y\}, \{xRy\}, x, y)\}$. Note that R and G_R have the same meaning in every model.

4 Non-negative Graph Relational Calculus

We now introduce a derivation system for our graph relational framework.

The deductive apparatus of RG^- is given by a set of graph transforming rules: some rules transform a graph into an equivalent one, and a rule to compare graphs. To state the transformation rules, we use the *node substitution* notation $\frac{u}{v}$ for replacing u by v, which we extend naturally to pairs and triples as well as sets, e.g., for a set A of arcs, we put $A\frac{u}{v} := \{w\frac{u}{v}Rz\frac{u}{v} : wRz \in A\}$.

The *transformation rules* are given in Tables 1, 2 and 3. Table 1 covers the labels of the graphs. Table 2 gives the crucial rule for comparing graphs. Table 3 introduces the new rule for hypotheses. The rules in Tables 1 and 3 can be applied in both directions; each one of these rules is an abbreviation for two rules: downward and upward. The rules in Table 1 allow the *elimination* (downwards) and the *introduction* (upwards) of the operators.

We will explain each two-way rule in the downward direction. Each rule in Table 1 involves the application of the local transformation specified in the rule, leaving the rest of the graph untouched. The meaning of the graph is to remain unchanged. Soundness of the rules follows from these explanations.

Table 1. Elimination/Introduction rules for transforming graphs

$$\mathsf{Unv}\ \frac{G \cup \{(N, A \cup \{u\mathsf{E}v\}, x, y)\}}{G \cup \{(N, A, x, y)\}}$$

$$\mathsf{Idn}\ \frac{G \cup \{(N, A \cup \{u\mathsf{I}v\}, x, y)\}}{G \cup \{(N\frac{u}{v}, A\frac{u}{v}, x\frac{u}{v}, y\frac{u}{v})\}}$$

$$\mathsf{Cnv}\ \frac{G \cup \{(N, A \cup \{uR^{\mathsf{T}}v\}, x, y)\}}{G \cup \{(N, A \cup \{vRu\}, x, y)\}}$$

$$\mathsf{Int}\ \frac{G \cup \{(N, A \cup \{uR \sqcap Sv\}, x, y)\}}{G \cup \{(N, A \cup \{uRv, uSv\}, x, y)\}}$$

$$\mathsf{Cmp}\ \frac{G \cup \{(N, A \cup \{uR \circ Sv\}, x, y)\}}{G \cup \{(N \cup \{w\}, A \cup \{uRw, wSv\}, x, y)\}}\ \text{if } w \notin N$$

$$\mathsf{Uni}\ \frac{G \cup \{(N, A \cup \{uR \sqcup Sv\}, x, y)\}}{G \cup \{(N, A \cup \{uRv\}, x, y), (N, A \cup \{uSv\}, x, y)\}}$$

$$\mathsf{Vd}\ \frac{G \cup \{(N, A \cup \{u\mathsf{O}v\}, x, y)\}}{G}$$

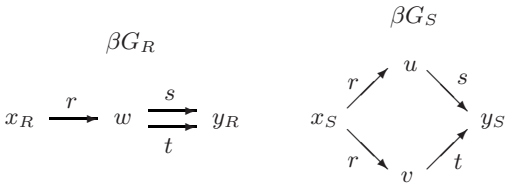


Fig. 1.

Rules in Table 1 are similar to those of +RG [7], with one new rule to deal with the O operator.

Rule Unv allows erasing an arc labeled by E from a slice. Rule Idn allows erasing an arc $u|v$ and a node u, renaming nodes and redirecting arcs accordingly. Rule Cnv allows replacing arcs: $uR^{\intercal}v$ by vRu. Rule Int allows replacing an arc $uR \sqcap Sv$ by arcs uRv and uSv. Rule Cmp allows replacing an arc $uR \circ Sv$ by arcs uRw and wSv, with a new node w. Rule Uni allows replacing a slice T having an arc $uR \sqcup Sv$ by two other slices T_R and T_S, obtained from T by replacing the arc $uR \sqcup Sv$ by new arcs uRv and uSv, respectively. The new rule Vd allows erasing a slice having an arc $u\mathsf{O}v$.

We can reduce each graph G to a basic graph βG equivalent to G by applying the elimination rules in Table 1 (except Idn).

The next example will illustrate the idea of arc preservation and motivate the Graph Cover rule (GrCvr, in Table 2).

Example 1. Consider the terms $R := r \circ (s \sqcap t)$ and $S := (r \circ s) \sqcap (r \circ t)$. To establish the term inclusion $R \sqsubseteq S$, we form the corresponding term graphs G_R and G_S and reduce them to basic forms βG_R and βG_S, shown in Figure 1. Now, consider the node mapping $\theta : x_S \mapsto x_R$, $y_S \mapsto y_R$, $u \mapsto w$, $v \mapsto w$. We see that it preserves arcs, mapping arcs in βG_S to arcs in βG_R. So, we will be able to finish the derivation by applying the Graph Cover rule.

Given slices $S = (N, A, x, y)$ and $S' = (N', A', x', y')$, a *draft homomorphism* $\theta : S' \xrightarrow{d} S$ is a function $\theta : N' \to N$ such that if $u'Rv' \in A'$, then $\theta u' R \theta v' \in A$. A *homomorphism* $\theta : S' \to S$ is a draft homomorphism $\theta : S' \xrightarrow{d} S$ such that $\theta x' = x$ and $\theta y' = y$. Given graphs G and H, we say that H *covers* G, denoted by $G \leftarrow H$, iff for each slice $S \in G$ there exist a slice $T \in H$ and a homomorphism $\theta : T \to S$. The downward rule GrCvr (Table 2), allows one to replace a graph by another one covering it.

