

# Lecture Notes in Computer Science

2800

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

**Springer**

*Berlin*

*Heidelberg*

*New York*

*Hong Kong*

*London*

*Milan*

*Paris*

*Tokyo*

Stefano Spaccapietra  
Sal March  
Karl Aberer (Eds.)

# Journal on Data Semantics I



Springer

### Series Editors

Gerhard Goos, Karlsruhe University, Germany  
Juris Hartmanis, Cornell University, NY, USA  
Jan van Leeuwen, Utrecht University, The Netherlands

### Volume Editors

Stefano Spaccapietra  
Database Laboratory  
School of Computer and Communication Science  
EPFL  
1015 Lausanne, Switzerland  
E-mail: stefano.spaccapietra@epfl.ch

### Sal March

Vanderbilt University  
Owen Graduate School of Management  
Nashville, TN, USA  
E-mail: Sal.March@Owen.Vanderbilt.edu

### Karl Aberer

Distributed Information Systems Laboratory (LSIR)  
Institute of Core Computing Science (IIF)  
School of Computer and Communication Science (I&C)  
EPFL  
1015 Lausanne, Switzerland  
E-mail: karl.aberer@epfl.ch

### Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

### Bibliographic information published by Die Deutsche Bibliothek

Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <<http://dnb.ddb.de>>.

### CR Subject Classification (1998): H.2, I.2, H.3, H.4, C.2

### ISSN 0302-9743

ISBN 3-540-20407-5 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York  
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springeronline.com>

© Springer-Verlag Berlin Heidelberg 2003  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin GmbH  
Printed on acid-free paper SPIN: 10963210 06/3142 5 4 3 2 1 0

# Foreword

This book constitutes the first volume of the first journal in the new LNCS Journal Subline, the Journal on Data Semantics. Publishing a journal in a book series might come as a surprise to customers, readers, and librarians, thus we would like to provide some background information and our motivation for introducing this new LNCS subline.

As a consequence of the very tight interaction between the Lecture Notes in Computer Science series and the international computer science research and development community, we receive quite a few proposals for new archive journals. From the successful launch of workshops or conferences and publication of their proceedings in the LNCS series, it might seem like a natural step to approach the publisher about launching a journal once this specific field has gained a certain level of maturity and stability. Each year we receive about a dozen such proposals and even more informal inquiries.

Like other publishers, it has been our experience that launching a new journal and making it a long-term success is a hard job nowadays, due to a generally difficult market situation, and library budget restrictions in particular. Because many of the proceedings in LNCS, and especially many of the LNCS postproceedings, apply the same strict reviewing and selection criteria as established journals, we started discussing with proposers of new journals the alternative of devoting a few volumes in LNCS to their field, instead of going through the painful Sisyphean adventure of establishing a new journal on its own. The advantages are obvious: the worldwide circulation and the international visibility of LNCS are as high as that of an average established journal, and the page price in LNCS is factors lower. Furthermore, such a journal-substitute volume would be available in LNCS Online and would thus be embedded into a comprehensive and popular digital library, in the direct neighborhood of many related conference papers.

As you will see from the frontmatter pages that follow, the LNCS Journal on Data Semantics has an Editor-in-Chief, who holds the overall mid-term editorial responsibility, and an Editorial Board consisting of excellent researchers in the area, who will devote their expertise to thoroughly reviewing the papers submitted for evaluation. In certain cases, like the present one, comparable to a special issue of a journal, the Editor-in-Chief may co-opt guest editors to jointly review the candidate papers for the volume. The Journal on Data Semantics can be subscribed to independently of the LNCS mother series, institutionally as well as individually, and, in contrast to a journal, each volume of the Journal on Data Semantics can be purchased individually at an affordable price.

Given that we have a number of pending proposals, several other LNCS journals might follow soon. We are convinced that our approach is a timely service to the international computer science research community, adding one more facet to the Lecture Notes in Computer Science series, as the most comprehensive publication platform for this community.

September 2003

Alfred Hofmann,  
Springer-Verlag

# LNCS Journal on Data Semantics Preface

Congratulations! What you have in your hands is the first issue of this new LNCS Journal on Data Semantics. We hope you will find much information of interest to you in the papers contained in this issue, and that you are already looking forward to the next one.

We felt that producing a new journal on this theme was most appropriate at this time, considering the evolution of computer science and practice. Computerized information handling has recently changed its focus from centralized data management systems to decentralized data exchange facilities. Modern distribution channels, such as high-speed Internet networks and wireless communication infrastructures, provide reliable technical support for data distribution and data access, realizing the new popular idea that data may be available to anybody, anywhere, anytime. However, providing huge amounts of data on request often turns out to be a counterproductive service, making the data useless because of poor relevance or an inappropriate level of detail. Semantic knowledge is the essential missing piece that allows the delivery of information that matches user requirements. Semantic agreement, in particular, is essential to meaningful data exchange.

Semantic issues have long been open topics in data and knowledge management. However, the boom in semantically poor technologies, such as the Web and XML, has excited renewed interest in semantics. For instance, conferences on the Semantic Web attract crowds of participants, while ontologies on their own has become a hot topic in the database and artificial intelligence communities.

This new journal aims to provide a highly visible dissemination channel for remarkable work that in one way or another addresses research and development on issues related to data semantics. The target domain ranges from theories supporting the formal definition of semantic content to innovative domain-specific applications of semantic knowledge. We expect such a publication channel to be of highest interest to researchers and advanced practitioners working on the Semantic Web, interoperability, mobile information services, data warehousing, knowledge representation and reasoning, conceptual database modeling, ontologies, and artificial intelligence.

Topics of relevance to this journal include:

- Semantic interoperability, semantic mediators
- Ontologies
- Ontology, schema and data integration, reconciliation and alignment
- Multiple representations, alternative representations
- Knowledge representation and reasoning
- Conceptualization and representation
- Multimodel and multiparadigm approaches
- Mappings, transformations, reverse engineering
- Metadata

- Conceptual data modeling
- Integrity description and handling
- Evolution and change
- Web semantics and semistructured data
- Semantic caching
- Data warehousing and semantic data mining
- Spatial, temporal, multimedia and multimodal semantics
- Semantics in data visualization
- Semantic services for mobile users
- Supporting tools
- Applications of semantic-driven approaches

These topics are to be understood as specifically related to semantic issues. Contributions dealing with the semantics of data may be considered even if they are not covered by the topics in the list.

While the physical appearance of each journal issue looks like that of the books in the well-known Springer LNCS series, the mode of operation will be that of a journal. That is to say, publication results from contributions freely submitted by authors and reviewed by the Editorial Board. Contributions may also be invited, and nevertheless carefully reviewed, as was the case for this first issue that publishes extended versions of some of the best papers addressing data semantics topics presented at major conferences in 2002. Special issues are foreseen, focusing on specific topics under the responsibility of guest editors. Finally, it is also possible that a journal issue could be devoted to a single text.

The journal is currently aiming at one to three volumes per year.

The Editorial Board comprises one Editor-in-Chief (with overall responsibility) and a board with a number of members. The Editor-in-Chief has a four-year mandate to run the journal. Board members have three-year mandates. Mandates are renewable. More members may be added to the board as appropriate.

We are happy to welcome you aboard our readership and hope we will share this privileged contact for a long time.

September 2003

Stefano Spaccapietra  
Editor-in-Chief



# LNCS Journal on Data Semantics – Editorial Board

Carlo Batini	Università di Milano-Bicocca, Italy
Tiziana Catarci	Università di Roma, La Sapienza, Italy
Lois Delcambre	Oregon Health and Science University, USA
David W. Embley	Brigham Young University, USA
Jérôme Euzenat	INRIA Rhône-Alpes, France
Dieter Fensel	University of Innsbruck, Austria
	and National University of Ireland, Galway
Nicola Guarino	National Research Council, Italy
Jean-Luc Hainaut	FUNDP, Namur, Belgium
Ian Horrocks	University of Manchester, UK
Yahiko Kambayashi	Kyoto University, Japan
Larry Kerschberg	George Washington University, USA
Maurizio Lenzerini	Università di Roma, La Sapienza, Italy
Tok Wang Ling	National University of Singapore, Singapore
Salvatore T. March	Vanderbilt University, USA
Robert Meersman	Vrije Universiteit Brussel (VUB), Belgium
John Mylopoulos	University of Toronto, Canada
Antoni Olivé	Universitat Politècnica de Catalunya, Spain
José Palazzo M. de Oliveira	Universidade Federal do Rio Grande do Sul, Brazil
Christine Parent	Université de Lausanne, Switzerland
John Roddick	Flinders University, Australia
Klaus-Dieter Schewe	Massey University, New Zealand
Bernhard Thalheim	Brandenburg Technical University, Germany
Yair Wand	University of British Columbia, Canada
Esteban Zimanyi	Université Libre de Bruxelles (ULB), Belgium

# Formal Reasoning Techniques for Goal Models<sup>\*</sup>

Paolo Giorgini<sup>1</sup>, John Mylopoulos<sup>2</sup>, Eleonora Nicchiarelli<sup>1</sup>, and  
Roberto Sebastiani<sup>1</sup>

<sup>1</sup> Department of Information and Communication Technology  
University of Trento - Italy  
{pgiorgini,eleonora,rseba}@dit.unitn.it

<sup>2</sup> Computer Science Department - University of Toronto - Canada  
jm@cs.toronto.edu

**Abstract.** Over the past decade, goal models have been used in Computer Science in order to represent software requirements, business objectives and design qualities. Such models extend traditional AI planning techniques for representing goals by allowing for partially defined and possibly inconsistent goals. This paper presents a formal framework for reasoning with such goal models. In particular, the paper proposes a qualitative and a numerical axiomatization for goal modeling primitives and introduces label propagation algorithms that are shown to be sound and complete with respect to their respective axiomatizations. In addition, the paper reports on experimental results on the propagation algorithms applied to a goal model for a US car manufacturer.

## 1 Introduction

The concept of goal has been used in many areas of Computer Science for quite some time. In AI, goals have been used in planning to describe desirable states of the world since the 60s [11]. In AI planning problems, an agent is given a description of the environment and a set of admissible actions, and searches for a *plan* —i.e., a sequence of actions and intermediate sub-goals— which allows for achieving a final goal from a given initial state. Very efficient planning algorithms and tools, including those based on graph-based planning [3] and on SAT conversions [9] have been conceived (see, e.g., [17] for an overview). More recently, goals have been used in Software Engineering to model (early) requirements [6, 1, 2, 8] and non-functional requirements [10] for a software system. For instance, an early requirement for a library information system might be “*Every book request will eventually be fulfilled*”, while “*The new system will be highly reliable*” is an example of a non-functional requirement. Goals are also useful for knowledge management systems which focus on strategic knowledge, e.g., “*Increase profits*”, or “*Become the largest auto manufacturer in North America*” [7].

---

<sup>\*</sup> This is an extended and updated version of the paper: Giorgini, P., Mylopoulos, J., Nicchiarelli, E., and Sebastiani, R., “Reasoning with Goal Models”. In S. Spaccapietra, S. T. March, and Y. Kambayashi (Eds.), Conceptual Modeling - ER 2002, proceedings of the 21st International Conference on Conceptual Modeling, LNCS 2503 Springer, 2002.

Traditional goal analysis consists of decomposing goals into subgoals through an AND- or OR-decomposition. If goal  $G$  is AND-decomposed (respectively, OR-decomposed) into subgoals  $G_1, G_2, \dots, G_n$ , then if all (at least one) of the subgoals are satisfied so is goal  $G$ . Given a goal model consisting of goals and AND/OR relationships among them, and a set of initial labels for some nodes of the graph (S for Satisfied, D for Denied) there is a simple label propagation algorithm which can generate labels for all other nodes of the graph [12].

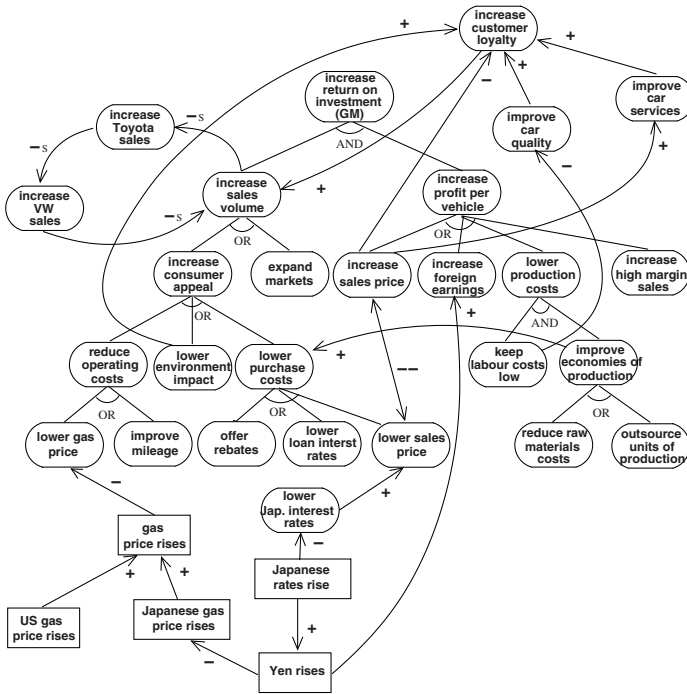
Unfortunately, this simple framework for modeling and analyzing goals won't work for many domains where goals are not formalizable and the relationships among them can't be captured by semantically well-defined relations such as AND/OR ones. For example, goals such as "*Highly reliable system*" have no formally defined predicate which prescribes their meaning, though one may want to define necessary conditions for such a goal to be satisfied. Moreover, such a goal may be related to other goals, such as "*Thoroughly debugged system*", "*Thoroughly tested system*" in the sense that the latter obviously contribute to the satisfaction of the former, but this contribution is partial and qualitative. In other words, if the latter goals are satisfied, they certainly contribute to the satisfaction of the former goal, but don't guarantee its satisfaction. This framework will also not work in situations where there are contradictory contributions to a goal. For instance, we may want to allow for multiple decompositions of a goal  $G$  into sets of subgoals, where some decompositions suggest satisfaction of  $G$  while others suggest denial.

We are interested in a modeling framework for goals which includes AND/OR relationships among goals, but also allows for other, more qualitative goal relationships and can accommodate contradictory situations [4, 16]. We accomplish this by introducing goal relationships labelled "+" and "-" which model respectively a situation where a goal contributes positively or negatively towards the satisfaction of another goal.

A major problem that arises from this extension is giving a precise semantics to the new goal relationships. This is accomplished in two different ways in this paper. In section 3 we offer a qualitative formalization and a label propagation algorithm which is shown to be sound and complete with respect to the formalization. In section 4, we offer a quantitative semantics for the new relationships which is based on a probabilistic model, and a label propagation algorithm that is also shown to be sound and complete. Section 5 presents experimental results on our label propagation algorithms using medium- to large-size goal models, while section 6 summarizes the contributions of the paper and sketches directions for further research.

## 2 An Example

Suppose we are modeling the strategic objectives of a US car manufacturer, such as Ford or GM. Examples of such objectives are **increase return on investment** or **increase customer loyalty**. Objectives can be represented as goals, and can be analyzed using goal relationships such as AND, OR, "+" and "-". In addition,



**Fig. 1.** A partial goal model for GM.

we will use “++” (respectively “--”) as a binary goal relationship such that if ++(G,G') (–(G,G')) then satisfaction of G implies satisfaction (denial) of G'.

For instance, increase return on investment may be AND-decomposed into increase sales volume and increase profit per vehicle. Likewise, increase sales volume might be OR-decomposed into increase consumer appeal and expand markets. This decomposition and refinement of goals can continue until we have goals that are tangible (i.e., someone can satisfy them through an appropriate course of action) and/or are observable (i.e., they can be confirmed satisfied/denied by simply observing the application domain).

For vaguely stated goals, such as increase customer loyalty we may want to simply model other relevant goals, such as improve car quality, improve car service and relate them through “+” and “-” relationships, as shown in Figure 1. These goals may influence positively or negatively some of the goals that have already been introduced during the analysis of the goal increase return on investment. Such lateral goal relationships may introduce cycles in our goal models.

Examples of observable goals are Yen rises, gas prices rise etc. When such a goal is satisfied, we will call it an *event* (the kind of event you may read about in a news story) and represent it in our graphical notation as a rectangle (see lower portion of Figure 1).

Figure 1 shows a partial and fictitious goal model for GM focusing on the goal increase return of investment. In order to increase return of investment, GM has to satisfy both goals increase sales and increase profit per vehicle. In turn, increase

sales volume is OR-decomposed into increase consumer appeal and expand markets, while the goal increase profit per vehicle is OR-decomposed into increase sales price, lower production costs, increase foreign earnings, and increase high margin sales. Additional decompositions are shown in the figure. For instance, the goal increase consumer appeal can be satisfied by satisfying lower environment impact, trying to lower purchase costs, or reducing the vehicle operating costs (reduce operating costs).

The graph shows also lateral relationships among goals. For example, the goal increase customer loyalty has positive (+) contributions from goals lower environment impact, improve car quality and improve car services, while it has a negative (−) contribution from increase sales price. The root goal increase return on investment (GM) is also related to goals concerning others auto manufacturer, such as Toyota and VW. In particular, if GM increases sales, then Toyota loses a share of the North American market; if Toyota increases sales (increase Toyota sales), it does so at the expense of VW; finally, if VW increases sales (increase VW sales), it does so at the expense of GM. These lateral contributions introduce a cycle in the GM goal model.