Table 2. Graph Cover rule

$\mathsf{GrCvr}\ \dfrac{G}{H}$ if $G \leftarrow H$

Table 3. Hypothesis rule

Hyp_Γ	$\dfrac{G \cup \{S\}}{G \cup \mathsf{glue}_\theta(H,S)}$	if $G' \cup \{S'\} \sqsubseteq H \in \Gamma$ and $\theta : S' \xrightarrow{d} S$

Soundness of GrCvr follows since draft homomorphisms transfer assignments by composition: if $\theta : T \xrightarrow{d} S$ and $g : S \to \mathcal{M}$, then $g.\theta : T \to \mathcal{M}$.

The next example will illustrate the idea of "gluing" slices and motivate the hypothesis rule Hyp_Γ (Table 3).

Example 2. Consider the inclusion $(r \circ s) \sqcap (r \circ t) \sqsubseteq r \circ (s \sqcap t)$. It is well-known that this inclusion is not valid, but it does hold if r is functional, which can be expressed by the inclusion $r^\intercal \circ r \sqsubseteq \mathsf{I}$. With the notation of Example 1, we wish to derive the inclusion $S \sqsubseteq R$ from the hypothesis $T_1 \sqsubseteq T_2$, where $T_1 := r^\intercal \circ r$ and $T_2 := \mathsf{I}$. As before, we reduce the question to graph inclusions: deriving $G_S \sqsubseteq G_R$ from $G_{T_1} \sqsubseteq G_{T_2}$. The corresponding basic forms appear in Figures 1 and 2. Now, there is no homomorphism from βG_R to βG_S: there is no node in βG_S to map w to. But, the lefthand side βG_{T_1} of the hypothesis occurs in βG_S. If we add to this occurrence the righthand side βG_{T_2}, we can then reduce the result to a slice into which we can find a homomorphism from βG_R. Indeed, we have a draft homomorphism $\theta : \beta G_{T_1} \xrightarrow{d} \beta G_S$ given by $\theta x_{T_1} := u$, $\theta y_{T_1} := v$ and $\theta z := y_S$. We now "glue" slice βG_{T_2} onto βG_S using the nodes $\theta x_{T_1} = u$ and $\theta y_{T_1} = v$, obtaining slice T in Figure 2. Now, an application of the downward Idn rule to T yields a slice isomorphic to βG_R.

The rule for hypotheses uses a few concepts which we now introduce. Consider slices $S = (N_S, A_S, x, y)$ and $T = (N_T, A_T, w, z)$, as well as designated nodes $u, v \in N_S$. The result of *gluing* T onto S via (u, v) is the slice defined by $\mathsf{glue}_{(u,v)}(T,S) := (N_S \uplus N_T{}^{\frac{w}{u}\,\frac{z}{v}}, A_S \uplus A_T{}^{\frac{w}{u}\,\frac{z}{v}}, x, y)$. One glues a graph H by gluing its slices: $\mathsf{glue}_{(u,v)}(H,S) := \{\mathsf{glue}_{(u,v)}(T,S) : T \in H\}$. Now, given a slice $S' = (N', A', x', y')$ and a draft homomorphism $\theta : S' \xrightarrow{d} S$, we set $\mathsf{glue}_\theta(H,S) := \mathsf{glue}_{(\theta x', \theta y')}(H,S)$.

Rule Hyp_Γ allows one to glue a graph in a slice of a graph.

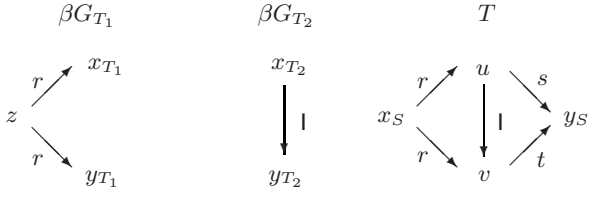


Fig. 2. "Glued" slice: βG_{T_2} into βG_S

The notion of derivation is standard. By a Γ-*derivation* we mean a sequence $G_0, \ldots, G_n$ of graphs such that for each $i \in \{1, \ldots, n\}$, graph G_i is obtained from graph G_{i-1} by application of one of the rules in Tables 1, 2 and 3. We say that a graph inclusion $G \sqsubseteq H$ is *derivable from a set* Γ *of graph inclusions*, denoted by $\Gamma \vdash G \sqsubseteq H$, iff there is a Γ-derivation $G_0, \ldots, G_n$ such that $G_0 = G$ and $G_n = H$. Examples of derivations (without hypotheses) appear in [6] and [7].

Soundness of the rules in Tables 1 and 2 is clear from the explanations given above. Soundness of rule Hyp_Γ in Table 3 follows from Lemma 1, whose proof is given in the Appendix.

Lemma 1. *If* $\mathcal{M} \models G' \cup \{S'\} \sqsubseteq H$*, then* $[\![S]\!]_\mathcal{M} \subseteq [\![\mathsf{glue}_\theta(H, S)]\!]_\mathcal{M}$*, for any slice* S *and draft homomorphism* $\theta : S' \xrightarrow{d} S$*.*

Now, to establish a graph inclusion $G \sqsubseteq H$ from a set Γ of graph inclusions, all in basic form, we can use the strategy of iterating both removal from G of every slice covered by H and application of Hyp_Γ to a slice in G. If and when graph G has no more slices, we can conclude that $\Gamma \models G \sqsubseteq H$. The reason is as follows. Starting with the graphs $G_0^- := G$ and $G_0^+ := \emptyset$, we have graphs such that $\Gamma \vdash G \sqsubseteq G_n^- \cup G_n^+$ with G_n^+ covered by H. Thus, if $G_n^- = \emptyset$, we have $\Gamma \vdash G \sqsubseteq G_n^+$, and, by the graph cover rule GrCvr, $\Gamma \vdash G_n^+ \sqsubseteq H$. Hence, when this construction does terminate, we have $\Gamma \vdash G \sqsubseteq H$. Figure 3 in the Appendix gives a gist of this construction.

The above construction may fail to terminate. A simple example, based on the density-like term inclusion $r \sqsubseteq r \circ r$, will illustrate why. Imagine that the graph inclusion $G_r \sqsubseteq G_{r \circ r}$ is in Γ. The application of $G_r \sqsubseteq \beta G_{r \circ r}$ to a slice will produce a larger slice still having G_r within it, so this may lead to an infinite sequence of increasing slices. See also Figure 4 in the Appendix.

We show (in the Appendix) that, whenever the above construction fails to terminate, we have such an infinite sequence of increasing basic slices, from which we can obtain a canonical counter-model.

To define a canonical model, we need an auxiliary concept. Given a slice $S = (N, A, x, y)$, we define the relation $\sim_S$ on N by $u \sim_S v$ iff there is an arc $u|v \in A$. We use $\sim_S^*$ for the equivalence closure of $\sim_S$. The *canonical model* $\mathcal{S}$ based on a sequence $(S_n)_{n \in \omega}$ of increasing basic slices is defined as follows. Form the set N of all nodes occurring in the sequence and define the (equivalence) relation on N by $u \sim v$ iff $u \sim_{S_n}^* v$, for some n. The universe is the quotient $\tilde{N}$ of N by $\sim$, with natural map $q : N \to \tilde{N}$. Now define the relations by $(\tilde{u}, \tilde{v}) \in r^\mathcal{S}$ iff there exist n such that the arc $u' r v'$ is in S_n, with $u' \sim u$ and $v' \sim v$.

The canonical model has the factorization property: given a basic slice T, a function $g : N_T \to \tilde{N}$ is an assignment $g : T \to \mathcal{S}$ iff there exists a draft homomorphism $\theta : T \xrightarrow{d} S_n$, such that $g = q.\theta$. This leads to the crucial property of the canonical model: $\mathcal{S} \models \Gamma$ and $(\tilde{x}_{S_0}, \tilde{y}_{S_0}) \in [\![H]\!]_\mathcal{S}$ iff H covers some S_n, for any basic graph H. We then have completeness of our graph calculus.

Theorem 1. *If* $\Gamma \models G \sqsubseteq H$*, then* $\Gamma \vdash G \sqsubseteq H$*.*

The following corollary shows that the graph calculus achieves the aim it was originally designed for.

Corollary 1. *Let* $\{R_i \sqsubseteq S_i : i \in I\} \cup \{R \sqsubseteq S\}$ *be a set of relational inclusions. Then* $\{R_i \sqsubseteq S_i : i \in I\} \models R \sqsubseteq S$ *iff* $\{G_{R_i} \sqsubseteq G_{S_i} : i \in I\} \vdash G_R \sqsubseteq G_S$*.*

5 Full-Relational Graph Calculus

We now indicate how to handle complementation, so as to obtain a full-relational graph calculus. We know that the complement of a relation $X \subseteq M \times M$ can be characterized as the unique relation $Y \subseteq M \times M$ such that $X \cap Y \subseteq \emptyset$ and $M \times M \subseteq X \cup Y$. So, whenever we have a complemented term $\overline{R}$ within the label of an arc, we select a new relational variable r_R, replace $\overline{R}$ by r_R and add the two hypotheses $R \sqcap r_R \sqsubseteq \mathsf{O}$ and $\mathsf{E} \sqsubseteq R \sqcup r_R$.

Example 3. Consider the inclusion $(r^{\mathsf{T}} \circ \overline{r \circ s}) \sqcap s \sqsubseteq \mathsf{O}$. We will indicate how one can establish it, using $(r \circ s) \sqcap \overline{r \circ s} \sqsubseteq \mathsf{O}$ and $\mathsf{E} \sqsubseteq (r \circ s) \sqcup \overline{r \circ s}$ as hypotheses. The term graphs $G_{r \circ s}$ and $G_{(r^{\mathsf{T}} \circ \overline{r \circ s}) \sqcap s}$ are respectively equivalent (by Table 1) to $\{T\}$, with $T := (\{w, v, z\}, \{wrv, vsz\}, w, z)$, and $\{S\}$, with $S := (\{x, u, y\}, \{urx, xsy, u\overline{r \circ s}y\}, x, y)$. We have a draft homomorphism $\theta : T \xrightarrow{d} S$, yielding $\mathsf{glue}_\theta(\{T\}, \{S\}) = \{T'\}$, where

$$T' := (N_S \cup N_T, \{urx, xsy, u\overline{r \circ s}y, wrv, vsz, w|u, z|y\}, x, y).$$

Now, by Table 1, graph $\{T'\}$ is equivalent to

$$H := \{(N_S \cup N_T, \{urx, xsy, u\overline{r \circ s}y, ur \circ sy\}, x, y)\},$$

which is equivalent (by $(r \circ s) \sqcap \overline{r \circ s} \sqsubseteq \mathsf{O}$) to the empty graph βG_{O}.

The preceding example shows a rather simple case of how one can handle complementation. More generally, we can use an overall strategy of employing the hypotheses as follows: use $\mathsf{E} \sqsubseteq \overline{R} \sqcup R$ to expand a slice into two, and use $\overline{R} \sqcap R \sqsubseteq \mathsf{O}$ to erase a slice with parallel arcs $\overline{R}$ and R.

Example 4. We indicate how to show that complementation inverts inclusion: $r \sqsubseteq s$ yields $\overline{s} \sqsubseteq \overline{r}$. We start with the single-slice term graph $G_{\overline{s}}$. We expand it (by $\mathsf{E} \sqsubseteq \overline{r} \sqcup r$) to a two-slice graph G_1, which is equivalent (by $\overline{s} \sqcap s \sqsubseteq \mathsf{O}$) to a single-slice graph G_2. Now, we have a draft homomorphism $\theta : G_{\overline{r}} \xrightarrow{d} G_2$, so, the downward graph cover rule GrCvr (in Table 2) yields the desired graph $G_{\overline{r}}$.