So far, we have assumed that every goal relationship treats S and D in a dual fashion. For instance, if we have  $+(G, G')$ , then if G is satisfied, G' is partially satisfied, and (dually) if G is denied G' is partially denied. Note however, that sometimes a goal relationship only applies for S (or D). In particular, the − contribution from increase sales (GM) to increase sales (Toyota) only applies when increase sales (GM) is satisfied (i.e., if GM hasn't increased sales, this doesn't mean that Toyota has.) To capture this kind of relationship, we introduce  $-_S$ ,  $-_D$ ,  $+_S$ ,  $+_D$  (see also Figure 1). Details about the semantics of these relationships are given in the next section.

### 3 Qualitative Reasoning with Goal Models

Formally, a *goal graph* is a pair  $\langle \mathcal{G}, \mathcal{R} \rangle$  where  $\mathcal{G}$  is a set of goals and  $\mathcal{R}$  is a set of goal relations over  $\mathcal{G}$ . If  $(G_1, \dots, G_n) \xrightarrow{r} G$  is a goal relation in  $\mathcal{R}$ , we call  $G_1 \dots G_n$  *source goals* and  $G$  the *target goal* of  $r$ . To simplify the discussion, we consider only binary *OR* and *AND* goal relations. This is not restrictive, as all the operators we consider in this section and in Section 4 —i.e.,  $\wedge$ ,  $\vee$ ,  $\min$ ,  $\max$ ,  $\otimes$ ,  $\oplus$ — are associative and can be thus trivially used as n-ary operators.

#### 3.1 Axiomatization of Goal Relationships

Let  $G_1, G_2, \dots$  denote goal labels. We introduce four distinct predicates over goals,  $FS(G)$ ,  $FD(G)$  and  $PS(G)$ ,  $PD(G)$ , meaning respectively that there is (at least) *full* evidence that goal  $G$  is satisfied and that  $G$  is denied, and that there is at least *partial* evidence that  $G$  is satisfied and that  $G$  is denied. We also use the proposition  $\top$  to represent the (trivially true) statement that there is at least null evidence that the goal  $G$  is satisfied (or denied). Notice that the predicates state that there is *at least* a given level of evidence, because in a goal graph

Goal	Invariant Axioms	
$G :$	$FS(G) \rightarrow PS(G)$	(1)
	$FD(G) \rightarrow PD(G)$	(2)
Goal relation	Relation Axioms	
$(G_2, G_3) \xrightarrow{and} G_1 :$	$(FS(G_2) \wedge FS(G_3)) \rightarrow FS(G_1)$	(3)
	$(PS(G_2) \wedge PS(G_3)) \rightarrow PS(G_1)$	(4)
	$FD(G_2) \rightarrow FD(G_1), \quad FD(G_3) \rightarrow FD(G_1)$	(5)
	$PD(G_2) \rightarrow PD(G_1), \quad PD(G_3) \rightarrow PD(G_1)$	(6)
$(G_2, G_3) \xrightarrow{or} G_1 :$	$FS(G_2) \rightarrow FS(G_1), \quad FS(G_3) \rightarrow FS(G_1)$	(7)
	$PS(G_2) \rightarrow PS(G_1), \quad PS(G_3) \rightarrow PS(G_1)$	(8)
	$(FD(G_2) \wedge FD(G_3)) \rightarrow FD(G_1)$	(9)
	$(PD(G_2) \wedge PD(G_3)) \rightarrow PD(G_1)$	(10)
$G_2 \xrightarrow{+S} G_1 :$	$PS(G_2) \rightarrow PS(G_1)$	(11)
$G_2 \xrightarrow{-S} G_1 :$	$PS(G_2) \rightarrow PD(G_1)$	(12)
$G_2 \xrightarrow{++S} G_1 :$	$FS(G_2) \rightarrow FS(G_1),$	(13)
	$PS(G_2) \rightarrow PS(G_1)$	(14)
$G_2 \xrightarrow{--S} G_1 :$	$FS(G_2) \rightarrow FD(G_1),$	(15)
	$PS(G_2) \rightarrow PD(G_1)$	(16)
$G_2 \xrightarrow{+D} G_1 :$	$PD(G_2) \rightarrow PD(G_1)$	(17)
$G_2 \xrightarrow{-D} G_1 :$	$PD(G_2) \rightarrow PS(G_1)$	(18)
$G_2 \xrightarrow{++D} G_1 :$	$FD(G_2) \rightarrow FD(G_1),$	(19)
	$PD(G_2) \rightarrow PD(G_1)$	(20)
$G_2 \xrightarrow{--D} G_1 :$	$FD(G_2) \rightarrow FS(G_1),$	(21)
	$PD(G_2) \rightarrow PS(G_1)$	(22)

**Fig. 2.** Ground axioms for the invariants and the propagation rules in the qualitative reasoning framework.

there may be multiple sources of evidence for the satisfaction/denial of a goal. We introduce a total order  $FS(G) \geq PS(G) \geq \top$  and  $FD(G) \geq PD(G) \geq \top$ , with the intended meaning that  $x \geq y$  iff  $x \rightarrow y$ .

We want to allow the deduction of *positive* ground assertions of type  $FS(G)$ ,  $FD(G)$ ,  $PS(G)$  and  $PD(G)$  over the goal constants of a goal graph. We refer to externally provided assertions as *initial conditions*. To formalize the propagation of satisfiability and deniability evidence through a goal graph  $\langle \mathcal{G}, \mathcal{R} \rangle$ , we introduce the axioms described in Figure 2. (By “dual” we mean that we invert satisfiability with deniability.)

As indicated in Section 2, the propagation rules for goal satisfaction through  $++$ ,  $--$ ,  $+$ ,  $-$  relationships may or may not be symmetric w.r.t. those for denial. For example, the relationship  $G_2 \xrightarrow{+} G_1$  might propagate only the satisfaction of  $G_2$ , only the denial, or both.

Thus, for every relation type  $r \in \{++, --, +, -\}$ , it makes sense to have three possible labels: “ $r_S$ ”, “ $r_D$ ”, and “ $r$ ”, meaning respectively that satisfaction is propagated, that denial is propagated, and that both satisfaction and denial are propagated. (We call the first two cases *asymmetric*, the latter *symmetric*.) For example,  $G_2 \xrightarrow{-S} G_1$  means that if  $G_2$  is satisfied, then there is some evidence that  $G_1$  is denied, but if  $G_2$  is denied, then nothing is said about the satisfaction of  $G_1$ ;  $G_2 \xrightarrow{-D} G_1$  means that if  $G_2$  is denied, then there is some evidence that  $G_1$  is satisfied, but if  $G_2$  is satisfied, then nothing is said about the denial of  $G_1$ ;  $G_2 \xrightarrow{-} G_1$  means that, if  $G_2$  is satisfied [denied], then there is some evidence that  $G_1$  is denied [satisfied]. In other words, a symmetric relation  $G_2 \xrightarrow{r} G_1$  is a shorthand for the combination of the two corresponding asymmetric relationships  $G_2 \xrightarrow{rS} G_1$  and  $G_2 \xrightarrow{rD} G_1$ .

(1) and (2) state that full satisfiability and deniability imply partial satisfiability and deniability respectively. For an AND relation, (3) and (4) show that the full and partial satisfiability of the target node require respectively the full and partial satisfiability of all the source nodes; for a “ $+_S$ ” relation, (11) show that only the partial satisfiability (but not the full satisfiability) propagates through a “ $+_S$ ” relation. Combining (1) with (3), and (1) with (11), we have, respectively,

$$(G_2, G_3) \xrightarrow{and} G_1 : (FS(G_2) \wedge PS(G_3)) \rightarrow PS(G_1) \quad (23)$$

$$G_2 \xrightarrow{+_S} G_1 : FS(G_2) \rightarrow PS(G_1). \quad (24)$$

Thus, an AND relation propagates the minimum satisfiability value (and the maximum deniability one), while a “ $+_S$ ” relation propagates at most a partial satisfiability value. To this extent, notice that a “ $+_S$ ” relation can be seen as an AND relation with an unknown partially satisfiable goal.

From now on, we implicitly assume that axioms (1) and (2) are always applied whenever possible. Thus, we say that  $PS(G_1)$  is deduced from  $FS(G_2)$  and  $FS(G_3)$  by applying (3) —meaning “applying (3) and then (1)” — or that  $PS(G_1)$  is deduced from  $FS(G_2)$  and  $PS(G_3)$  by applying (4) —meaning “applying (1) and then (4)”.

We say that an atomic proposition of the form  $FS(G)$ ,  $FD(G)$ ,  $PS(G)$  and  $PD(G)$  *holds* if either it is an initial condition or it can be deduced via modus ponens from the initial conditions and the ground axioms of Figure 2. We assume conventionally that  $\top$  always holds. Notice that all the formulas in our framework are propositional Horn clauses, so that deciding if a ground assertion holds not only is decidable, but also it can be decided in polynomial time.

We say that there is a *weak conflict* if either  $PS(G)$  and  $PD(G)$ ,  $FS(G)$  and  $PD(G)$ ,  $PS(G)$  and  $FD(G)$  hold for some goal  $G$ . We say that there is a *strong conflict* if  $FS(G)$  and  $FD(G)$  hold for some  $G$ .

### 3.2 The Label Propagation Algorithm

Based on the logic framework of Section 3.1, we have developed an algorithm for propagating through a goal graph  $\langle \mathcal{G}, \mathcal{R} \rangle$  labels representing evidence for

**Table 1.** Propagation rules in the qualitative framework.

	$(G_2, G_3) \xrightarrow{and} G_1$	$G_2 \xrightarrow{+S} G_1$	$G_2 \xrightarrow{-S} G_1$	$G_2 \xrightarrow{++S} G_1$	$G_2 \xrightarrow{--S} G_1$
$Sat(G_1)$	$\min \left\{ \begin{array}{l} Sat(G_2), \\ Sat(G_3) \end{array} \right\}$	$\min \left\{ \begin{array}{l} Sat(G_2), \\ P \end{array} \right\}$	$N$	$Sat(G_2)$	$N$
$Den(G_1)$	$\max \left\{ \begin{array}{l} Den(G_2), \\ Den(G_3) \end{array} \right\}$	$N$	$\min \left\{ \begin{array}{l} Sat(G_2), \\ P \end{array} \right\}$	$N$	$Sat(G_2)$

	$(G_2, G_3) \xrightarrow{or} G_1$	$G_2 \xrightarrow{+D} G_1$	$G_2 \xrightarrow{-D} G_1$	$G_2 \xrightarrow{++D} G_1$	$G_2 \xrightarrow{--D} G_1$
$Sat(G_1)$	$\max \left\{ \begin{array}{l} Sat(G_2), \\ Sat(G_3) \end{array} \right\}$	$N$	$\min \left\{ \begin{array}{l} Den(G_2), \\ P \end{array} \right\}$	$N$	$Den(G_2)$
$Den(G_1)$	$\min \left\{ \begin{array}{l} Den(G_2), \\ Den(G_3) \end{array} \right\}$	$\min \left\{ \begin{array}{l} Den(G_2), \\ P \end{array} \right\}$	$N$	$Den(G_2)$	$N$

the satisfiability and deniability of goals. To each node  $G \in \mathcal{G}$  we associate two variables  $Sat(G), Den(G)$  ranging in  $\{F, P, N\}$  (full, partial, none) such that  $F > P > N$ , representing the current evidence of satisfiability and deniability of goal  $G$ . For example,  $Sat(G_i) \geq P$  states that there is at least partial evidence that  $G_i$  is satisfiable. Starting from assigning an initial set of input values for  $Sat(G_i), Den(G_i)$  to (a subset of) the goals in  $\mathcal{G}$ , we propagate the values through the goal relations in  $\mathcal{R}$  according to the propagation rules of Table 1.

The schema of the algorithm is described in Figure 3. *Initial*, *Current* and *Old* are arrays of  $|\mathcal{G}|$  pairs  $\langle Sat(G_i), Den(G_i) \rangle$ , one for each  $G_i \in \mathcal{G}$ , representing respectively the initial, current and previous labeling states of the graph. We call the pair  $\langle Sat(G_i), Den(G_i) \rangle$  a *label* for  $G_i$ . Notationally, if  $W$  is an array of labels  $\langle Sat(G_i), Den(G_i) \rangle$ , by  $W[i].sat$  and  $W[i].den$  we denote the first and second field of the  $i$ th label of  $W$ .

The array *Current* is first initialized to the initial values *Initial* given as input by the user. At each step, for every goal  $G_i$ ,  $\langle Sat(G_i), Den(G_i) \rangle$  is updated by propagating the values of the previous step. This is done until a fixpoint is reached, in the sense that no further updating is possible ( $Current == Old$ ).

The updating of  $\langle Sat(G_i), Den(G_i) \rangle$  works as follows. For each relation  $R_j$  incoming in  $G_i$ , the satisfiability and deniability values  $sat_{ij}$  and  $den_{ij}$  derived from the old values of the source goals are computed by applying the rules of Table 1. The result is compared with the old value, and the maximum is returned as new value for  $G_i$ .

### 3.3 Termination and Complexity

**Theorem 1.** *Label\_Graph( $\langle \mathcal{G}, \mathcal{R} \rangle, Initial$ ) terminates after at most  $6|\mathcal{G}| + 1$  loops.*

*Proof.* First, from lines 6 and 14 in Figure 3 we have that, for every goal  $G_i$ ,

$$\begin{aligned} Current[i].sat &= \max(\dots, Old[i].sat), \\ Current[i].den &= \max(\dots, Old[i].den) \end{aligned}$$



```

1  label_array Label_Graph(graph  $\langle \mathcal{G}, \mathcal{R} \rangle$ , label_array Initial)
2    Current=Initial;
3    do
4      Old=Current;
5      for each  $G_i \in \mathcal{G}$  do
6        Current[i] = Update_label(i,  $\langle \mathcal{G}, \mathcal{R} \rangle$ , Old);
7      until not (Current==Old);
8    return Current;
9
10 label Update_label(int i, graph  $\langle \mathcal{G}, \mathcal{R} \rangle$ , label_array Old)
11   for each  $R_j \in \mathcal{R}$  s.t. target( $R_j$ ) ==  $G_i$  do
12     satij = Apply_Rules_Sat( $G_i, R_j, Old$ );
13     denij = Apply_Rules_Den( $G_i, R_j, Old$ );
14   return  $\langle \max(\max_j(\text{sat}_{ij}), \text{Old}[i].\text{sat}), \max(\max_j(\text{den}_{ij}), \text{Old}[i].\text{den}) \rangle$ 

```

**Fig. 3.** Schema of the label propagation algorithm.

so that their values are monotonically non-decreasing. In order not to terminate, at least one value per step should monotonically increase. Each of the  $2|\mathcal{G}|$  variables  $Sat(G_i)$  and  $Den(G_i)$  admits 3 possible values and at each non-final loop at least one value increases. Thus the procedure must terminate after at most  $6|\mathcal{G}| + 1$  loops.  $\square$

Notice that the upper bound is very pessimistic, as many value updates are done in parallel. In Section 5, we report on experiments we have conducted which suggest that the algorithm generally converges after a few loops.

### 3.4 Soundness and Completeness

We call a *value statement* an expression of the form  $(v \geq c)$ ,  $v \in \{Sat(G_i), Den(G_i)\}$  for some goal  $G_i$  and  $c \in \{F, P, N\}$ , with the intuitive meaning “there is at least evidence  $c$  for  $v$ ”. Thus from now on we rewrite the assertion  $FS(G)$ ,  $PS(G)$ ,  $\top$  as  $(Sat(G) \geq F)$ ,  $(Sat(G) \geq P)$ ,  $(Sat(G) \geq N)$  and  $FD(G)$ ,  $PD(G)$ ,  $\top$  as  $(Den(G) \geq F)$ ,  $(Den(G) \geq P)$ ,  $(Sat(G) \geq N)$  respectively.

We say that  $(v_1 \geq c_1)$  is *deduced* from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] by a relation axiom meaning that the corresponding assertions are deduced. For instance,  $(Sat(G_1) \geq P)$  is deduced from  $(Sat(G_2) \geq F)$  by axiom (11) as  $PS(G_1)$  is deduced from  $FS(G_2)$  by axiom (11).

We say that  $(v_1 \geq c_1)$  *derives* from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] by means of one propagation rule  $x = f(y, z)$  [ $x = f(y)$ ] of Table 1 if  $c_1 = f(c_2, c_3)$  [ $c_1 = f(c_2)$ ]. For instance,  $(Sat(G_1) \geq P)$  derives from  $(Sat(G_2) \geq P)$  and  $(Sat(G_3) \geq F)$  by means of the first propagation rule. Notice that this is possible because the

operators  $\min$  and  $\max$  are *monotonic*, that is, e.g.,  $\max(v_1, v_2) \geq \max(v'_1, v'_2)$  iff  $v_1 \geq v'_1$  and  $v_2 \geq v'_2$ .

To this extent, the rules in Table 1 are a straightforward translation of the axioms (1)-(22), as stated by the following result.

**Lemma 1.**  $(v_1 \geq c_1)$  derives from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] by means of the propagation rules of Table 1 if and only if  $(v_1 \geq c_1)$  is deduced from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] with the application of one of the relation axioms (3)-(22).

*Proof.* For short, we consider only the AND and  $+_S$  cases for  $Sat(G_1)$ , as the other cases are either analogous or trivial and can be verified by the reader.