Example 5. As part of De Morgan's Theorem K [19] one has the implication *if* $r \circ s \sqsubseteq t$*, then* $r^{\mathsf{T}} \circ \overline{t} \sqsubseteq \overline{s}$. To establish this, it is enough to show

$$\{r \circ s \sqsubseteq t, s \sqcap \overline{s} \sqsubseteq \mathsf{O}, \mathsf{E} \sqsubseteq s \sqcup \overline{s}, t \sqcap \overline{t} \sqsubseteq \mathsf{O}, \mathsf{E} \sqsubseteq t \sqcup \overline{t}\} \models r^{\mathsf{T}} \circ \overline{t} \sqsubseteq \overline{s}, \qquad (1)$$

where $\overline{s}$ and $\overline{t}$ are taken themselves as the new relational variables $r_{\overline{s}}$ and $r_{\overline{t}}$. Figure 5, in the Appendix, presents a graph-calculus derivation establishing (1).

6 Proof Theory for Linear Lattices

M. Haiman [9] presented a set of rules based on graphs to prove the (infinitary) Horn sentences of the form

$$\&_{i\in I}(P_i \leq Q_i) \Rightarrow P \leq Q, \tag{2}$$

where P_i, Q_i, $i \in I$, and P, Q, are lattices terms, which are valid in all linear lattices. We show how to translate (2) in the relational language and prove that the main result of [9] follows from the strong soundness and completeness of the graph calculus.

Let $\textsc{Lvar} = \{p_i : i \in I\}$ be a set of lattices variables. The *lattice terms*, typically denoted by P, Q, are generated from the p_is by applications of the lattice constructing terms operators $\wedge$ and $\vee$. A *lattice inclusion* is an expression of the form $P \leq Q$, where P and Q are lattice terms.

A *lattice* is a structure $\mathcal{L} = (L, \wedge^{\mathcal{L}}, \vee^{\mathcal{L}}, \leq^{\mathcal{L}})$, where $(L, \leq^{\mathcal{L}})$ is a partially ordered set and for every $a, b \in L$, the elements $a \wedge^{\mathcal{L}} b$ and $a \vee^{\mathcal{L}} b$ are, respectively, the inf and sup of a and b according to $\leq^{\mathcal{L}}$. An *assignment* of the lattice variables in a lattice $\mathcal{L}$ is a mapping $v : \textsc{Lvar} \to L$. The *value* of a term under an assignment is the element $[\![P]\!]^v_{\mathcal{L}}$ of L defined by the following rules: $[\![p_i]\!]^v_{\mathcal{L}} := vp_i$, $[\![P \wedge Q]\!]^v_{\mathcal{L}} := [\![P]\!]^v_{\mathcal{L}} \wedge^{\mathcal{L}} [\![Q]\!]^v_{\mathcal{L}}$ and $[\![P \vee Q]\!]^v_{\mathcal{L}} := [\![R_P]\!]^v_{\mathcal{L}} \vee^{\mathcal{L}} [\![R_Q]\!]^v_{\mathcal{L}}$. Let $\{P_j \leq Q_j : j \in J\} \cup \{P \leq Q\}$ be a set of lattice inclusions and K be a class of lattices. We say that $P \leq Q$ is a *consequence of* $\{P_j \leq Q_j : j \in J\}$ *in* K when for every lattice $\mathcal{L} \in \mathsf{K}$ and assignment v of the lattice variables in $\mathcal{L}$, the assumption that $[\![P_j]\!]^v_{\mathcal{L}} \leq^{\mathcal{L}} [\![Q_j]\!]^v_{\mathcal{L}}$, for every $j \in J$, implies $[\![P]\!]^v_{\mathcal{L}} \leq^{\mathcal{L}} [\![Q]\!]^v_{\mathcal{L}}$.

It is well known that given the set M, the structure $\mathcal{E} = (\mathsf{Eq}M, \wedge^{\mathcal{E}}, \vee^{\mathcal{E}}, \leq^{\mathcal{E}})$ is a lattice when $\mathsf{Eq}M$ is the set of all equivalence relations on M, $\leq^{\mathcal{E}}$ and $\wedge^{\mathcal{E}}$ are, respectively, the set inclusion and set intersection, and $X \vee^{\mathcal{E}} Y$ is defined as

$$X \vee^{\mathcal{E}} Y := (X \mid Y) \cup (X \mid Y \mid X) \cup (X \mid Y \mid X \mid Y) \cup \ldots$$

An important class of lattices of equivalence relations is defined by requiring that $X \vee^{\mathcal{E}} Y := X \mid Y$. Since this condition is equivalent to requiring commutativity of the composition $\mid$, we say that $\mathcal{L}$ is a *lattice of commuting equivalence relations*. Also, call a lattice *linear* when it is isomorphic to a lattice of commuting equivalence relations.

In [9] M. Haiman presented a set of rules based on graphs to prove the (infinitary) Horn sentences of the form $\&_{j\in J}(P_j \leq Q_j) \Rightarrow P \leq Q$, where P_j, Q_j, $j \in J$, and P, Q, are lattices terms, which are valid in all linear lattices. In the terminology above this means that the system proposed in [9] proves the sentence $\&_{j\in J}(P_j \leq Q_j) \Rightarrow P \leq Q$ iff $P \leq Q$ is a consequence of the set $\{P_j \leq Q_j : j \in J\}$ in the class of all linear lattices. Now, we prove that the main result of [9] follows from the strong completeness of the graph calculus.

We assume that the set of lattice variables $\textsc{Lvar} := \{p_i : i \in I\}$ is in one-to-one correspondence with the set of relation symbols $\textsc{Rsym} := \{r_i : i \in I\}$ and that these sets are disjoint and fixed throughout.

First, we define the translation mapping from lattice terms P to relational terms R_P, by $R_{p_i} := r_i$ $(i \in I)$, $R_{P \wedge Q} := R_P \sqcap R_Q$, $R_{P \vee Q} := R_P \circ R_Q$. According to the above, we say that a $+\mathsf{RG}$ inclusion $R \sqsubseteq S$ is *linear lattice like* iff the only operators occurring in $R \sqsubseteq S$ are $\sqcap$ and $\circ$. Second, we denote by LLat the set of linear lattices like inclusions containing as elements all the inclusions of the form $\mathsf{I} \sqsubseteq r$, $r^{\mathsf{T}} \sqsubseteq r$, $r \circ r \sqsubseteq r$, and $r \circ s \sqsubseteq s \circ r$, where r, s are relation symbols.

Theorem 2. *Let $\{P_j \leq Q_j : j \in J\} \cup \{P \leq Q\}$ be a set of lattice inclusions. Then the following are equivalent:*
(a) $\&_{j \in J}\ P_j \leq Q_j \Rightarrow P \leq Q$ *is valid in the class of all linear lattices;*
(b) $\{R_{P_j} \sqsubseteq R_{Q_j} : j \in J\} \cup \mathsf{LLat} \vdash R_P \sqsubseteq R_Q$ *in* $+\mathsf{RG}$.

7 Conclusion

We have presented a sound and complete logical system for deriving inclusions between graphs from inclusions between graphs. We can extend our system to a full-relational graph calculus with complementation, as indicated in Section 4. Our system uses linear derivations, in contrast with other systems for relations [15].

The monotonicity rule was suggested to handle simple hypotheses [4]. Our Rule Hyp_Γ was inspired by it, but it involves more elaborated formulation. The proof of our main result, the completeness theorem, gives a strategy for deriving a graph inclusion from a set of graph inclusions.

The importance of the the main result of [9] and, consequently, of our Theorem 2, is discussed in [5,17,20,18], where the proof theory for linear lattices is extended and applied in the study of expressions involving joins and meets of subspaces of vectors spaces. The main differences between the graph calculi presented in these works and ours is that they do not use neither the notion of normal form for graphs nor that of homomorphism between graphs. As future work we intend to investigate the exact relationship between their calculi and ours.

References

1. Brown, C., Hutton, G.: Categories, allegories and circuit design. In: Proc. Ninth Annual IEEE Symp. on Logic in Computer Science, pp. 372–381. IEEE Computer Society Press, New York (1994)
2. Brown, C., Jeffrey, A.: Allegories of circuits. In: Proc. Logical Foundations of Computer Science, St. Petersburg, pp. 56–68 (1994)
3. Curtis, S., Lowe, G.: A graphical calculus. In: Möller, B. (ed.) MPC 1995. LNCS, vol. 947, pp. 214–231. Springer, Heidelberg (1995)
4. Curtis, S., Lowe, G.: Proofs with graphs. Sci. Comput. Programming 26, 197–216 (1996)
5. Finberg, D., Mainetti, M., Rota, G.-C.: The logic of commuting equivalence relations. In: Logic and Algebra. Lecture Notes in Pure and Applied Mathematics, vol. 180, pp. 63–99. Dekker, New York (1996)

6. de Freitas, R., Veloso, P.A.S., Veloso, S.R.M., Viana, P.: Reasoning with graphs. ENTCS 165, 201–212 (2006)
7. de Freitas, R., Veloso, P.A.S., Veloso, S.R.M., Viana, P.: On positive relational calculi. Logic J. IGPL 15, 577–601 (2007)
8. Freyd, P.J., Scedrov, A.: Categories, Allegories. North-Holland, Amsterdam (1990)
9. Haiman, M.: Proof theory for linear lattices. Adv. in Math. 58, 209–242 (1985)
10. Hirsch, R., Hodkinson, I.: Relation Algebras by Games. Elsevier, Amsterdam (2002)
11. Hutton, G.: A relational derivation of a functional program. In: Lecture Notes of the STOP Summer School on Constructive Algorithms (1992)
12. Kahl, W.: Algebraic graph derivations for graphical calculi. In: D'Amore, F., Marchetti-Spaccamela, A., Franciosa, P.G. (eds.) WG 1996. LNCS, vol. 1197, pp. 224–238. Springer, Heidelberg (1997)
13. Kahl, W.: Relational treatment of term graphs with bound variables. Logic J. IGPL 6, 259–303 (1998)
14. Kahl, W.: Relational matching for graphical calculi of relations. Inform. Sciences 119, 253–273 (1999)
15. Maddux, R.D.: A sequent calculus for relation algebras. Annals Pure Applied Log. 25, 73–101 (1983)
16. Maddux, R.D.: Relation Algebras. Elsevier, Amsterdam (2006)
17. Mainetti, M., Yan, C.H.: Arguesian identities in linear lattices. Adv. in Math. 44, 50–93 (1999)
18. Mainetti, M., Yan, C.H.: Geometric identities in lattice theory. J. Combin. Theory Ser. A 91, 411–450 (2000)
19. De Morgan, A.: On the syllogism: IV, and on the logic of relations. Cambridge Phil. Soc. Trans. 10, 331–358 (1864)
20. Yan, C.H.: Arguesian identities in the congruence variety of Abelian groups. Adv. in Math. 150, 36–79 (2000)

Appendix

Proof of Lemma 1. Given $(a,b) \in [\![S]\!]_{\mathcal{M}}$, there is an assignment $g : S \to \mathcal{M}$ such that $gx = a$ and $gy = b$. This induces the assignment $g.\theta : S' \to \mathcal{M}$. So, with $c := g.\theta x_{S'}$ and $d := g.\theta y_{S'}$, we have $(c,d) \in [\![S']\!]_{\mathcal{M}} \subseteq [\![G' \cup S']\!]_{\mathcal{M}}$, whence $(c,d) \in [\![H]\!]_{\mathcal{M}}$ (as $\mathcal{M} \models G' \cup \{S'\} \sqsubseteq H$). Thus, $(c,d) \in [\![T]\!]_{\mathcal{M}}$, for some slice T of H, so, there is an assignment $g' : T \to \mathcal{M}$ such that $(c,d) = (g'x_T, g'y_{S'})$. Now, define $g'' : N_S \uplus N_T \to M$ naturally to agree with g on N_S and with g' on N_T. This gives an assignment $g'' : \mathsf{glue}_\theta(T,S) \to \mathcal{M}$, with $(g''x_S, g''y_S) = (a,b)$. Therefore, $(a,b) \in [\![\mathsf{glue}_\theta(T,S)]\!]_{\mathcal{M}} \subseteq [\![\mathsf{glue}_\theta(H,S)]\!]_{\mathcal{M}}$. □

Proof of Theorem 1. The proof is based on the construction of a tree with nodes labeled by basic graphs, based on applications of derived rules iHyp_Γ and GrRCvr (Tables 4 and 5). These derived rules uses a few concepts which we now introduce.