$(G_2, G_3) \xrightarrow{and} G_1$ : If either  $(Sat(G_2) \geq N)$  or  $(Sat(G_3) \geq N)$ , then  $(Sat(G_1) \geq N)$  is derived. This matches one-to-one the fact that from  $\top$  nothing else is deduced;

otherwise, if either  $(Sat(G_2) \geq P)$  or  $(Sat(G_3) \geq P)$ , then  $(Sat(G_1) \geq P)$  is derived. This matches one-to-one the fact that from  $(Sat(G_2) \geq P)$  and  $(Sat(G_3) \geq P)$  axiom (4) is applied, so that  $(Sat(G_1) \geq P)$  is deduced;

finally, if both  $(Sat(G_2) \geq F)$  and  $(Sat(G_3) \geq F)$ , then  $(Sat(G_1) \geq F)$ . This matches one-to-one the fact that from  $(Sat(G_2) \geq F)$  and  $(Sat(G_3) \geq F)$ , axiom (3) is applied, so that  $(Sat(G_1) \geq F)$  is deduced.

$G_2 \xrightarrow{+_S} G_1$ : If  $(Sat(G_2) \geq N)$ , then  $(Sat(G_1) \geq N)$ . Again, this matches one-to-one the fact that from  $\top$  nothing else is deduced;

otherwise, if either  $(Sat(G_2) \geq P)$  or  $(Sat(G_2) \geq F)$ , then we have that  $Sat(G_1) \geq \min(Sat(G_2), P) \geq P$ . This matches one-to-one the fact that from either  $(Sat(G_2) \geq P)$  or  $(Sat(G_2) \geq F)$  axiom (11) is applied and  $(Sat(G_1) \geq P)$  is deduced.  $\square$

Given an array of labels  $W$ , we say that  $(Sat(G_i) \geq c)$  [resp.  $(Den(G_i) \geq c)$ ] is true in  $W$  if and only if  $W[i].sat \geq c$  [resp.  $W[i].den \geq c$ ]. This allows us to state the correctness and completeness theorem for *Label\_Graph*().

**Theorem 2.** Let *Final* be the array returned by *Label\_Graph*( $\langle \mathcal{G}, \mathcal{R} \rangle, Initial$ ).  $(v \geq c)$  is true in *Final* if and only if  $(v \geq c)$  can be deduced from *Initial* by applying the relation axioms (3)-(22).

*Proof.* First we define inductively the notion of “deduced from *Initial* in  $k$  steps”: (i) an assertion in *Initial* can be deduced from *Initial* in 0 steps; (ii) an assertion can be deduced from *Initial* in up to  $k + 1$  steps if either it can be deduced from *Initial* in up to  $k$  steps or it can be deduced by applying a relation axiom to some assertions, all of which can be deduced from *Initial* in up to  $k$  steps.

Let  $Current_k$  be the value of *Current* after  $k$  loops. We show that  $(v \geq c)$  is true in  $Current_k$  if and only if  $(v \geq c)$  can be deduced from *Initial* in up to  $k$  steps. The thesis follows from the fact that  $Final = Current_k$  for some  $k$ .

We reason by induction on  $k$ . The base case  $k = 0$  is obvious as  $Current_0 = Initial$ . By inductive hypothesis, we assume the thesis for  $k$  and we prove it for  $k + 1$ .

**If.** Consider  $(v \geq c)$  such that  $(v \geq c)$  is deduced from *Initial* in up to  $k + 1$  steps. Thus,  $(v \geq c)$  is obtained by applying some relation axiom  $AX$  to some assertion(s)  $(v_1 \geq c_1)$  [and  $(v_2 \geq c_2)$ ], which can be both deduced from *Initial* in up to  $k$  steps. Thus, by inductive hypothesis,  $(v_1 \geq c_1)$  [and  $(v_2 \geq c_2)$ ] occur in  $Current_k$ . Then, by Lemma 1,  $(v \geq c)$  derives from  $(v_1 \geq c_1)$  [and  $(v_2 \geq c_2)$ ] by means of the propagation rules of Table 1. Thus, if  $v$  is  $Sat(G_i)$  [resp  $Den(G_i)$ ], then  $c$  is one of the values  $sat_{ij}$  [resp.  $den_{ij}$ ] of lines 14, 15 in Figure 3, so that  $Current_{k+1}[i].sat \geq c$  [ $Current_{k+1}[i].den \geq c$ ]. Thus  $(v \geq c)$  is true in  $Current_{k+1}$ .

**Only if.** Consider a statement  $(v \geq c)$  true in  $Current_{k+1}$ . If  $(v \geq c)$  is true also in  $Current_k$ , then by inductive hypothesis  $(v \geq c)$  can be deduced from *Initial* in up to  $k$  steps, and hence in  $k+1$  steps. Otherwise, let  $R_j$  be the rule and let  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] the statements(s) true in  $Current_k$  from which  $v$  has been derived. By inductive hypothesis  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] can be deduced from *Initial* in up to  $k$  steps. By Lemma 1  $(v \geq c)$  can be deduced from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] by the application of one relation axiom, and thus can be deduced from *Initial* in up to  $k + 1$  steps.  $\square$

Thus, from Theorem 2, the values returned by  $Label\_Graph(\langle \mathcal{G}, \mathcal{R} \rangle, Initial)$  are the maximum evidence values which can be deduced from *Initial*.

## 4 Quantitative Reasoning with Goal Models

The qualitative approach of Section 3 allows for setting and propagating partial evidence about the satisfiability and deniability of goals and the discovery of conflicts.

We may want to provide a more fine-grained evaluation of such partial evidence. For instance, when we have  $G_2 \xrightarrow{+s} G_1$ , from  $PS(G_2)$  we can deduce  $PS(G_1)$ , whilst one may argue that the satisfiability of  $G_1$  is in some way less evident than that of  $G_2$ . For example, in the goal graph of Figure 1, the satisfaction of the goal **lower environment impact** may not necessarily imply satisfaction of **increase customer loyalty**, so it may be reasonable to assume "less evidence" for the satisfaction of the latter compared to the former. Moreover, the different relations which mean partial support – i.e.,  $+s$ ,  $-s$ ,  $+D$ ,  $-D$  – may have different strengths. For instance, in the goal graph of Figure 1, **US gas price rises** may have a bigger impact on **gas price rises** than **japanese gas price rises** if the manufacturer's market is mainly in the US.

To cope with these facts, we need a way for representing different *numerical* values of partial evidence for satisfiability/deniability and for attributing different weights to the  $+s$ ,  $-s$ ,  $+D$ ,  $-D$  relations. And, of course, we also need a formal framework to reason with such quantitative information.

### 4.1 An Extended Framework

In our second attempt, we introduce a *quantitative* framework, inspired by [5]. We introduce two real constants *inf* and *sup* such that  $0 \leq inf < sup$ . For

**Table 2.** Propagation rules in the quantitative framework.

	$(G_2, G_3) \xrightarrow{and} G_1$	$G_2 \xrightarrow{w+S} G_1$	$G_2 \xrightarrow{w-S} G_1$	$G_2 \xrightarrow{++S} G_1$	$G_2 \xrightarrow{--S} G_1$
$Sat(G_1)$	$Sat(G_2) \otimes Sat(G_3)$	$Sat(G_2) \otimes w$	$Sat(G_2) \otimes w$	$Sat(G_2)$	$Sat(G_2)$
$Den(G_1)$	$Den(G_2) \oplus Den(G_3)$				
	$(G_2, G_3) \xrightarrow{or} G_1$	$G_2 \xrightarrow{w+D} G_1$	$G_2 \xrightarrow{w-D} G_1$	$G_2 \xrightarrow{++D} G_1$	$G_2 \xrightarrow{--D} G_1$
$Sat(G_1)$	$Sat(G_2) \oplus Sat(G_3)$	$Den(G_2) \otimes w$	$Den(G_2) \otimes w$	$Den(G_2)$	$Den(G_2)$
$Den(G_1)$	$Den(G_2) \otimes Den(G_3)$				

each node  $G \in \mathcal{G}$  we introduce two *real* variables  $Sat(G), Den(G)$  ranging in the interval  $\mathcal{D} =_{def} [inf, sup]$ , representing the current evidence of satisfiability and deniability of the goal  $G$ . The intended meaning is that  $inf$  represents no evidence,  $sup$  represents full evidence, and different values in  $]inf, sup[$  represent different levels of partial evidence.

To handle the goal relations we introduce two operators  $\otimes, \oplus : \mathcal{D} \times \mathcal{D} \mapsto \mathcal{D}$  representing respectively the evidence of satisfiability of the conjunction and that of the disjunction [deniability of the disjunction and that of the conjunction] of two goals.  $\otimes$  and  $\oplus$  are associative, commutative and monotonic, and such that  $x \otimes y \leq x, y \leq x \oplus y$ ; there is also an implicit unary operator  $inv()$ , representing negation, such that  $inf = inv(sup)$ ,  $sup = inv(inf)$ ,  $inv(x \oplus y) = inv(x) \otimes inv(y)$  and  $inv(x \otimes y) = inv(x) \oplus inv(y)$ .

We also attribute to each goal relation  $+_S, -_S, +_D, -_D$  a weight  $w \in ]inf, sup[$  stating the strength by which the satisfiability/deniability of the source goal influences the satisfiability/deniability of the target goal. The propagation rules are described in Table 2. As in the qualitative approach, a symmetric relation —such as,  $G_2 \xrightarrow{w+} G_1$ — is a shorthand for the combination of the two corresponding asymmetric relationships sharing the same weight  $w$  —e.g.,  $G_2 \xrightarrow{w+S} G_1$  and  $G_2 \xrightarrow{w+D} G_1$ .

There are a few possible models following the schema described above. In particular, here we adopt a *probabilistic* model, where the evidence of satisfiability  $Sat(G)$  [resp. deniability  $Den(G)$ ] of  $G$  is represented as the probability that  $G$  is satisfied (respectively denied). As usual, we adopt the simplifying hypothesis that the different sources of evidence are independent. Thus, we fix  $inf = 0$ ,  $sup = 1$ , and we define  $\otimes, \oplus, inv()$  as:

$$p_1 \otimes p_2 =_{def} p_1 \cdot p_2, \quad p_1 \oplus p_2 =_{def} p_1 + p_2 - p_1 \cdot p_2, \quad inv(p_1) = 1 - p_1$$

that is, respectively the probability of the conjunction and disjunction of two independent events of probability  $p_1$  and  $p_2$ , and that of the negation of the first event. To this extent, the propagation rules in Table 2 are those of a Bayesian network, where, e.g., in  $G_2 \xrightarrow{w+S} G_1$   $w$  has to be interpreted as the conditional probability  $P[G_1 \text{ is satisfied} \mid G_2 \text{ is satisfied}]$ . Notice that the qualitative

Goal relation	Axioms	
$(G_2, G_3) \xrightarrow{and} G_1 :$	$(Sat(G_2) \geq x \wedge Sat(G_3) \geq y) \rightarrow Sat(G_1) \geq (x \otimes y)$	(25)
	$(Den(G_2) \geq x \wedge Den(G_3) \geq y) \rightarrow Den(G_1) \geq (x \oplus y)$	(26)
$(G_2, G_3) \xrightarrow{or} G_1 :$	$(Sat(G_2) \geq x \wedge Sat(G_3) \geq y) \rightarrow Sat(G_1) \geq (x \oplus y)$	(27)
	$(Den(G_2) \geq x \wedge Den(G_3) \geq y) \rightarrow Den(G_1) \geq (x \otimes y)$	(28)
$G_2 \xrightarrow{w+S} G_1 :$	$Sat(G_2) \geq x \rightarrow Sat(G_1) \geq (x \otimes w)$	(29)
$G_2 \xrightarrow{w-S} G_1 :$	$Sat(G_2) \geq x \rightarrow Den(G_1) \geq (x \otimes w)$	(30)
$G_2 \xrightarrow{++S} G_1 :$	$Sat(G_2) \geq x \rightarrow Sat(G_1) \geq x$	(31)
$G_2 \xrightarrow{--S} G_1 :$	$Sat(G_2) \geq x \rightarrow Den(G_1) \geq x$	(32)
$G_2 \xrightarrow{w+D} G_1 :$	$Den(G_2) \geq x \rightarrow Den(G_1) \geq (x \otimes w)$	(33)
$G_2 \xrightarrow{w-D} G_1 :$	$Den(G_2) \geq x \rightarrow Sat(G_1) \geq (x \otimes w)$	(34)
$G_2 \xrightarrow{++D} G_1 :$	$Den(G_2) \geq x \rightarrow Den(G_1) \geq x$	(35)
$G_2 \xrightarrow{--D} G_1 :$	$Den(G_2) \geq x \rightarrow Sat(G_1) \geq x$	(36)

**Fig. 4.** Axioms for the propagation rules in the quantitative reasoning framework.

framework of Section 3 can be seen as another such model with  $\mathcal{D} = \{F, P, N\}$ ,  $\otimes = \min()$  and  $\oplus = \max()$ .<sup>3</sup>

## 4.2 Axiomatization

As with the qualitative case, we call a *value statement* an expression of the form  $(v \geq c)$ ,  $v \in \{Sat(G_i), Den(G_i)\}$  for some  $G_i$  and  $c \in [0, 1]$ , with the intuitive meaning “there is at least evidence  $c$  of  $v$ ”. We want to allow the user to state and deduce non-negated value statements of the kind  $(Sat(G) \geq c)$  and  $(Den(G) \geq c)$  over the goal constants of the graph. As before, we call externally provided assertions about the satisfaction/denial of goals *initial conditions*.

To formalize the propagation of satisfiability and deniability evidence values through a goal graph, for every goal and goal relation in  $\langle \mathcal{G}, \mathcal{R} \rangle$ , we introduce the axioms (25)–(32) in Figure 4. Unlike those of Figure 2, the relation axioms in Figure 4 are not ground Horn clauses—thus, propositional—but rather first-order closed Horn formulas, so that they require a first-order deduction engine.

We say that a statement  $(v \geq c)$  holds if either it is an initial condition or it can be deduced from the initial conditions and the axioms of Figure 4. We implicitly assume that  $(Sat(G_i) \geq 0)$  and  $(Den(G_i) \geq 0)$  hold for every  $G_i$ , and that the deduction engine—either human or machine—can semantically evaluate  $\otimes$  and  $\oplus$  and perform deductions deriving from the values of the evaluated terms and the semantics of  $\geq$ . For instance, we assume that, if  $(G_2, G_3) \xrightarrow{and} G_1$  is in

<sup>3</sup> Another model of interest is that based on a serial/parallel resistance model, in which  $inf = 0$ ,  $sup = +\infty$ ,  $x \oplus y = x + y$ ,  $x \otimes y = \frac{x \cdot y}{x + y}$  and  $inv(x) = \frac{1}{x}$  [5].

$\mathcal{R}$ , then  $(Sat(G_1) \geq 0.1)$  can be deduced from  $(Sat(G_2) \geq 0.5)$ ,  $(Sat(G_3) \geq 0.4)$ , as from (25) it is deduced  $(Sat(G_1) \geq 0.5 \otimes 0.4)$ , which is evaluated into  $(Sat(G_1) \geq 0.2)$ , from which it can be deduced  $(Sat(G_1) \geq 0.1)$ .

We say that there is a *weak conflict* if both  $(Sat(G) \geq c_1)$  and  $(Den(G) \geq c_2)$  hold for some goal  $G$  and constants  $c_1, c_2 > 0$ . We say that there is a *strong conflict* if there is a weak conflict s.t.  $c_1 = c_2 = 1$ .

### 4.3 The Label Propagation Algorithm

Starting from the new numeric framework, we have adapted the label propagation algorithm of Figure 3 to work with numeric values. The new version of the algorithm differs from that of Section 3 in that: the elements in *Initial*, *Current* and *Old* range in  $[0, 1]$ ; the input graph contain also weights to the  $+_S$ ,  $-_S$ ,  $+_D$ ,  $-_D$  goal relations; and, the propagation rules applied are those of Table 2.

### 4.4 Soundness and Completeness

We say that  $(v_1 \geq c_1)$  *derives* from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] by means of one propagation rule  $x = f(y, z)$  [ $x = f(y)$ ] of Table 2 if  $c_1 = f(c_2, c_3)$  [ $c_1 = f(c_2)$ ]. For instance,  $(Sat(G_1) \geq 0.4)$  derives from  $(Sat(G_2) \geq 0.8)$  and  $(Sat(G_3) \geq 0.5)$  by means of the first and propagation rule. Again, this is possible because the the operators  $\otimes$  and  $\oplus$  are monotonic.

**Lemma 2.**  $(v_1 \geq c_1)$  *derives* from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] *by means of the propagation rules of Table 1 if and only if  $(v_1 \geq c_1)$  is deduced from  $(v_2 \geq c_2)$  [and  $(v_3 \geq c_3)$ ] with the application of one of the relation axioms (25)-(32).*

*Proof.* Trivial observation that the axioms (25)-(32) are straightforward translation of the propagation rules in Table 1.  $\square$

Given an array of labels  $W$ , we say that  $(Sat(G_i) \geq c)$  [resp.  $(Den(G_i) \geq c)$ ] *is true in  $W$*  if and only if  $W[i].sat \geq c$  [resp.  $W[i].den \geq c$ ]. This allows for stating the correctness and completeness theorem for *Label\_Graph()*.

**Theorem 3.** *Let  $Final$  be the array returned by  $Label\_Graph(\langle \mathcal{G}, \mathcal{R} \rangle, Initial)$ .  $(v \geq c)$  is true in  $Final$  if and only if  $(v \geq c)$  can be deduced from  $Initial$  by applying the relation axioms (25)-(32).*

*Proof.* Identical to that of Theorem 2, substituting the axioms (25)-(32) for the axioms (3)-(22), the rules in Table 2 for those in Table 1 and Lemma 2 for Lemma 1.  $\square$

Again, from Theorem 3, the values returned by  $Label\_Graph(\langle \mathcal{G}, \mathcal{R} \rangle, Initial)$  are the maximum evidence values which can be deduced from *Initial*.

## 4.5 Termination

To guarantee termination, the condition ( $Current == Old$ ) of line 7 in Figure 3 is implemented as:

$$\begin{aligned} \max_i (|Current[i].sat - Old[i].sat|) < \epsilon \text{ and} \\ \max_i (|Current[i].den - Old[i].den|) < \epsilon, \end{aligned} \quad (37)$$

$\epsilon$  being a sufficiently small real constant. (This is a standard practice to avoid numeric errors when doing computations with real numbers.) Thus, the algorithm loops until all satisfiability or deniability value variations have become negligible.

**Theorem 4.** *Label\_Graph( $\langle \mathcal{G}, \mathcal{R} \rangle$ , Initial) terminates in a finite number of loops.*

*Proof.* For the same reason as in Theorem 1, the values  $Current[i]_k.sat$  and  $Current[i]_k.den$  are monotonically non-decreasing with  $k$ . As every monotonically non-decreasing upper-bounded sequence is convergent, both  $Current[i]_k.sat$  and  $Current[i]_k.den$  are convergent. Thus they are also Cauchy-convergent.<sup>4</sup> It follows that condition (37) becomes true after a certain number of loops.  $\square$

*Remark 1.* In the proof of Theorem 3, we have assumed the terminating condition ( $Current == Old$ ), whilst in our implementation we use condition (37). Thus, whilst *Label\_Graph()* stops when (37) is verified, the corresponding deductive process might go on, and thus deduce a value slightly bigger than the one returned by the algorithm. Since we can reduce  $\epsilon$  at will, this “incompleteness” has no practical consequences.

## 5 Experimental Results

Both qualitative and quantitative algorithms have been implemented in Java and a series of tests were conducted on a Dell Inspiron 8100 laptop with a Pentium III CPU and 64 MB RAM (OS: GNU/Linux, kernel 2.4.7-10). The tests were intended to demonstrate the label propagation algorithms, also to collect some experimental results on how fast the algorithms converge.

In the following, we present the experimental results obtained applying the algorithms to the auto manufacturer goal graph introduced in Figure 1.

### 5.1 Experiments with the Qualitative Approach

The first set of experiments was carried out in order to demonstrate the qualitative label propagation algorithm. For each experiment, we assigned a set of labels to some of the goals and events of the auto manufacturer graph (Figure 1) and see their consequences for other nodes. The label propagation algorithm reached a steady state after at most five iterations.

<sup>4</sup> An infinite sequence  $a_n$  is Cauchy-convergent if and only if, for every  $\epsilon > 0$ , there exist an integer  $N$  s.t.  $|a_{n+1} - a_n| < \epsilon$  for every  $n \geq N$ .  $a_n$  is convergent if and only if it is Cauchy-convergent.

Table 3 reports the results of four different experiments (Experiments 1, 2, 3, and 4).

For each goal/event, the table shows the initial (Init) label assignment to the variables S and D, also their final (Fin) value after label propagation. For instance, in the first experiment, the initial assignment for the goal *expand markets* is  $S=P$  and  $D=N$ , while the final values for *increase return on investment (GM)* are  $S=N$  and  $D=P$ .

In the first experiment, we start with goals *lower environment impact*, *improve mileage* and *lower sales price* totally satisfied ( $S=F$ ), goals *expand markets* and *offer rebates* partially satisfied ( $S=P$ ), goals *increase foreign earnings* and *keep labour costs low* totally denied ( $D=F$ ), and goals *increase sales price* and *increase high margin sales* partially denied ( $D=P$ ). The propagation of such values produces a final state in which the main goal *increase return on investment (GM)* is partially denied.

In the second experiment we start with the same initial assignment of the first experiment, except that the event *Yen rises* is now totally satisfied. As reported in Table 3, this new assignment produces a different result for the root goal. Now, it is both partially satisfied and partially denied, namely we have a contradiction. Since the analysis is qualitative, it is not possible to distinguish between the two values  $S=P$  and  $D=P$  and we cannot say anything about this contradiction.

For the third experiment, we assume that goals *keep labour costs low* and *reduce raw material costs* are totally satisfied. This new assignment produce a contradiction for the root goal (totally satisfied and partially denied). Unlike from the previous experiment, we have now more evidence for the satisfaction of this goal, rather than its denial.

Finally, in the fourth experiment, though goals *increase Toyota sales* and *increase VW sales* are totally satisfied, for the root goal we still have the same contradictory result. Note that the new assignment also produces contradictory values for goals about GM, Toyota and VW sales ( $S=F$  and  $D=P$ ).

## 5.2 Experiments with the Quantitative Approach

In a second set of experiments we assigned numerical weights to “+”, “−” and “ $-_S$ ” lateral relationships as reported in Table 4. For instance, the goal *increase sales volume* contributes negatively to the goal *increase Toyota sales* with a weight 0.6, while the goal *increase car quality* contributes positively to the goal *increase customer loyalty* with weight 0.8. Table 5 reports the results of four different experiments.

The results of these experiments confirm those obtained with the qualitative algorithm. However, the numeric approach allows us to draw more precise conclusions about the final value of goals. This is particularly helpful for evaluating contradictions. For instance, in the second experiment (Exp 2), even though we have a contradiction for the final values of the root goal *increase return on investment (GM)* (i.e.,  $S=0.8$  and  $D=0.4$ ), we have more evidence for its satisfaction than its denial. With the qualitative approach we had  $S=P$  and  $D=P$ , and there



**Table 3.** Experiments with the qualitative approach

Goals/Events	Exp 1				Exp 2				Exp 3				Exp 4			
	Init		Fin		Init		Fin		Init		Fin		Init		Fin	
	S	D	S	D	S	D	S	D	S	D	S	D	S	D	S	D
increase return on investment (GM)	N	N	N	P	N	N	P	P	N	N	F	P	N	N	F	P
increase sales volume	N	N	F	P	N	N	F	P	N	N	F	P	N	N	F	P
increase profit per vehicle	N	N	N	P	N	N	P	P	N	N	F	N	N	N	F	N
increase customer appeal	N	N	F	N	N	N	F	N	N	N	F	N	N	N	F	N
expand markets	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N
increase sales price	N	P	N	F	N	P	N	F	N	P	N	F	N	P	N	F
increase foreign earnings	N	F	N	F	N	F	P	F	N	F	P	F	N	F	P	F
lower production costs	N	N	N	F	N	N	N	F	N	N	F	N	N	N	F	N
increase high margin sales	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P
reduce operating costs	N	N	F	N	N	N	F	N	N	N	F	N	N	N	F	N
lower environmental impact	F	N	F	N	F	N	F	N	F	N	F	N	F	N	F	N
lower purchase costs	N	N	F	N	N	N	F	N	N	N	F	N	N	N	F	N
keep labour costs low	N	F	N	F	N	F	N	F	F	N	F	N	F	N	F	N
improve economies of production	N	N	N	N	N	N	N	N	N	N	F	N	N	N	F	N
lower gas price	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
improve mileage	F	N	F	N	F	N	F	N	F	N	F	N	F	N	F	N
offer rebates	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N
lower loan interest rates	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
lower sales price	F	N	F	N	F	N	F	N	F	N	F	N	F	N	F	N
reduce raw materials costs	N	N	N	N	N	N	N	N	F	N	F	N	F	N	F	N
outsource units of production	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
gas price rises	N	N	N	N	N	N	N	P	N	N	N	P	N	N	N	P
lower Japanese interest rates	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
US gas price rises	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Japanese gas price rises	N	N	N	N	N	N	N	P	N	N	N	P	N	N	N	P
Yen rises	N	N	N	N	F	N	F	N	F	N	F	N	F	N	F	N
Japanese rates rise	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
improve car quality	N	N	P	N	N	N	P	N	N	N	N	P	N	N	N	P
improve car services	N	N	N	P	N	N	N	P	N	N	N	P	N	N	N	P
improve customer loyalty	N	N	P	P	N	N	P	P	N	N	P	P	N	N	P	P
increase Toyota sales	N	N	N	P	N	N	N	P	N	N	N	P	F	N	F	P
increase VW sales	N	N	N	N	N	N	N	N	N	N	N	N	F	N	F	P

was nothing else to say about this contradiction. Analogous comments apply for the fourth experiment.

The threshold  $\epsilon$  (equation 37) used in the experiments has been chosen as the smallest positive value of type float (i.e., `Float.MIN_VALUE`). With this threshold, the algorithm converged in five iterations for all four experiments.

**Table 4.** Quantitative relationships for the auto manufacturer example of Figure 2

Goal/Event	Relationship	Goal/Event
increase sales volume	$0.6-S$ $\mapsto$	increase Toyota sales
increase Toyota sales	$0.6-S$ $\mapsto$	increase VW sales
increase VW sales	$0.6-S$ $\mapsto$	increase sales volume
increase customer loyalty	$0.4+$ $\mapsto$	increase sales volume
increase sales prices	$0.5-$ $\mapsto$	increase customer loyalty
increase car quality	$0.8+$ $\mapsto$	increase customer loyalty
improve car services	$0.7+$ $\mapsto$	increase customer loyalty
lower environment impact	$0.4+$ $\mapsto$	increase customer loyalty
increase sales prices	$0.3+$ $\mapsto$	improve car services
keep labour costs low	$0.7-$ $\mapsto$	increase car quality
improve economies of production	$0.8+$ $\mapsto$	lower purchase costs
Yen rises	$0.8+$ $\mapsto$	increase foreign earnings
lower Japanese interest rates	$0.4+$ $\mapsto$	lower sales price
Japanese rates rises	$0.8-$ $\mapsto$	lower Japanese interest rates
Japanese rates rises	$0.6+$ $\mapsto$	Yen rises
Yen rises	$0.4-$ $\mapsto$	Japanese gas price rises
Japanese gas price rises	$0.6+$ $\mapsto$	gas price rises
US gas price rises	$0.6+$ $\mapsto$	gas price rises
gas price rises	$0.8-$ $\mapsto$	lower gas price

### 5.3 A Final Experiment

A final set of experiments were carried out with graphs with a bigger number of cycles. The goal of the experiments was to evaluate the growth of the number of iterations with respect to the growth of the number of cycles.

We generated such graphs randomly. Starting from the auto manufacturer model of Figure 1, we generated new graphs adding randomly new contribution links between goals and thereby generating new cycles. Of course, since we were only interested on the structure of the graphs, we have not associated to them any semantics. Figure 5 shows an example of such a random graph.

The results of the experiments showed that new cycles increase significantly the number of iterations. For instance, applying the quantitative algorithm to the graph showed in Figure 5, the number of iterations for the last three experiments presented in Table 5 (Exp2, Exp2, and Exp3) increased from 6 to 33, from 6 to 80 and from 5 to 24, respectively.

Finally, a set of experiments were carried out with graphs with a bigger number of nodes (from hundred to thousand nodes). The results showed that also for bigger graphs value propagation performs in negligible time ( $\leq 10^{-2}s$ ). This suggests us that our approach can be applied in real life applications where goals models can count more than hundred goals.

**Table 5.** Results with the quantitative approach

Goals/Events	Exp 1				Exp 2				Exp 3				Exp 4			
	Init		Fin		Init		Fin		Init		Fin		Init		Fin	
	S	D	S	D	S	D	S	D	S	D	S	D	S	D	S	D
increase return on investment (GM)	0	0	0	.4	0	0	.8	.4	0	0	.9	.2	0	0	.9	.6
increase sales volume	0	0	1	.1	0	0	1	.1	0	0	1	.2	0	0	1	.6
increase profit per vehicle	0	0	0	.4	0	0	.8	.4	0	0	.9	0	0	0	.9	0
increase customer appeal	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
expand markets	.3	0	.3	0	.3	0	.3	0	.3	0	.3	0	.3	0	.3	0
increase sales price	0	.5	0	.8	0	.5	0	.8	0	.5	0	.8	0	.5	0	.8
increase foreign earnings	0	.9	0	.9	0	.9	.8	.9	0	.9	.8	.9	0	.9	.8	.9
lower production costs	0	0	0	.9	0	0	0	.9	0	0	.6	0	0	0	.6	0
increase high margin sales	0	.6	0	.6	0	.6	0	.6	0	.6	0	.6	0	.6	0	.6
reduce operating costs	0	0	.8	0	0	0	.8	0	0	0	.8	0	0	0	.8	0
lower environmental impact	.9	0	.9	0	.9	0	.9	0	.9	0	.9	0	.9	0	.9	0
lower purchase costs	0	0	.9	0	0	0	.9	0	0	0	.9	0	0	0	.9	0
keep labour costs low	0	.9	0	.9	0	.9	0	.9	.9	0	.9	0	.9	0	.9	0
improve economies of production	0	0	0	0	0	0	0	0	0	0	.7	0	0	0	.7	0
improve mileage	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lower gas price	.8	0	.8	0	.8	0	.8	0	.8	0	.8	0	.8	0	.8	0
offer rebates	.3	0	.3	0	.3	0	.3	0	.3	0	.3	0	.3	0	.3	0
lower loan interest rates	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lower sales price	.8	0	.8	0	.8	0	.8	0	.8	0	.8	0	.8	0	.8	0
reduce raw materials costs	0	0	0	0	0	0	0	0	.7	0	.7	0	.7	0	.7	0
outsource units of production	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
gas price rises	0	0	0	0	0	0	0	.2	0	0	0	.2	0	0	0	.2
lower Japanese interest rates	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
US gas price rises	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Japanese gas price rises	0	0	0	0	0	0	0	.4	0	0	0	.4	0	0	0	.4
Yen rises	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0
Japanese rates rise	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
improve car quality	0	0	.6	0	0	0	.6	0	0	0	0	.6	0	0	0	.6
improve car services	0	0	0	.2	0	0	0	.2	0	0	0	.2	0	0	0	.2
improve customer loyalty	0	0	.5	.2	0	0	.5	.2	0	0	.5	.2	0	0	.4	.5
increase Toyota sales	0	0	0	.6	0	0	0	.6	0	0	0	.6	1	0	1	.6
increase VW sales	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	.6

## 6 Conclusions

We have presented a formal framework for reasoning with goal models. Our goal models use AND/OR goal relationships, but also allow more qualitative relationships, as well as contradictory situations. A precise semantics has been given for all goal relationships which comes in a qualitative and a numerical form. Moreover, we have presented label propagation algorithms for both the qualitative and the numerical case that are shown to be sound and complete with respect to their respective axiomatization. Finally, the paper reports experimental results on the label propagation algorithms that suggest that the algorithms are scalable and can work with goal models having hundreds of nodes.

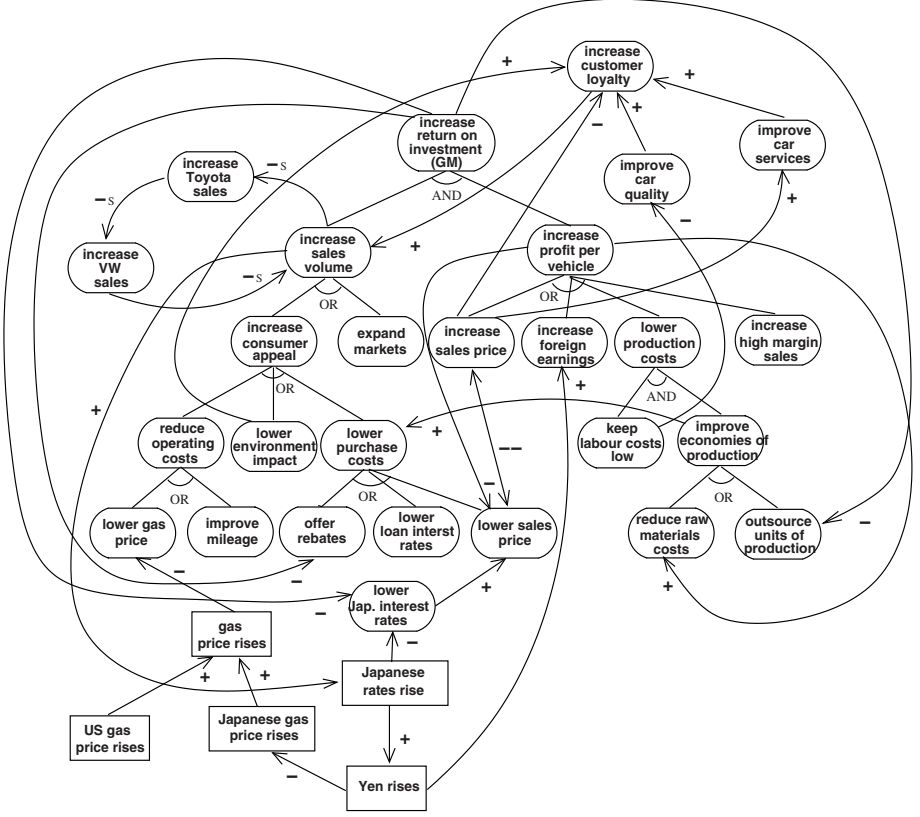


Fig. 5. A partial goal model for GM, with added cycles.