Given a slice $S' = (N', A', x', y')$, an *i-draft homomorphism* $\theta : S' \xrightarrow{i} S$ is a function $\theta : N' \to N$ such that (1) if $u' \mathsf{I} v' \in A'$, then $\theta u' \sim_S^* \theta v'$, and (2) if $u' R v' \in A'$, then there exist an arc $uRv \in A$ such that $\theta u' \sim_S^* u$ and $\theta v' \sim_S^* v$. A

Table 4. Graph Relaxed Cover rule

$$\mathsf{GrRCvr}\ \frac{G}{H} \quad \text{if } G \overset{r}{\leftarrow} H$$

Table 5. i-Hypothesis rule

$$\mathsf{iHyp}_{\Gamma}\ \frac{G \cup \{S\}}{G \cup \mathsf{iglue}_{\theta}(H,S)} \quad \text{if } G' \cup \{S'\} \sqsubseteq H \in \Gamma \text{ and } \theta : S' \overset{d}{\to} S$$

relaxed homomorphism $\theta : S' \overset{r}{\to} S$ is an i-draft homomorphism $\theta : S' \overset{i}{\to} S$ such that $\theta x' {\sim}^*_S x$ and $\theta y' {\sim}^*_S y$.

Given graphs G and H, we say that H *r-covers* G, denoted by $G \overset{r}{\leftarrow} H$, iff for each slice $S \in G$ there exist a slice $T \in H$ and a relaxed homomorphism $\theta : T \overset{r}{\to} S$.

Consider slices $S = (N_S, A_S, x, y)$ and $T = (N_T, A_T, w, z)$, as well as designated nodes $u, v \in N_S$. The result of *i-gluing* T onto S via (u, v) is the slice defined by $\mathsf{iglue}_{(u,v)}(T,S) := (N_S \uplus N_T, A_S \uplus A_T \cup \{u\mathsf{l}x_T, v\mathsf{l}y_T\}, x_S, y_S)$. One i-glues a graph H by i-gluing its slices: $\mathsf{iglue}_{(u,v)}(H,S) := \{\mathsf{iglue}_{(u,v)}(T,S) : T \in H\}$. Given a slice $S' = (N', A', x', y')$ and a i-draft homomorphism $\theta : S' \overset{i}{\to} S$, we set $\mathsf{iglue}_{\theta}(H,S) := \mathsf{iglue}_{(\theta x', \theta y')}(H,S)$.

A set of graph inclusions Γ is *basic* iff every inclusion $G \sqsubseteq H \in \Gamma$ is such that G and H are basic. Let us consider, wlog, that G, H, and Γ are basic and that each inclusion in Γ has a single-slice graph at the left-hand side.

Given slice S, define

- $H(S) := \{(\theta, T) : T \in H \text{ and } \theta : T \overset{r}{\to} S\}$ and
- $\Gamma(S) := \{(\theta, G' \cup \{S'\} \sqsubseteq H') : G' \cup \{S'\} \sqsubseteq H' \in \Gamma \text{ and } \theta : S' \overset{d}{\to} S\}$.

We say that a slice S is *red* iff $H(S) \neq \emptyset$. We say that a slice S is *yellow* iff $H(S) = \emptyset$ and $\Gamma(S) = \emptyset$. We say that a slice S is *green* iff $H(S) = \emptyset$ and $\Gamma(S) \neq \emptyset$. We say that a graph G is *red* iff every slice in G is red.

Consider an enumeration $S_1 \sqsubseteq H_1, S_2 \sqsubseteq H_2, \ldots, S_n \sqsubseteq H_n, \ldots$ of graph inclusions in Γ such that each graph inclusion appears infinitely many times in the enumeration.

We shall build a tree whose nodes are labeled by graphs beginning with its root, labeled by G, and expanding the tree considering the slices of each leaf, based on the following idea: we will have no gain in expanding the tree from red slices, from yellow slices it is impossible to expand the tree (since expansion is made based on the hypotheses in Γ), green slices are the ones used to expand the tree.

We say that a slice $S' = (N', A', x', y')$ is an *extension* of a slice $S = (N, A, x, y)$ iff $N \subseteq N'$, $A \subseteq A'$, $x = x'$, and $y = y'$. Then every slice in $\mathsf{glue}_{\theta}(H,S)$

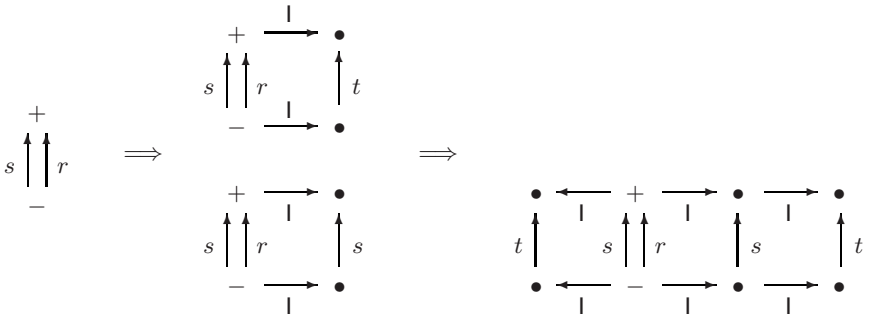


Fig. 3. Construction for $\{r \sqsubseteq t \sqcup s, s \sqsubseteq t\} \vdash r \sqcap s \sqsubseteq t$