The main limitation of our framework is about the definition of contribution links and the labels assignment. This is a problem that an analyst can encounter in modeling complex situations where many goals are related one another and different perspectives have to be considered in assigning initial values to the goals. We are currently working on a methodology that can help and drive an analyst in building goal models considering different point of views and discovering relationships between goals.

Another important limitation of our approach is that we deal with conflicts but we do not resolve them. Also here, we are currently working to top-down approach that should allow us to find a set of possible assignments that can guarantee the satisfaction (or the denial) of specific goals without any conflict.

Future research directions include applying different techniques, such as Dempster Shafer theory [15], to take into consideration the sources of information in the propagation algorithms. This will allow us to consider, for instance, the reliability and/or the competence of a source of evidence and then model and analyze different scenarios [14, 13]. We also propose to apply our framework to real case studies to confirm its validity.

## References

1. A. I. Anton. Goal based requirements analysis. In *Proceedings of the 2nd International Conference on Requirements Engineering ICRE'96*, pages 136–144, 1996.
2. A. I. Anton and C. Potts. The use of goals to surface requirements for evolving systems. In *Proceedings of the International Conference on Software Engineering (ICSE '98)*, pages 157–166, Kyoto, Japan, April 1998.
3. A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1-2), 1997.
4. B. Boehm. Identify quality-requirements conflicts. In *Proceedings of the 2nd International Conference on Requirements Engineering ICRE'96*, Colorado spring, Colorado, 1996.
5. A. Bundy, F. Giunchiglia, R. Sebastiani, and T. Walsh. Calculating Criticalities. *Artificial Intelligence*, 88(1-2), December 1996.
6. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1–2):3–50, 1993.
7. R. Jarvis, G. McArthur, J. Mylopoulos, P. Rodriguez-Gianolli, and S. Zhou. Semantic Models for Knowledge Management. In *Proc. of the Second International Conference on Web Information Systems Engineering (WISE'01)*, 2001.
8. H. Kaindl. A design process based on a model combining scenarios with goals and functions. *IEEE Transactions on Systems, Man and Cybernetic*, 30(5):537–551, September 2000.
9. H. Kautz, D. McAllester, and B. Selman. Encoding Plans in Propositional Logic. In *Proceedings International Conference on Knowledge Representation and Reasoning*. AAAI Press, 1996.
10. J. Mylopoulos, L. Chung, and B. Nixon. Representing and Using Non-Functional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 6(18):483–497, June 1992.
11. A. Newell and H. Simon. GPS: A Program that Simulates Human Thought. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw Hill.
12. N. Nilsson. *Problem Solving Methods in Artificial Intelligence*. McGraw Hill, 1971.
13. C. Rolland, Grosz, and R. Kla. Experience with goal-scenario coupling. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering*, Limerik, Ireland, 1999.
14. C. Rolland, C. Souveyet, and C. Ben Achour. Guiding goal modelling using scenarios. *IEEE Transactions on Software Engineering*, 24(12), December 1998.
15. G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press., 1976.
16. A. v. Lamsweerde, R. Darimont, and E. Letier. Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Engineering*, 24(11):908–926, November 1998.
17. Daniel S. Weld. Recent Advances in AI Planning. *AI Magazine*, 20(2):93–123, 1999.

# Attribute-Based Semantic Reconciliation of Multiple Data Sources

Jeffrey Parsons<sup>1</sup> and Yair Wand<sup>2</sup>

<sup>1</sup>Faculty of Business Administration, Memorial University of Newfoundland,  
St. John's, NL, A1B 3X5, Canada  
jeffreyp@mun.ca

<sup>2</sup>Sauder School of Business, The University of British Columbia,  
Vancouver, BC, V6T 1Z2, Canada  
yair.wand@ubc.ca

**Abstract.** The main challenge in information integration is reconciling data semantics. Semantic reconciliation usually follows one of two approaches. Schema-based approaches assume instances belong to types and use schema information to reconcile types before reconciling attributes and relationships of these entity types. Attribute-based approaches use attribute names and non-semantic information, such as ranges of attribute values, to seek similarities in attributes across sources. We suggest an attribute-based approach that uses semantic information. Two principles guide our method. First, reconciliation does not require that instances belong to specific types. Instead, sources can be reconciled by analyzing properties. Second, properties that appear different may be manifestations of the same higher-level property, enabling the treatment of inter-source attributes, with different names and value ranges, as specializations of a more general attribute. We discuss the approach, analyze its potential advantages, suggest a formalization, demonstrate its use with examples, and suggest directions for further research.

## 1 Introduction

The need to reconcile the meaning of data derived from multiple sources has grown with the proliferation of data sources used by organizations and individuals. Since different sources are usually developed independently, and often for different purposes, integration typically requires resolving the semantics of the data in the separate sources.

Several phenomena have contributed to growing needs in this area. First, the widespread diffusion of database technology has led to a proliferation of independent information sources. As organizations develop databases for specific applications, often information about the same (kinds of) entities is distributed among independent sources. Consequently it is necessary to combine or integrate data from these sources.

Second, the growing tendency of organizations to integrate their business processes by enabling their information systems to communicate and exchange data has created the need to reconcile the semantics of data in the communicating systems.

It is reasonable to expect that problems related to reconciling the meaning of data are even more difficult across organizations than within organizations.

Third, the Internet has made available a wealth of sources that can be accessed easily, but that were usually created completely independently and for different purposes. Reconciling data from sources across the Internet creates an additional difficulty. When data reside in structured databases, much of the semantics of data is carried in the conceptual schemas of the sources. However, sources on the Internet often contain information in the form of semi-structured data, for which the underlying schema may only be partially or implicitly specified.

The importance of integrating and reconciling data from different sources has led to a great deal of work addressing this issue. A recent survey by Rahm and Bernstein [23] offers a taxonomy of schema matching approaches, and classifies a variety of previous research according to the taxonomy. However, there has been limited success in this area. According to Smith and Obrst, while there has been progress in addressing “syntactical and structural aspects of integration... no similar infrastructure is in place to help address [the] challenge of semantic reconciliation” [26, p. 26].

We suggest that a root cause underlying the apparent lack of progress on semantic integration is the absence of a strong theoretical foundation for understanding what it means to combine/integrate/reconcile data from independent sources.

Common methods for reconciliation can be broadly categorized into schema-based and attribute-based approaches [23]. Schema-based approaches employ information embedded in the schema to identify semantic relationships in data. Such approaches are suitable for structured databases and are commonly used in the context of schema integration and schema mapping. Attribute-based approaches seek to identify similarity among attributes in different sources based on attribute characteristics such as names and ranges of values. According to Clifton et al: “Perhaps the most basic problem in database integration is determining attribute correspondences: what attributes (e.g., columns in a relational table) in two databases reflect the same kind of real-world information” [7, p. 1]. Proposed attribute-based approaches tend to be statistical or structural in nature, however, rather than based on semantics. Statistical methods look at factors like the frequency of specific values of attributes across sources, while structural approaches look at factors like the defined structure (e.g., name, data type, integrity constraints) of attributes<sup>1</sup>. Such approaches can be used for structured (perhaps in combination with schema-based approaches) and semi-structured sources.

Schema-based approaches rely on two fundamental assumptions. First, they assume that records in the sources include information about instances of entity types or classes (e.g., ‘customer’). This means that class reconciliation is a precondition to reconciling data about instances. Second, they assume that data elements in a source can be mapped into a well-defined semantic data model. Semantic models typically employ constructs such as entities, relationships, and attributes. Thus, prior to reconciling specific data elements (or groups of elements), one would need to map them into semantic constructs. However, it is well known that the same piece of

---

<sup>1</sup> Some of this attribute information will be stored with the schema in a structured database.

information might be represented using different semantic constructs (e.g. an entity type or an attribute). Thus, the need to resolve construct mismatches adds a level of complexity to the reconciliation process, even before one can explore similarities in meanings among the various sources.

For several reasons, we contend that a more fruitful approach to semantic reconciliation is to focus first on reconciling the meaning of attributes. First, attributes are more basic constructs than entity types or classes. For example, ‘is red’ can be an attribute of many different types of entities, but retains a common meaning across types. Second, class or entity-type reconciliation depends on the attributes defining these classes. Third, the attribute construct is used in all semantic models. Therefore, methods to reconcile attributes can be independent of specific semantic models. Despite these advantages, we contend that common attribute-based approaches are limited in their ability to reconcile semantics, as they are typically statistical or structural, rather than based on semantic information.

In this article, we suggest a theoretical foundation for semantic-based reconciliation at the attribute level. Based on this foundation we describe an approach to attribute-based reconciliation, and indicate the potential usefulness of the approach by demonstrating how one can apply it to query data distributed over heterogeneous sources and to form a generalized view of data in different sources.

## 2 Semantic Interoperability in Heterogeneous Systems

Research on semantic interoperability addresses the practical need to combine information from several independent sources that contain information about a common underlying subject domain, but differ in the way they represent the semantics of that domain. Different sources might differ on implementation technologies, underlying data models, the approaches used to map the subject domain into the data structures, or on their views of the subject domain.

As mentioned above, the challenges of semantic interoperability have received renewed attention with the growth and evolution of the Internet. In that environment, sources typically are independent of each other and tend to lack the degree of structure found in traditional databases.

In this section, we review research on semantic interoperability in both these domains.

### 2.1 Semantic Interoperability in Databases

Traditionally, organizations have maintained their data in highly structured repositories that provide logical structures to facilitate transaction processing and/or query operations. The hierarchical, network, and relational models of data reflect an evolution of logical data structures that offer increasing flexibility in accessing and using data resources.

The independent development of applications to serve different needs inevitably leads to semantic heterogeneity. There are three main sources of difficulties in reconciling the semantics of independent data sources. First, information sources may



be based on different data models. For example, information from a data source based on the relational model may need to be combined with information from a source based on an object-oriented model. Second, information sources may be based on the same data model, with each source containing different semantics reflecting the views and needs of the users for whom the source was developed. Third, the data may lack a well-defined schema based on a data model. For example, an XML document may include untagged textual information.

Most of the existing work on semantic reconciliation has been in the area of traditional structured databases. Early work addressed schema integration and focused on the development of a unified global schema [1], [24]. Such an approach is termed *tightly coupled*. The practical difficulty of developing global schemas and the need to preserve component database autonomy led to *loosely coupled* multidatabase approaches that map component databases onto a canonical data model [14]. In addition to approaches that focus on data model structure, additional work has focused on: identifying types of semantic conflict and suggesting mechanisms for reconciling differences [18], [20]; using the notion of *semantic values* to provide a context for reconciliation [25]; reconciling information represented as data or as schema in different sources [16]; using statistical and other approaches to measure affinity or similarity to resolve heterogeneity [4], [5], [8], [11]; using semantic networks to detect and resolve heterogeneities among component databases [11]; and separating heterogeneity in the semantics of data from heterogeneity in the representation of data [13]. Recently, a number of projects have added AI techniques to facilitate reconciliation, including: SemInt (neural networks) [13]; LSD (machine learning) [9]; and Clio (data mining, coupled with user input) [17]. One large-scale application was work done on US Air Force flight operations databases by the MITRE Corporation [7], which focused on attribute matching and used similarities between data element names and statistics about data element values.

## 2.2 Semantic Interoperability in Semi-structured Sources

Research to support semantic heterogeneity in semi-structured sources originated in digital libraries [19], addressing such issues as developing query languages to search documents containing some structured elements [6]. More recent work has largely focused on resolving heterogeneity by developing and using specific domain ontologies onto which document concepts can be mapped [10], [15], [27]. In addition, the construction of domain ontologies can be driven by structural and statistical analysis [15], in much the same way such approaches have been used in structured databases.

## 2.3 Observations about Semantics

In prior work on semantic integration, three basic approaches have been used. In traditional database integration, approaches that address semantics typically use schema (class or entity type) information. Approaches that use attributes typically rely on statistical and structural similarity, and do not address the actual semantics. In

work on the integration of semi-structured sources, the notion of domain ontologies (which may be constructed on the basis of statistical and structural analysis of similarity) has been used to deduce semantic similarities.

Our objective in this paper is to suggest a foundation for attribute similarity based on the notion of generalization of attributes. This approach uses semantic similarity, but does not require complete domain ontologies.

In the remainder of this paper, discussion is limited to resolving semantic heterogeneity in structured sources; however, we believe the approach can also be applied to support interoperability in semi-structured sources, since it focuses on the semantics of data elements.

### 3 A Framework for Understanding Semantic Reconciliation

We begin by identifying three generic kinds of situations that arise when interoperating multiple sources. First, the same instance (or thing) can be represented in different sources, and there might be a need to combine the data in the sources that pertain to the instance. For example, data about a specific student might be split across an academic records database and a tuition payments database. We term this *instance reconciliation*. Second, different instances or things considered “similar” for the purpose of a specific application might be represented in the different sources, and there might be a need to combine data about these different instances for a higher-level analysis. For example, when two banks merge, they may need to combine their customer databases to continue to serve their (now larger) customer base,<sup>2</sup> or to generate summary reports about their overall customer base. We term this *instance integration*. Third, a source or sources may contain information about the same instances using different formats or representations spanning different periods of time, and there might be a need to show the history of the instance over time. For example, to create an integrated medical history for patients, it will be necessary to combine information about those patients across doctor and hospital visits that span long periods of time. We term this *instance history*. Common to all three of these cases is that the available data may reflect different views of a domain, but refer to the same instances or to instances that are considered the “same” in some sense. Of course, more than one of the three cases may apply in a given application.

Within this context, semantic reconciliation and integration involve mapping and matching the meaning of terms in different sources. Sources might be either structured or semi-structured. Structured sources conform to a common structure defined outside of the data (by a *schema*). Semi-structured sources do not conform to a common structure and typically include information about data items within the data. Business databases are usually structured, while data available over the Internet are often semi-structured.

The need to reconcile meaning of terms among schemas is not unique to semantic reconciliation across sources. It arises in single-source design in two ways. First, the

---

<sup>2</sup> The instance integration problem may also include instance reconciliation issues. For example, the same person may have been a customer of two banks before they merge.

problem of *view* or *schema integration* [1] involves reconciling meaning of elements comprising the different schemas defined over the same data instances. Schema integration aims at identifying a common schema. Second, the schema used for a source might change over time, leading to issues of *schema evolution* of one source. In contrast, semantic reconciliation refers to reconciling the meaning of data residing in different sources. Furthermore, it does not necessarily seek to define a common schema.

The above discussion has identified various cases where semantic reconciliation might be needed. These cases can be classified according to: 1) whether the data are structured (defined by a schema) or not; 2) whether one deals with one or several sources; and 3) whether changes of representation might occur over time. Table 1 summarizes the various cases according to these dimensions. As indicated earlier (and illustrated by the example of medical data reconciliation), a practical situation might include several of these cases.

**Table 1.** Classification of various cases of semantic reconciliation

Case	Structure of data	Multiplicity of Sources	Time spanned	Semantics contained in	Goal of semantic reconciliation
View Integration	structured	single	same	schema	One common schema
Schema evolution	structured	single	varies	schema	Translation of older to newer schema
Instance reconciliation	structured	multiple	same	instance	Interoperate data about an instance
Instance reconciliation	semi-structured	multiple	same	instance	Interoperate data about an instance
Instance integration	structured	multiple	same	multiple instances	Interoperate data about similar instances
Instance integration	semi-structured	multiple	same	multiple instances	Interoperate data about similar instances
Instance history	structured	single or multiple	varies	instance	Use data about an instance over time
Instance history	semi-structured	single or multiple	varies	same instances	Use data about an instance over time

In what follows, we are interested in reconciling semantics over multiple sources or the same source over time, without necessarily having to create one common schema.

## 4 The Root Problem and a Candidate Resolution

In this section, we argue that problems of data reconciliation can be alleviated by addressing the semantics of attributes independent of higher-level constructs such as classes (or entity types).

We begin our analysis of the problem and our proposed solution by first observing that using schema information in structured sources typically involves two types of *commitment*, each of which has to be addressed in semantic reconciliation. First, various sources might employ different data models based on different representation constructs (e.g. the relational model versus some object-oriented models). Furthermore, even if the same data model is used, the same concept might be represented using different semantic constructs. For example, Figure 1 shows two ER diagram segments depicting an order as an entity type and a relationship, respectively. Associating a concept with specific semantic construct (or constructs) within a data model or across data models can be termed *construct commitment*.<sup>3</sup> It requires the resolution of *structural* heterogeneity (related to representation) prior to the consideration of semantic heterogeneity (related to meaning).

Second, different sources might impose different views of the domain being represented. In particular, sources might differ in the choice of classes or types into which entities or objects are organized. Such models incorporate a *class commitment*, in which information about individual entities (or objects) is tied to the entity types (or classes) to which the entities (or objects) belong. Typically, different classes will be named differently and defined by different attributes. For example, the same item, manufactured by a firm, can be represented in different databases under different entity types. In one source – related to manufacturing – as a product with product specification and manufacturing-related data. In the other – related to marketing – the same specifications might appear in user-oriented terms; there will be no manufacturing data, but rather cost and “consumer feature” information. A third source might be related to order fulfillment, and include information about the dimensions and weight of the product.

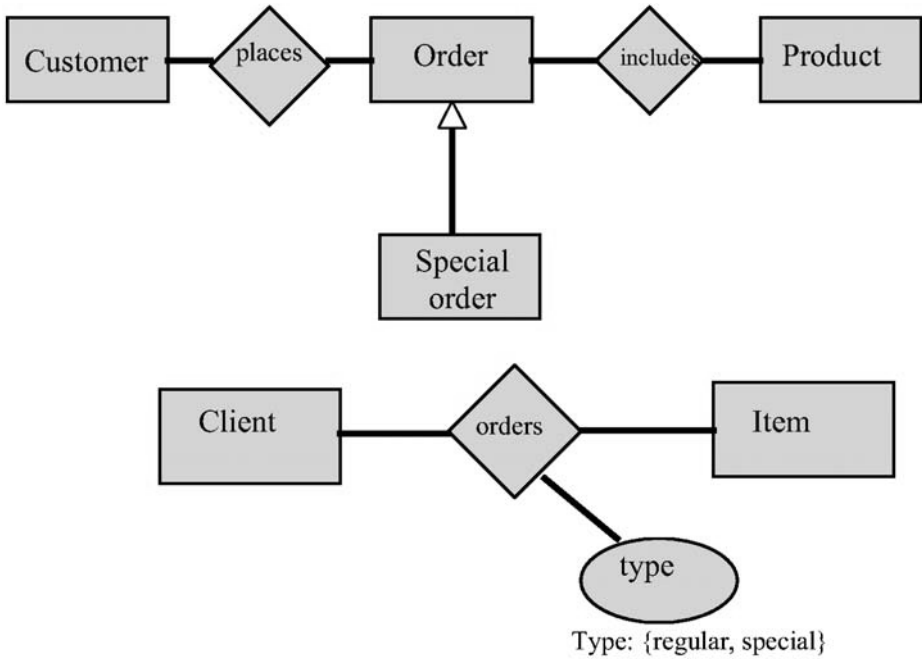
Anchoring representation to classes is widely practiced, and reflects the way humans relate to things in their environment as instances of concepts. However, “binding” instances to classes can create numerous problems in information modeling and database operations, even within a single source database [21]. A possible solution is to separate the way instances and properties are represented in the database from any specific classification. Classes can then be defined as a second “layer” above the instances and their properties [21].

Both types of commitment exist in all cases where the semantics is represented in a schema (see Table 1). In contrast, in semi-structured sources the construct commitment problem does not necessarily exist.

Beyond single source issues of schema integration, the need to reconcile classes is a major issue in reconciling meanings across sources. Similar to the case of a single source database, we claim that semantic integration can be facilitated by eliminating the problems resulting from binding instances to classes. However, if we do not use

---

<sup>3</sup> It is possible the same construct will be used in different models.



**Fig. 1.** Two ER diagrams for the same data

classification, we might lose some semantic information. We believe, however, that we can capture and use semantic information at the attribute level. Our approach to attribute-based semantic reconciliation is based on three observations.

**Observation 1.** Irrespective of the problems of data model commitment and class commitment, what is common to databases is that they represent information about individual entities or things [21]. This information can reflect either the state of entities (synchronous information) or changes to this state (diachronous information). In either case, information is stored as *data items* reflecting *attributes* of entities.

**Observation 2.** Data attributes that appear to be different in different sources may represent the same concept at a more generic level.

**Observation 3.** When data models exhibit class commitment, designers use attributes to identify similarity of entities. Entity types are represented in terms of attributes common to a group of entities.

Observation 2 is the key to our approach. To demonstrate it, we use observation 1 – that attributes reflect information about entities. Consider the following two examples. First, in a database describing animals, an “entity” that can run and an entity that can hop both move on land. Hence, both ‘to hop’ and ‘to run’ mean ‘to move on land’ at a more generic level. Second, in two independent sources, the attributes ‘customer number’ in one and ‘telephone number’ in the other may be used to realize the more general concept of an entity ‘identifier’. Note that in both

examples, there is no need to know the “type” to which instances possessing the attribute belong.

Once we have used observations 1 and 2 to identify generic properties, we can use observation 3 as the basis for viewing instances having different attributes (that are generically similar) as being in the same class or type.

We contend that these three observations provide a basis for resolving semantic heterogeneity both in structured databases and in semi-structured information sources. In particular, they point to the importance of attributes for semantic reconciliation. Furthermore, since virtually all data models include attributes as a fundamental construct, focusing on attributes can also serve to alleviate the data model commitment problem mentioned above.

## 5 Property Precedence as a Basis for Semantic Reconciliation

### 5.1 Foundations

In order to reconcile and integrate independent information sources, the semantics or meaning of the data in each source must be established. The meaning of data in a database can only be established in terms of what the data represent. On the assumption that information systems represent domains that exist (or might exist) in the world, we turn to ontology, which deals with concepts for describing “what is out there” – in particular, things and their properties. Since we want our approach to be general (independent of any specific domain), we use a generalized ontology of concepts (as opposed a specific domain ontology for some subject matter). We have chosen to use Bunge’s ontology as adapted by Wand and Weber to information systems concepts [2], [3], [30], [32]. This ontology has been used to analyze and gain insights into a variety of information systems and database concepts [28], [29], [30], [31], [32]. Most relevant to the current analysis, it was used to propose a data model in which instances exist separate of classes [21].

In this subsection we present the ontological basis for our approach. In presenting the ontological constructs and premises we follow the discussion in Parsons and Wand [21]. Since we are interested in reconciling information sources, we use our observations about the importance of attributes as a guide for choosing the relevant ontological concepts. Nevertheless, the concepts are ontological and refer to “real world” domains, not to information sources.

**Postulate\*:**<sup>4</sup> The world is made of *things* that possess *properties*.

This postulate can be viewed as the underlying ontological justification for the above Observation 1 (Section 4): namely, regardless of data model, databases contain information about the properties of things. We assume databases represent human perceptions of existing or possible worlds; hence things may have a real existence or

---

<sup>4</sup> Postulates and definitions adapted from Bunge’s model are indicated with \*.

may be perceived. For example, both an existing and a planned product are considered things.

**Principle\*:** There are no things without properties and *properties are always attached to things* [2, pp. 36, 58, 62].

Based on the above Observation 3 (Section 4) – namely, that attributes provide the basis for determining the similarity of things in establishing entity types or classes – when organizing information there is a special interest in identifying which things possess which properties.

**Definition\*:** The *scope* of a property,  $P$ , is the set of things possessing the property, denoted  $\text{Scope}(P)$ .

**Definition\*:** The *form* of a thing,  $x$ , is the set of properties possessed by the thing, denoted  $\text{Form}(x)$ .

Based on the above ontological notions and Observation 1 (Section 4), we posit the following:

**Premise:** The attributes of databases represent properties of individual things at a given time (synchronous data) or changes to these properties (diachronous data, events).<sup>5</sup> For example, in a customer account database, the customer's address and balance reflect properties at a given time, while transactions reflect changes to some properties.

The key to our approach to semantic reconciliation is Observation 2 above (Section 4): namely, attributes that appear to be different at one level can be regarded as the same at a more abstract level. This idea is formalized via the notion of *property precedence*.

**Definition\*:** Let  $P_1$  and  $P_2$  designate two properties.  $P_1$  will be said to *precede*  $P_2$  iff for every thing  $x$  possessing  $P_2$ ,  $x$  also possesses  $P_1$ .

It follows from the definition that  $P_1$  precedes  $P_2$  if and only if the  $\text{Scope}(P_2) \subseteq \text{Scope}(P_1)$ .

For example, the property 'move on land' precedes the properties 'walk' and 'run' since every thing that can walk and every thing that can run can move on land. The set of things that run is a subset of the set of things that can move on land.

---

<sup>5</sup> We note that attributes might not represent a specific property directly. In particular, identifying fields (e.g. customer number) might be artifacts of the information system. However, in their deep meaning such identifiers indicate uniqueness. Bunge's ontology guarantees uniqueness by the statement that no two things possess exactly the same properties. Thus, an identifier can be viewed as a "shorthand" representation of the additional properties to make each thing unique.

**Definition:** Let  $P$  denote a set of properties. A *preceding property* of a property  $P$  in  $P$  is a property that must be possessed by any instance possessing  $P$ .<sup>6</sup>

The following are examples of precedence:

1. To ‘propel by rowing’ is preceded by to ‘propel in water.’
2. To ‘be red’ is preceded by to ‘have colour.’

Clearly, if  $Q$  precedes  $P$  then  $\text{Scope}(Q) \supseteq \text{Scope}(P)$ . This leads us to the following.

**Definition:** Let  $\wp(P)$  denote the power set (set of all subsets) of  $P$ . The *preceding properties* of  $P$  in  $P$  are defined by the (set) function Preceding:  $P \rightarrow \wp(P)$ , such that  $\text{Preceding}(P) = \{Q \in P \mid \text{Scope}(Q) \supseteq \text{Scope}(P)\}$ .

**Definition:** The *preceded properties* of a property  $P$  are all properties for which  $P$  is a preceding property. That is, the function Preceded:  $P \rightarrow \wp(P)$  such that  $\text{Preceded}(P) = \{Q \in P \mid \text{Scope}(P) \supseteq \text{Scope}(Q)\}$ .

Recall our Observation 3 (Section 4) that a class (or type) in an information source is represented in terms of a set of attributes. All instances of a class possess these attributes. Precedence has special importance in the context of the relationship between properties and classes (or entity types). Often, classes are described by some *generic* properties, while the instances possess *specific* properties implying the generic ones. We say that the specific property *manifests* the generic one. Different instances might possess different properties preceded by the same generic property.

We formalize the above notion as follows.

**Definition:** A *manifestation* of  $P$  in an instance  $x$  is a property of  $x$  preceded by  $P$ . The manifestation of  $P$  in  $x$  is all properties of  $x$  preceded by  $P$ :  $\text{Manifest}(P, x) = \text{Preceded}(P) \cap \text{Form}(x)$ .

We indicate manifestation in the following way: if  $P$  is a generic property, and  $Q$  is a manifestation of  $P$ , we denote this by  $(P, Q)$ .

We demonstrate manifestation with two examples. First, in an animal database ‘has four legs’ can be represented as (‘has legs’, 4), and ‘being able to propel by crawling’ can be represented as (‘propel on land’, ‘crawl’). Second, consider a frequent flyer program, where a person can be considered of a “preferred status” after flying a certain “high” mileage, or a certain “large” number of flight segments (each of which can be operationalized by a specific integer). In this case, the following are possible: (‘has high status’, ‘high mileage’), (‘has high status’, ‘large number of flight segments’), and (‘has high status’, {‘high mileage’, ‘large number of flight segments’}). Note, in the last case the manifestation is a set.

---

<sup>6</sup> If property  $Q$  is a preceding property of property  $P$ , we say that  $P$  is *preceded by*  $Q$ , or  $Q$  *precedes*  $P$ .



## 5.2 Using Precedences and Manifestation to Convey Semantic Similarity

The notions of precedence and manifestation enable us to identify semantic similarity between different properties. We observe that there are two main types of manifestation. First, a given generic property can be manifested by a specific value (termed *value manifestation*). For example, having a given weight is preceded by the property ‘has weight.’ Each instance having a specific weight value can be a member of a class that includes in its definition ‘has weight.’ In this case, the value has no meaning unless “attached” to the generic property (consider the example of weight). Second, the generic property can be specialized in different ways where the individual cases do have meaning independent of the more generic property. We term this case *specialization manifestation*. For example ‘moves on land’ may be manifested as ‘crawls,’ ‘walks,’ ‘hops,’ or ‘runs.’ Animals propelling in one of these ways can be members of a class that includes in its definition the property ‘moves on land.’

A specialization manifestation can lead to viewing properties that have different meaning as “values” of a more generic property. For example, ‘run’, ‘walk’ and ‘hop’ can be considered “values” of ‘to propel on land’. When all manifestations are taken from the same domain, defined independently of the generic property, the specific property can be considered the *value* of the generic property P. Thus, manifestation can be considered a *generalization of the notion of value* of a given (generic) property. This generalization enables a representation of the specific properties of an instance in terms of generic (preceding) properties of classes.

Viewing properties that are manifestations of a more generic property as “similar” enables us to consider attributes with different domains of values as being similar. For example, consider the properties above – “run” and “hop” – they are usually attributed to living creatures - thus can be viewed as “values” of “animal ways of moving on land”. Consider now the properties “to be self-propelled by wheels” and “to be self-propelled on tracks”. Both can be viewed as “values” of “self-propelled on land”. The two more generic properties can be further generalized to “move on land”. Note that the generalization is done without specific reference to the “type” of things that move. However, underlying the whole analysis is the assumption that every property describes a “thing.”

The idea of manifestation also enables us to extend what would be considered instances of a class. As noted, classes are often defined by generic properties (e.g. ‘having legs’) while instances have specific properties (e.g. ‘having two legs’). The generalization of the notion of value can enable us to view instances that possess specific properties that have a common preceding property to be viewed as being in a class that has the common property in its definition. If we refer to the previous example, the class could be “things that can move” with one of its defining properties “medium of movement” (with values “land”, “water”, “air”).

The above discussion and examples demonstrate that property precedence provides a specific type of semantics that can be used to identify semantic similarities between attributes. Thus, we define:

**Definition:** The *precedence semantics* of a property P is the pair: (Properties preceding P, Properties preceded by P).

We end this subsection by pointing out that, since the notion of manifestation provides a way to view different properties that manifest the same property as similar, it will be the key to our approach for semantic reconciliation of attributes.

### 5.3 A Precedence Algebra

When reconciling two information sources, one source might contain more generalized properties than the other. For example, a marketing database might contain a field ‘age group’ while a demographic database might contain a field ‘age’ (likely represented as birth date). Similarly, in a university, one database can include the status of a student (graduate or undergraduate), while another database might specify the program of studies the student is in (which implies the status).

The following two lemmas establish some relationships between the observations that can be made about general properties and the observations that can be made about their manifestations. These lemmas can be used to explicate the precedence semantics that implicitly exists in a set of properties.

**Definition:** A property  $G$  is termed *fully manifested* by a set of preceded properties  $S$  if and only if every thing that possesses  $G$  possesses at least one of the properties in  $S$ .

For example, if there are only four ways for animals to propel on land (i.e.  $S$  is the set  $\{\text{‘crawl’}, \text{‘walk’}, \text{‘hop’}, \text{‘run’}\}$ ), then the property ‘propel on land’ ( $G$ ) is fully manifested by  $S$ . Note that, in a database context, full manifestation means that a null value implies “unknown”. If a property is not fully manifested, the instance still possesses the generic property, independent of whether it possesses a manifestation or not. In such cases, our approach allows for null values that mean “not applicable” in relation to manifestations. In terms of a database schema, an instance in a table would still possess the generic attribute defined by the column name. For instance, in the university example, it might mean the student is neither graduate nor undergraduate (such situations are possible).

The following lemma shows how precedence can be inferred between generic properties based on precedences between their manifestations.

**Lemma 1:** Let  $S_1$  be a set of preceded properties of a property  $G_1$ . If  $G_2$  is fully manifested by  $S_2$ , and for each property in  $S_2$  there is a preceding property in  $S_1$ , then  $G_1$  precedes  $G_2$ .

*Proof:* Since  $G_2$  is fully manifested, every thing possessing  $G_2$  also possesses a property in  $S_2$ . This property is preceded by a property in  $S_1$  that is a manifestation of  $G_1$ . Hence the thing possesses also  $G_1$ .

Lemma 1 is depicted in Figure 2.

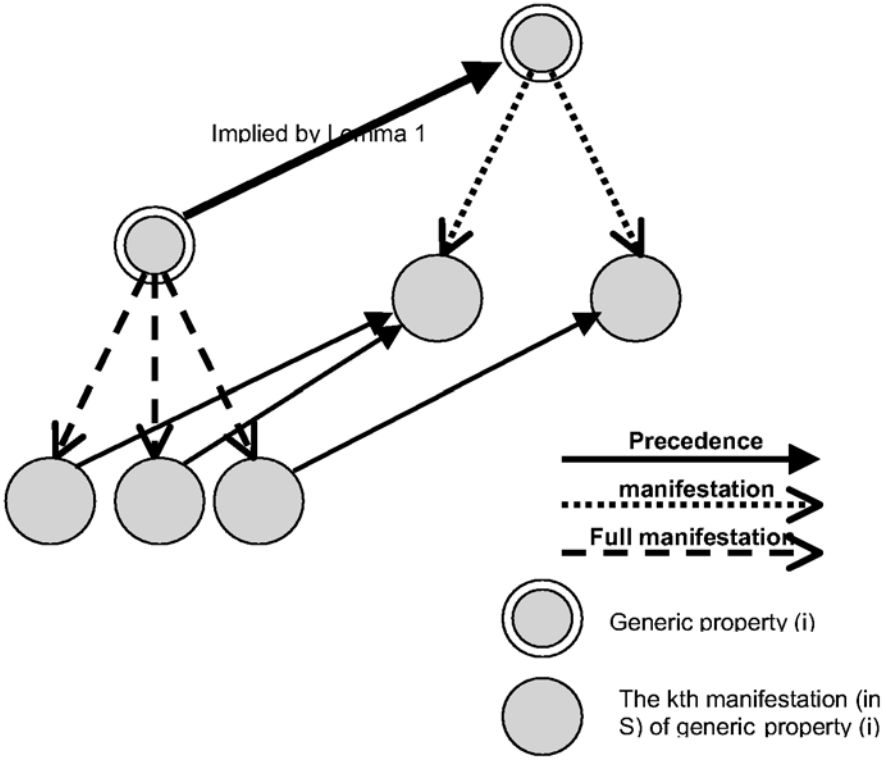


Fig. 2. Graphical representation of Lemma 1

This lemma enables the inference of precedence between general properties based on the relationships between manifestations. Consider two examples. First, given a marketing database containing the field ‘age group’ ( $G_1$ ) and a demographic database containing the field ‘age’ ( $G_2$ ), we may be interested in knowing if there is a precedence relation between ‘age group’ and ‘age.’ If there is a mapping from every age (value in  $S_2$ , e.g., 3) to a specific age group (value in  $S_1$ , e.g., ‘toddler’), then ‘age group’ precedes ‘age.’ Note that there generally will not be a mapping from ‘age group’ to ‘age.’ The second example refers to frequent flyer status ( $G_2$ ). Assume a certain airline has three status levels ( $S_2 = \{\text{‘high’, ‘medium’, ‘regular’}\}$ ). Assume each status is preceded by different types of privileges such as use of lounge, priority boarding, and class upgrade. “Having privileges” will be  $G_1$  and the various privileges the set  $S_1$ . Lemma 1 then implies that having a frequent status means having privileges. We can say that the semantics of being a frequent flyer is eligibility to receive some privileges.