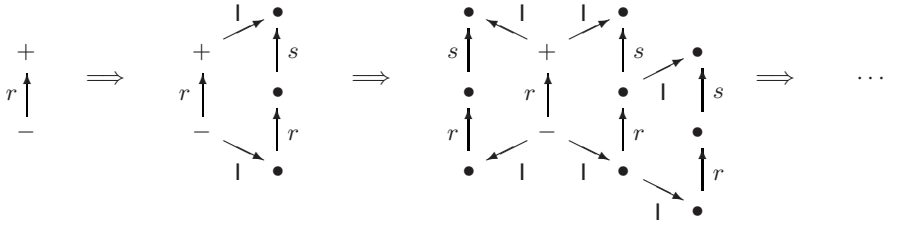


Fig. 4. Construction for $\{r \sqsubseteq r \circ s\} \not\models r \sqsubseteq s$

is an extension of S and, if $\theta : S' \xrightarrow{d} S$, then θ can be considered as a draft homomorphism from S' to every extension of S. If G is a graph whose slices are extensions of S and $\theta : S' \xrightarrow{d} S$, define $\mathsf{glue}_\theta(H, G) := \{\mathsf{glue}_\theta(H, S'') : S'' \in G\}$. Define also $\mathsf{glue}_{\theta_1 \cdots \theta_n}(H, G) := \mathsf{glue}_{\theta_1}(H, \mathsf{glue}_{\theta_2 \cdots \theta_n}(H, S))$.

Define $\mathcal{T}_0$ to be the tree whose only node is $G_0 = G$. Given $\mathcal{T}_k$, define $\mathcal{T}_{k+1}$ as follows. For each leaf G' of $\mathcal{T}_k$, for each $S \in G'$, if S is green, then add $\mathsf{glue}_{\theta_1 \cdots \theta_n}(H_{k+1}, S)$ as a son of G' and label its arc with S, where $\{\theta_1, \ldots, \theta_n\}$ is the set of all draft homomorphisms from S_{k+1} to S. If there is no draft homomorphism from S_{k+1} to S, add S itself as a son of G'. When a leaf in $\mathcal{T}_k$ does not have green slices, it gains no sons and remains a leaf in $\mathcal{T}_{k+1}$. Define tree $\mathcal{T} = \bigcup_{i \in \mathbb{N}} \mathcal{T}_i$.

Given a node G in $\mathcal{T}$, we say that G is red* iff G is red or for all $S \in G$, slice S is red or G has a son G' labeled S such that G' is red*. If G_0 is red*, then one can obtain a derivation of $G \sqsubseteq H$ from Γ, as follows. We start with a sequence of graphs $(G'_1, \ldots, G'_n)$, obtained as follows. Graph G'_1 is G_0 and G'_{k+1} is the result of a series of applications of Hyp_Γ on G'_k, one for each draft homomorphism from S_{k+1} to each $S \in G'_k$.

It is immediate to transform $(G'_1, \ldots, G'_n)$ into a derivation of $G \sqsubseteq H$ from Γ. Include H after G'_n, since H can be obtained from G'_n by an application of GrCvr (because every slice in G'_n is red). If G, H, or Γ are not basic, include steps of applications of the rules in Table 1 eliminating operators, except I, and introducing operators, to obtain H from the result of application rule of GrCvr on graph G'_n.

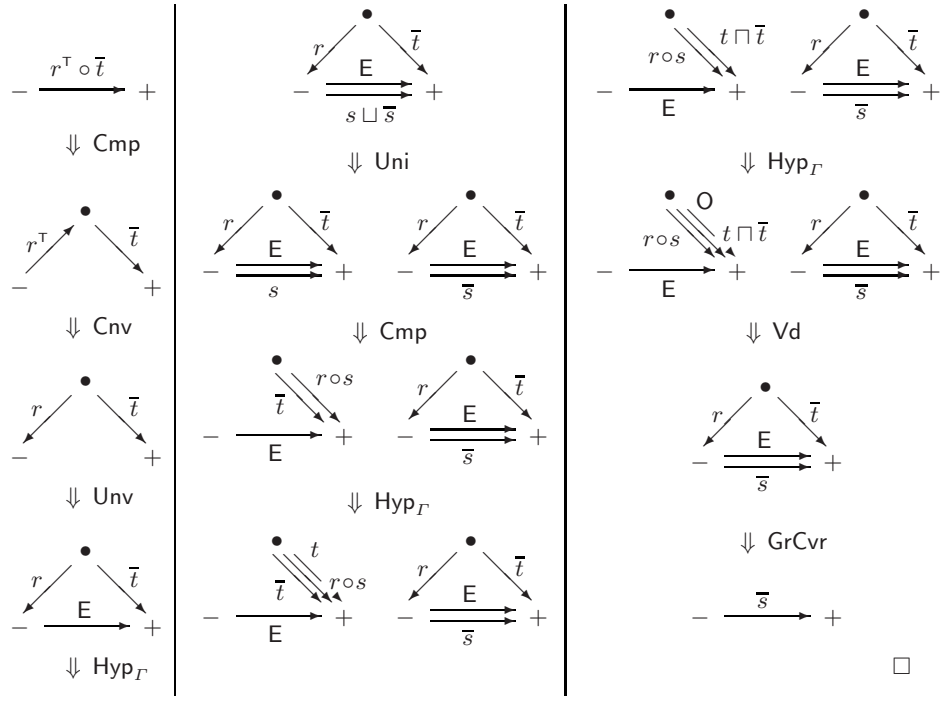


Fig. 5. Graph derivation for De Morgan's Theorem K

If G_0 is not red*, then one can construct a model $\mathcal{M}$ of Γ that is not a model of $G \sqsubseteq H$. If tree $\mathcal{T}$ has a graph with a yellow slice $S = (N, A, x, y)$, then define $\mathcal{M} = (M, r_i^{\mathcal{M}})_{i\in\mathbb{N}}$ with $M = \tilde{N}$ and $r_i^{\mathcal{M}} = \{(\tilde{u}, \tilde{v}) : uRv \in A\}$, where $\sim$ is $\sim_S^*$. Otherwise, there is an infinite path of graphs having green slices in tree $\mathcal{T}$. From this path we can obtain a chain $(S_n)_{n\in\omega}$ of green slices. In this case, take $\mathcal{M}$ to be the canonical model on $(S_n)_{n\in\omega}$.

In any case (tree $\mathcal{T}$ having a yellow slice or a chain of green slices), the constructed model $\mathcal{M}$ is such that $\mathcal{M} \models \Gamma$ and $\mathcal{M} \not\models G \sqsubseteq H$. In fact, $\mathcal{M} \models \Gamma$ since in the construction of $\mathcal{T}$ all possible applications of iHyp_Γ, related to each inclusion of Γ, are effectively applied on every slice in chain $(S_n)_{n\in\omega}$. Also, $(\tilde{x}_{S_0}, \tilde{y}_{S_0}) \in [\![S_0]\!]_{\mathcal{M}} \subseteq [\![G]\!]_{\mathcal{M}}$ but there is no slice $T \in H$ such that $(\tilde{x}_{S_0}, \tilde{y}_{S_0}) \in [\![T]\!]_{\mathcal{M}}$. Otherwise, witness assignment g would be a relaxed homomorphism from T to S_0 and S_0 would be red. □

Proof of Theorem 2. (a)⇒(b) Suppose, for a contradiction, that $\&_{j\in J}\, P_j \leq Q_j \Rightarrow P \leq Q$ is valid in all linear lattices but $\{R_{P_j} \sqsubseteq S_{Q_j} : j \in J\} \cup \mathsf{LLat} \not\vdash R_P \sqsubseteq S_Q$ in $+\mathsf{RG}$. Then, by the strong completeness of the graph calculus $\{R_{P_j} \sqsubseteq S_{Q_j} : j \in J\} \cup \mathsf{LLat} \not\models R_P \sqsubseteq S_Q$. So, there is a model $\mathcal{M}$ such that $\mathcal{M} \models \{R_{P_j} \sqsubseteq S_{Q_j} : j \in J\}$, $\mathcal{M} \models \mathsf{LLat}$, but $\mathcal{M} \not\models R_P \sqsubseteq S_Q$. Now, take $\mathcal{L} = (\{r_i^{\mathcal{M}} : i \in I\}, \cap, |, \subseteq)$ and $v : \textsc{Var} \to \{r_i^{\mathcal{M}} : i \in I\}$ such that $vp_i := r_i^{\mathcal{M}}$. We have that $\mathcal{L}$ is a lattice of commuting equivalence relations and that $[\![P]\!]_{\mathcal{L}}^v = [\![R_P]\!]_{\mathcal{M}}$ for every lattice term P. So, for $\mathcal{L}$ and v we have $[\![P_j]\!]_{\mathcal{L}}^v \subseteq [\![Q_j]\!]_{\mathcal{L}}^v$, for

every $j \in J$, but $[\![P]\!]^v_{\mathcal{L}} \not\subseteq [\![Q]\!]^v_{\mathcal{L}}$, a contradiction with $P \leq Q$ is a consequence of $\{P_j \leq Q_j : j \in J\}$ in the class of all linear lattices.

(b)⇒(a) Suppose, for a contradiction, that $\{R_{P_j} \sqsubseteq S_{Q_j} : j \in J\} \cup \mathsf{LLat} \vdash R_P \sqsubseteq S_Q$ in $+\mathsf{RG}$ but $\&_{j\in J}\ P_j \subseteq Q_j \Rightarrow P \subseteq Q$ is not valid in the class of all linear lattices. So, there is a linear lattice $\mathcal{L} = (L, \wedge^{\mathcal{L}}, \vee^{\mathcal{L}}, \leq^{\mathcal{L}})$ and an assignment $v : \textsc{Lvar} \to L$ such that $[\![P_j]\!]^v_{\mathcal{L}} \leq^{\mathcal{L}} [\![Q_j]\!]^v_{\mathcal{L}}$, for every $j \in J$, but $[\![P]\!]^v_{\mathcal{L}} \not\leq^{\mathcal{L}} [\![Q]\!]^v_{\mathcal{L}}$. Now, since $\mathcal{L}$ is a linear lattice, there are sets M and E, where E is a set of equivalence relations on M, $M = \bigcup_{X\in E} \mathsf{Fld}X$, being $\mathsf{Fld}X$ the field of the relation X, and a function $f : L \to E$ such that $\mathcal{E} = (E, \subseteq, \cap, \mid)$ is a lattice of commuting equivalence relations and f is an isomorphism from $\mathcal{L}$ onto $\mathcal{E}$. Taking $\mathcal{E}$ and observing that fv is an assignment for $\textsc{Lvar}$ in $\mathcal{E}$, since f is an isomorphism, we have also that $[\![P_j]\!]^{fv}_{\mathcal{E}} \subseteq [\![Q_j]\!]^{fv}_{\mathcal{E}}$, for every $j \in J$, but $[\![P]\!]^{fv}_{\mathcal{E}} \not\subseteq [\![Q]\!]^{fv}_{\mathcal{E}}$. Now, take $\mathcal{M} = (M, \{r^{\mathcal{M}}_j : j \in J\})$, where $r^{\mathcal{M}}_j := f(v(p_j))$, for every $j \in J$. We have that $\mathcal{M}$ is a model, $\mathcal{M} \models \mathsf{LLat}$, and that $[\![R_P]\!]_{\mathcal{M}} = [\![P]\!]^{fv}_{\mathcal{E}}$ for every lattice term P. So, for $\mathcal{M}$ we have $[\![R_{P_j}]\!]_{\mathcal{M}} \subseteq [\![R_{Q_j}]\!]_{\mathcal{M}}$, for every $j \in J$, but $[\![R_P]\!]_{\mathcal{M}} \not\subseteq [\![R_Q]\!]_{\mathcal{M}}$, a contradiction with the strong soundness theorem for $+\mathsf{RG}$. □

Author Index