The next lemma provides a condition under which inferences about manifestations can be made given precedences between generic properties.

**Lemma 2:** Let  $S_1$  be a set of preceded properties of a property  $G_1$ . Let  $G_1$  be a property that is fully manifested by  $S_1$ . If  $G_2$  is preceded by  $G_1$ , then each property in  $S_2$  is preceded by one or more properties in  $S_1$ .

*Proof:* Each property in  $S_2$  is preceded by  $G_2$ , hence by  $G_1$ .  $G_1$  is fully manifested, hence every thing possessing  $G_1$  possesses at least one property in  $S_1$ .

Lemma 2 is depicted in Figure 3.

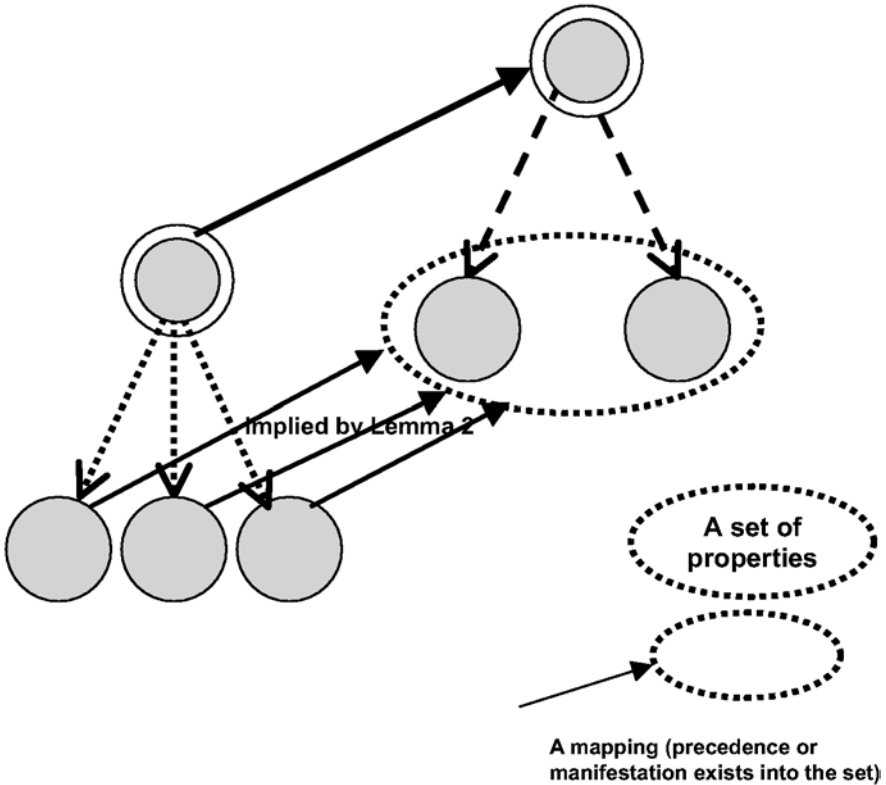


Fig. 3. Graphical representation of Lemma 2

Lemma 2 enables the inference that, in certain cases, if a thing possesses one of several values of a property, it must possess at least one of the manifestations of another property. For example, suppose ‘participating in an academic program’ ( $G_2$ ) is preceded by ‘being a student’ ( $G_1$ ). Suppose further that manifestations of  $G_1$  are  $S_1 = \{\text{‘undergraduate’ and ‘graduate’}\}$  and that manifestations of  $G_2$  are  $S_2 = \{\text{‘accounting’, ‘engineering’, ‘MBA’, ‘counseling’}\}$ . Lemma 2 enables the inference that a student in one of the programs must be (at least) a graduate or undergraduate.

In the examples, applying the two lemmas produces conclusions that might seem trivial when dealing with a single source. However, such conclusions become interesting when the information comes from several independent sources.

Consider the following example: Source 1 includes possible symptoms for different medical conditions. The generic property (common to all instances in the source) is ‘having a medical condition’. Source 2 contains data about patients who are undergoing different medical treatments or tests. The generic property (common to all instances in the source) is ‘being a patient’. Precedence exists between the two generic properties of the two sources in that to be a patient one must have a certain condition. If every condition has to be manifested by (possibly a subset of) symptoms, then from Lemma 2 we can infer a link exists between treatments or tests and groups of symptoms. Identifying the specific links can support interoperating the sources. The situation is depicted in Figure 4.

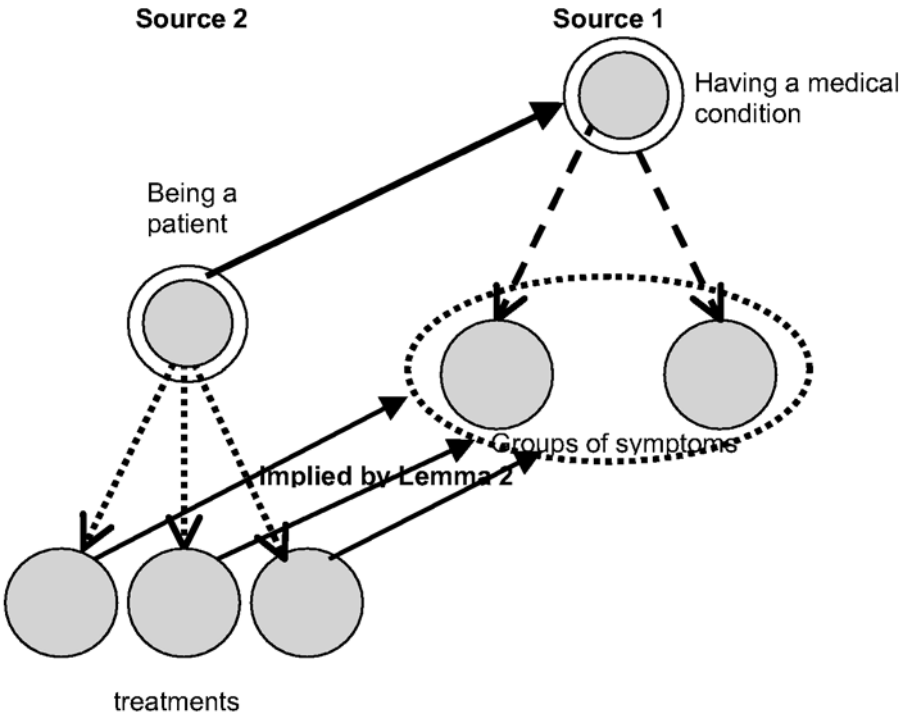


Fig. 4. Graphical representation of the medical data example

The precedence algebra can be useful in establishing semantic links between local properties of sources (source-dependent manifestations) across sources. While precedences among generic properties might be known across sources, manifestations might be source-specific. Links can exist between the values of two properties (each in a different source) indicating one of them is more general than the other. In that context, Lemma 1 can enable identification of a generic property across several sources that is not defined within the sources (the property will be  $G_1$ ) as was demonstrated by the examples above.

## 6 Using Precedence to Support Interoperability

In sub-section 5.2, we showed how the related concepts of precedence and manifestation provide a way to define common semantics of properties. In this section, we show how to use these concepts as a useful foundation for supporting the joint use of information from several sources. We proceed as follows. First, we show how to apply the notions of precedence and manifestation to data. We then demonstrate the use of the approach in two cases: (1) querying multiple sources, and (2) creating a common view over multiple sources.

As our purpose is to show the use of precedences, in what follows, we assume for now there is a way both to identify which properties, within and across sources, precede each other, and to identify common generic properties of properties in different sources. We return to this assumption in Section 7.

### 6.1 Operationalizing Semantic Similarity by Generalizing Properties

Consider an application that needs to use data from several sources having different schemas (or having no schema at all, but having well-defined attributes). Combining data from the various sources will require identifying data fields in the sources that have the same meaning. Before proceeding, note that a data field in a database often is composite, in the sense that it represents several meaningful facts, or in our terms, represents several properties (in the ontological sense). As well, sometimes several data fields need to be combined to create a meaningful fact representing a property. Thus, the mapping from data fields to properties in general is many-to-many. We assume in this exposition that all data fields have been “normalized” to represent a single property.

According to our assumptions, the data in each source describe either properties of things (instances), or changes to these properties. Thus, we focus our attention on identifying properties that are similar in some sense.

We propose that two properties,  $P_1$  and  $P_2$ , will be considered *similar* if there exists a property  $G$  that precedes both (thus,  $P_1$  and  $P_2$  will be manifestations of  $G$ ). The preceding property can be considered a generalization of the two properties. More formally, we define:

**Definition:** Let  $\{P_i: i=1, \dots, n\}$  be a set of properties where all are preceded by a property  $G$ . We will then term  $G$  a *generalization* of  $\{P_i: i=1, \dots, n\}$ .

Table 2 shows some examples for generalization of properties (again, assuming there is a way to identify precedences and generic properties).

We use the following notation. If  $P_i, i=1, \dots, n$  are similar in the sense that they are preceded by the same property,  $G$ , they will be represented in the joint application as:  $(G, \text{value}=P_i)$  or, simply by  $(G, P_i)$ .

Recall, the notion of value we are using is generalized, as it does not imply all values are taken from the same domain. For example, a graduate student can be defined in terms of having an advisor or in terms of being enrolled in some program. Using this approach, we can consider  $G$  as the common property represented in the

data, and the  $P_i$  as its manifestations. Furthermore, any  $P_i$  can, in turn, be further manifested by a value (for example, the specific program in which the student is enrolled).

**Definition:** Let  $P_i$  have a value  $v_i$  for  $i=1,\dots,n$ . Let  $G$  be a generalization of  $\{P_i: i=1,\dots,n\}$ . We will term  $v_i$  the *manifestation of  $G$  with respect to  $P_i$*  and denote it by  $(G, (P_i, v_i))$ .

Table 2. Examples of property generalization

Nature of property	Property 1	Property 2	Generic Property	Comments
Structural	Weight=80kg	Weight=70kg	‘Has weight’	The usual case of value
Structural	‘Enrolled in a program’	‘Registered in a course’	‘Is active student’	The two properties are not necessarily from the same domain
Behavioural	‘Walks’	‘Crawls’	‘Moves on land’	The two properties can be viewed as values
Behavioural	‘Moves on land’	‘Moves in water’	‘Moves’	The two properties can be viewed as values

In this notation,  $(P_i, v_i)$  is considered the “value” of  $G$ . It is possible to cascade this notation. For example, assume a student can be eligible for scholarship based on academic record or volunteer activity of certain types. The academic record can be assessed by the value of GPA or by instructor assessment. Then the following possibilities can exist:

- (scholarship-eligibility, ‘good academic record’ (GPA, ‘excellent’))
- (scholarship-eligibility, ‘good academic record’, (instructor’s recommendation, ‘good’))
- (scholarship-eligibility, ‘volunteer activity’ (type, ‘tutoring’)).

6.2 Querying Multiple Sources

In this sub-section, we demonstrate how queries can be processed over several sources using the property-precedence approach. We begin with an example. Assume there are two sources related to customer activity. In one source, customer activity is described in terms of value of goods purchased per year (value-of-goods). In the other source, customer activity is described in terms of number of purchase transactions per year (number-of-purchases). We will assign a common semantics to the two types of value by defining a generalized property termed ‘customer activity.’ One can query the two databases by asking: “what is the activity of customer x?” To describe the possible answers, we use the above notation. In the customer example, depending on the source, the answer for the above query can be given in two ways:

- From source 1: (activity, (value-of-goods, v))
- From source 2: (activity, (number-of-purchases, n)).

Furthermore, assume whether a customer is preferred or not depends on the level of the activity (where the level is defined according to the manifestation of activity in the source). If one wants to inquire about all preferred customers, then the clause “where customer-status = ‘preferred’” can be parsed as follows:

In source 1: ‘value-of-goods > ...’

In source 2: ‘number-of purchases > ...’.

Assume one wants to list the status for all customers. The status can be displayed with the complete list of manifestations, e.g. (‘preferred’, (value, v)). The list appearing to the right of a property can be termed the *evidence*, as it ‘explains’ why the specific case was retrieved by showing how a higher-level property is obtained via a precedence.

In summary, the examples above demonstrate how:

- 1) different data fields can be viewed as special cases (values) of a more generic attribute;
- 2) one can query multiple sources by using generic attributes; and
- 3) the responses to such queries might appear to the user of the multiple sources.

We now illustrate what could be the results of a query applied to two sources where one includes data with more specific properties. Let the data in source 1 represent values (manifestations) of the generic property  $G_1$  and the data in source 2 represent manifestations of the generic property  $G_2$ . Assume  $G_1$  precedes  $G_2$ . It is important to note that all instances of source 1 possess  $G_1$  and all instances of source 2 possess  $G_2$ . Consider the simple query: “List values of  $G_1$  based on both sources”. We denote specific manifestations (values) for  $G_i$  by  $S_i(k)$ .

To illustrate, we return to the student example mentioned earlier. In source 1, all instances are students ( $G_1$ ) and they can be at two levels  $S_1 = \{\text{‘undergraduate’}, \text{‘graduate’}\}$ . Source 2 contains the program of studies of a person ( $G_2$ ). It is known that only students can be assigned a program of studies. Let:  $S_2 = \{\text{‘accounting’}, \text{‘engineering’}, \text{‘MBA’}, \text{‘counseling’}\}$ . It is known that ‘MBA’ is a graduate program while ‘engineering’ can be either a graduate or undergraduate program.

The generic query could read: “list for each student whether they are graduate or undergraduate.”

Table 3 shows what the results might look like. For completeness, the table also includes the cases where the query cannot be answered, with the explanation of the missing information.

### 6.3 Creating a Common View over Different Sources

Consider the need to reconcile data from two sources. Assume source 1 includes records of people who made purchases, but that only some are considered active customers and have a customer number, implying having the property ‘has an account.’ Assume source 2 also includes records of people who have made purchases, but includes ‘time of last purchase.’ Further assume that in source 2, purchasers are considered active customers if time since last purchase is less than three months.



**Table 3.** Possible results of querying two sources

Source	Source contains	Value and evidence*	Explanation
1	A specific manifestation of $G_1$	$G_1=S_1(1)$	
example	The level of studies	Student is: 'undergraduate'	
1	No manifestation of $G_1$	$G_1=?$	The instance possesses $G_1$ but no specific "value" for it.
example	Level is not known	Student is: ?	Whether the student is graduate or undergraduate is unknown for this student.
2	A specific manifestation of $G_2$ The precedence from $S_2$ to $S_1$ is known	$G_1=(S_1(2),(G_2,S_2(3)))$	$G_1$ precedes $G_2$ and the manifestation $S_2(3)$ implies the "value" $S_1(2)$
example	The program of studies	Student is: ('graduate', (program, 'MBA'))	Being in the MBA program implies being a graduate student
2	An instance possess a specific manifestation of $G_2$ The precedence from $S_2$ to $S_1$ is not known	$G_1=(?, (G_2,S_2( )))$	$G_1$ precedes $G_2$ and the value in $S_1$ cannot be inferred from the manifestation $S_2( )$
example	Program is known	Student is: (? (program, 'engineering'))	This person attends a program, thus appears in source 2, however, it is not known if the program is a graduate or undergraduate one
2	A record contains no manifestation of $G_2$	$G_1=(? (G_2, ?))$	Because $G_1$ precedes $G_2$ it is known that the instance possesses $G_1$ , but the value in $S_1$ is not known because $S_2$ is not known.
example	No data for the specific program	Student is: (? (being in a program, ?))	This person attends a program, thus appears in source 2, however, the program is not known

\*  $S_i(k)$  designates the k-th value of the possible manifestations of property  $G_i$

If one now lists the properties of all people who made purchases, one possible generic property will be 'being active.' There are two ways in which this property can be manifested: In source 1: ('active', 'has an account'), and in source 2: ('active', 'made purchase in the past three months'). Thus, this representation enables querying the interoperated application on the property 'active'.

Assume there are three other generic properties. First, source 1 uses the property 'customer number' to identify some instances, while source 2 uses the property 'telephone number' to identify some instances. These two types of unique identifiers can be reconciled using a generic property 'id' with two manifestations – in source

1(id, 'customer number') and in source 2 (id, 'telephone number'). If one inquires about the identity of a purchaser, depending on the source, the answer can be: (id, ('customer number', '12345')) or (id, ('telephone number', '987-6543')).

The case of id is different than the previous one (of 'active' customer), as 'customer number' and 'telephone number' are not actual values, but a form of *intermediate* property (i.e., each has both preceding and preceded properties).

Second, assume source 1 uses the property 'age group' with values ('toddler', 'child', 'youth', 'young adult', 'adult', 'mature adult', 'senior'), while instances in source 2 have the property 'age' with a specific value. These properties can be reconciled by recognizing that a given value in 'age group' precedes (i.e., is a generic representation of) a group of values of 'age.' Thus, data in source 1 can be represented as ('age group', 'child'), ('age group', 'youth') and so on, while data in source 2 can be transformed and represented as (age group, ('toddler' (age, '1'))). The sources can then be reconciled through the generic representation in source 1 ('age group', 'xxx'), where 'xxx' denotes the age group and the evidence in terms of actual age in source 2. In this example, the sources contain different levels of granularity (age groups versus specific ages), but they are still able to interoperate. Furthermore, there is access to the more detailed data from source 2.

Third, assume source 1 includes 'sales person id' while source 2 includes 'sales team number'. These can be reconciled as manifestation of a preceding property: 'sales contact'.

Based on the properties, one can create a class:

**Customer:** (id, active, sales contact, age group)

Specific instances or records in source 1 can be represented as:

((id, (customer number, n)), (active, 'has customer id'), (sales contact, (team, t)), (age group, g)), where n, t, g, are specific values.

Specific instances or records in source 2 can be represented as:

((id, (telephone number, tn)), (active, (last purchase, 'less than 3 months')), (sales contact, (sales person id, p)), (age group, (g, (age, a)))), where tn, p, and a are specific values.

This approach achieves what can be termed *manifestation independence*. That is, it supports using information from multiple sources independent of how generic properties are manifested in the specific sources.

Finally, we note that the common semantics identified in the way described above can be used for instance reconciliation – where the data about one instance are assembled from more than one source. In some cases, the data from the two sources will be complementary. However, in other cases, the same generic data item might be manifested in more than one source. Two approaches can be then taken. First, one can list for each instance all manifestations of a certain generic property (for example, listing both customer id and an indicator of purchase in last three months as the multiple-value of 'active'. Second, some source priority can be established whereby a given generic attribute will be assigned a value from the preferred source. In cases where manifestations in one source are preceded by manifestations from another source (e.g. age is preceded by age group), the whole precedence sequence can be reported.

Finally, we note the possibility of conflicting values between two sources. For example, one database might indicate a person is a graduate student, while the other

database will show that the person is enrolled in a program only available to undergraduate students. We do not address possible value conflicts here, as our intention is only to show how common semantics can be established for attributes from different sources. Furthermore, this raises the issue of how to define and identify conflicts and how to determine which data value is the correct one. We take the view that one can list all manifestations and have the user or application program deal with conflicts. However, the methods suggested in this paper can be used to identify otherwise hidden data conflicts. As well, the listing of complete sequence of ‘evidence’ for each manifestation (based on precedences) can support value conflict resolution.

#### 6.4 Generalizing the Notion of Attribute Similarity

As discussed in our literature review, common attribute-based approaches seek similarities among attributes based on structural characteristics of attributes and on statistical analysis of the ranges of attribute values. In other words, these approaches assume that semantic similarity is manifested via non-semantic characteristics of attributes.

Our approach is different as it defines semantic similarity in terms of precedence semantics, namely, properties are viewed as being the “same” based on inferences that can be made about other properties possessed by an instance. It does not require that attributes have similar non-semantic characteristics. The examples above have demonstrated that attributes having different domains of values can still be viewed as similar. Moreover, structural and statistical information can only serve as indicators for possible similarities. In contrast, if it is known that two properties are preceded by the same generalized property, then this is a definite indication of higher-level similarity. In some cases, this similarity might not be important for a specific application (e.g., a query); nevertheless, it will still be discovered.

Of course, this benefit does not come without a cost. The cost associated with our generalization of similarity is the need to identify precedences and generalized properties that might not even be known within the individual sources. Obtaining this information might be difficult. We note, however, that it is not more difficult than constructing domain ontologies for ontology-based approaches. On the contrary, identifying precedences among properties can be viewed as identifying a limited ontology of the domain, rather than a more general one, since only links indicating precedences must be specified.

### 7 Practical Considerations

Clearly, our approach depends on the ability to identify precedences and generic properties. In our analysis above (Section 6) we assumed the precedences among properties within and among sources, and global generic properties, were known. In this section we address some issues related to the identification of precedences.

## 7.1 Identifying Precedences

In order for the notion of precedence to serve as a practical basis for reconciling sources to support interoperability, the sources will need to be examined to identify precedence relationships. The following steps constitute a general method for semantic reconciliation:

1. Identify all intra-source precedences for each available source.
2. Identify all inter-source precedences (i.e., where properties in one source precede or are preceded by properties in another).
3. In cases where inter-source precedences exist, use the two lemmas (section 5.2) to infer additional precedences or to seek precedences that might exist.
  - a. Lemma 1 can indicate higher-level properties that can be used for reconciliation. In cases where the conditions of Lemma 1 arise, identify  $G_1$  for reconciliation (it is the more generic).
  - b. Lemma 2 can be used to indicate that lower level properties in one source can be viewed as manifestations of lower level properties in the other.

To demonstrate the use of the Lemma 1 across sources, assume  $G_2$  is ‘having a loan’ and  $S_2$  specifies the types of loans possible. Each type of loan might imply a specific commitment to pay (value in  $S_1$ ). Within a specific institution, this is all that is needed. However, recognizing that the commitment to pay is a manifestation of liability, one can define a property  $G_1$  ‘liability’. This property can be generic over several financial institutions the person or firm does business with. It can be used to query for liabilities over several sources.

Conversely, Lemma 2 can be used to find out that specific values in one source may be linked (by precedences) to values in another source. For example, let  $G_2$  be ‘having loans’ manifested by type of loan, in one source. Let  $G_2$  be preceded by  $G_1$  which is ‘liabilities’. If, for each case where a person or a bank has liabilities, these liabilities are detailed in a second source, then we know that, in principle, each loan in the first source can imply a specific type of liability in the second source. Note, however, that the Lemma does not show how to find the specific relationships between values across sources.

Within a single data source, precedences can be identified in terms of subsumed scopes. The scopes can be computed by identifying, for each property, all instances possessing it (as suggested in the two-layer approach of Parsons and Wand [21]). We term the precedences within a source the *local precedence schema*. However, it might not be possible to infer all precedences from the data. The data may not be complete, leading to “spurious” precedences that will disappear when more instances are known. In addition, additional knowledge about property precedence that was not used originally to design the database might exist.

Another approach to identify both within-source and between-source precedences is to examine the domain of data attributes. If it is known a mapping exists from one domain to another, this points out that the conditions of Lemma 1 might hold. Such mapping might typically occur when attributes relate to quantitative characteristics (e.g. calculating weights based on physical dimensions, or costs based on producton data).

However, in general, the inter-source precedences might be hard to deduce by scanning multiple sources, as the data might need to be reconciled first, and this is the issue we are trying to solve.

Finally, we introduce a “principle of minimality” for reconciliation. When searching for more generic properties, one might identify a “hierarchy” of preceding properties for a given set of properties that need to be reconciled. An earlier example about movement identified “move on land” as a generic property. However, “be able to move” is a more generic property. In reconciling meaning, we will use the following principle:

**Principle of Minimality:** For any given set of properties for which common generic properties exist, in reconciling meaning, use as the generic property one that is lowest in the property hierarchy.

For example, if one considers sources that only refer to things that move on land, water, and air, then reconciliation might be done by the medium in which movement occurs. However, if one adds a source that includes things that are or are not self-propelling, but does not deal with the medium of movement, then it may be necessary to generalize to a higher-level generic property, such as “being able to move”.

To end this subsection, we note that most of our examples were simple, with only two “levels” of property: generic and specific. In general, properties might be arranged in deeper hierarchies.<sup>7</sup>

## 7.2 Towards a Precedence Ontology and Precedence Broker

The application of the property precedence approach to semantic reconciliation requires that the precedences among properties representing relevant data attributes in the various sources, and generic properties for these properties, will be known. For a given set of properties, we term the precedences that exist among them, and between them and any additional generic properties, a *precedence schema*. This schema can be considered a *property ontology* of the domain. We propose that such an ontology captures one aspect of the domain that is particularly important for semantic reconciliation of properties, although it may be difficult to construct.

An approach to overcome this potential problem is to define, for a given domain, a generic schema of accepted property precedences. We term such a schema a *global precedence schema*. A global precedence schema can be used to identify cases where a preceding property cannot be identified in the local precedence schema because it is not directly relevant in a specific database. Moreover, as more sources interoperate, such a schema can evolve to include more generic properties.

In a practical situation, such a global precedence schema might be operated by an independent party and act as a *precedence broker*. A multi-source application can identify possible generic properties by querying the precedence broker about precedences within and across sources.

---

<sup>7</sup> We use the word “hierarchy” as a simplification. In general, the precedence relation actually creates a directed acyclic graph over a given set of properties.

To demonstrate the use of a precedence broker, consider two sources about species of animals native to different countries. These sources keep information about physical characteristics such as limbs. In source A, each animal has a property indicating the number of limbs (e.g., ‘four limbs’), which is preceded by the property ‘has limbs.’ Source B includes the generic properties ‘has wings,’ ‘has arms,’ and ‘has legs’. These generic properties in turn precede manifestations such as ‘two arms’ or ‘four legs.’

Consider a query to both sources about all species having a specified number of limbs. Processing the query will require reference to the precedence relations explicitly defined or implicit in each source. In source A, the precedence relation ‘has four limbs’  $\rightarrow$  ‘has limbs’ (where  $\rightarrow$  means ‘is preceded by’) will be explicit, since both properties are defined. Source B, however does not contain a property meaning ‘has limbs.’ Instead, it only contains precedences such as ‘has two wings’  $\rightarrow$  ‘has wings,’ ‘has four legs’  $\rightarrow$  ‘has legs,’ ‘has two arms’  $\rightarrow$  ‘has arms.’ Thus, within the context of this source, a query specifying the number of limbs is meaningless.

To facilitate information sharing between these sources, it may be necessary to refer to a precedence broker containing generic precedence relations. For example, the broker may contain precedences such as ‘has legs’  $\rightarrow$  ‘has limbs,’ ‘has wings’  $\rightarrow$  ‘has limbs,’ and ‘has arms’  $\rightarrow$  ‘has limbs.’ These precedences will enable inferring that instances in source B that possess properties preceded by ‘has limbs’ should be retrieved in any query involving the property ‘has limbs,’ even though source B does not contain that property.

## 8 Conclusions and Further Research

We propose the concepts of property precedence and property manifestation as the basis for a novel approach to support attribute-based semantic reconciliation across independent data sources. The essence of the approach is that to reconcile data semantically, all one needs to know is the relationships among properties, not their actual meanings. The approach can be used in both traditional structured databases and in emerging semi-structured independent information sources on the Internet.

Our approach differs from extant attribute-based methods. First, depending on the level of abstraction, two properties can be *both* the same or different. In “traditional” semantic reconciliation methods, two properties are *either* the same (or similar) or they are different. Precedence semantics recognizes that, on some higher level, different attributes might be similar. Second, it supports the reconciliation of attributes that have different names and/or their value domains. This is in contrast to statistically-based methods. Third, it can provide with the result of a query the “evidence” why certain answers were obtained. Fourth, it can support the creation of common views (in terms of classes) based on generic (rather than same) properties.

All this has a “cost” attached to it – the method requires identification of precedences and of generic properties that might not be included in any of the sources. On the other hand, by specifying these requirements, the method provides a clear definition of what is the nature of the semantics needed to support attribute-based reconciliation.

This approach presents numerous opportunities for further research. Work is needed to explore methods to develop domain specific property precedence schemas, and to evaluate their usefulness. There are various additional kinds of precedence to be analyzed, in particular those related to different manifestations of the same property. Also, analysis here is restricted to what Bunge [2] terms *intrinsic properties*. The meaning and implications of precedence in the context of *mutual properties* (relating two or more instances, such as “product price for a given customer” or “source of specific components for a specific product”) need to be explored. From an implementation point of view, the use of XML-style tagging to represent precedence orders within information sources can be explored. Finally, the precedence broker architecture needs to be designed, implemented, and evaluated on a large practical example to concretely demonstrate feasibility and benefits of the approach.

## References

1. Batini, C., M. Lenzerini, and S.B. Navathe, “A Comparative Analysis of Methodologies for Database Schema Integration,” *ACM Computing Surveys*, 18(4), 1986, 323–364.
2. Bunge, M., *Treatise on Basic Philosophy (Volume 3), Ontology I: The Furniture of the World*, Boston: Reidel, 1977.
3. Bunge, M., *Treatise on Basic Philosophy (Volume 4), Ontology II: A World of Systems*, Boston: Reidel, 1979.
4. Castano, S., V. De Antonellis, M. Fugini, and B. Pernici (1999), “Conceptual Schema Analysis: Techniques and Applications,” *ACM Transactions on Database Systems*, 23(3), 286–333.
5. Castano, S., V. De Antonellis, and S. De Capitani di Vimercati (2001), “Global Viewing of Heterogeneous Data Sources,” *IEEE Transactions on Knowledge and Data Engineering*, 13(2), 277–297.
6. Clarke, C., G. Cormack, and F. Burkowski (1995), “Schema-Independent Retrieval from Heterogeneous Structured Text,” in *Fourth Annual Symposium on Document Analysis and Information Retrieval*, 279–289.
7. Clifton, C., E. Housman, and A. Rosenthal (1997), “Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases,” *Data Mining and Reverse Engineering. Proceedings of DS-7, IFIP 1997*.
8. Cohen, W. (1998), “Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity,” *Proceedings of the Conference on Information and Knowledge Management (CIKM'98)*, Seattle, WA, 201–212.
9. Doan, A., P. Domingos, and A. Halevy (2001), “Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach,” *Proceedings of ACM SIGMOD*, Santa Barbara, CA, 509–520.
10. Goble, C., S. Bechhofer, L. Carr, D. De Roure, and W. Hall (2001), “Conceptual Open Hypermedia = The Semantic Web?” *Proceedings of the 2001 Semantic Web Workshop*, Hong Kong, China, 7 pages.
11. Hulgeri, A., G. Bhalotia, C. Nakhe, and S. Chakrabarti (2001), “Keyword Search in Databases,” *IEEE Data Engineering Bulletin*, 24(3), 22–32.
12. Lee, J.-O. and D.-K. Baik (1999), “SemQL: A Semantic Query Language for Multidatabase Systems,” *Proceedings of the Conference on Information and Knowledge Management (CIKM'99)*, Kansas City, MO, 259–266.

13. Li, W.-S. and C. Clifton (1994), "Semantic Integration in Heterogeneous Databases Using Neural Networks," *Proceedings of the 20<sup>th</sup> VLDB Conference*, Santiago, Chile, 1–12.
14. Litwin, W., L. Mark, and N. Roussopoulos (1990), "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys*, 22(3), 267–293.
15. Maedche, A. and S. Staab (2001), "Learning Ontologies for the Semantic Web," *Proceedings of the 2001 Semantic Web Workshop*, Hong Kong, China, 10 pages.
16. Miller, R., "Using Schematically Heterogeneous Structures," *Proceedings SIGMOD '98*, Seattle, WA, 189–200.
17. Miller, R., M. Hernandez, L. Haas, L. Yan, H. Ho, R. Fagin, L. Popa (2001), "The Clío Project: Managing Heterogeneity," *SIGMOD Record*, 30(1), 78–83.
18. Naiman, C. and A. Ouksel, "A Classification of Semantic Conflicts in Heterogeneous Information Systems," *Journal of Organizational Computing*, 5(2), 1995, 167–193.
19. Paepcke, A., C.-C. Chang, H. Garcia-Molina, and T. Winograd (1998), "Interoperability for Digital Libraries Worldwide," *Communications of the ACM*, 41(4), 33–43.
20. Parent, P., S. Spaccapietra (1998), "Database Integration: an Overview of Issues and Approaches," *Communications of the ACM*, vol. 41, no 5, pp. 166–178, May 1998.
21. Parsons, J. and Y. Wand, (2000), "Emancipating Instances from the Tyranny of Classes in Information Modeling," *ACM Transactions on Database Systems*, 25(2), 228–268.
22. Qian, X. (1993), "Semantic Interoperation Via Intelligent Mediation," *Proceedings of the 1993 International Workshop on Research Issues in Data Engineering*, 228–231.
23. Rahm, E. and P. Bernstein (2001), A Survey of Approaches to Automatic Schema Matching," *VLDB Journal*, 10, 334–350.
24. Reddy, M.P., B.E. Prasad, P.G. Reddy, and A. Gupta (1994), "A Methodology for Integration of Heterogeneous Databases," *IEEE Transactions on Knowledge and Data Engineering*, 6(6), 920–933.
25. Sciore, E., M. Siegel, and A. Rosenthal (1994), "Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems," *ACM Transactions on Database Systems*, 19(2), 254–290.
26. Smith, K. and L. Obrst, "Unpacking the Semantics of Source and Usage to Perform Semantic Reconciliation in Large-Scale Information Systems," *ACM SIGMOD Record*, 28(1), 1999, 26–31.
27. Toivonen, S. (2001), "Using RDF(S) to Provide Multiple Views into a Single Ontology," *Proceedings of the 2001 Semantic Web Workshop*, Hongkong, China, 6 pages.
28. Wand, Y., "A Proposal for a Formal Model of Objects," in W. Kim (ed.) and F. Lochovsky (eds.), *Object-oriented Concepts, Databases and Applications*, Reading, MA: Addison-Wesley, 1989, 537–559.
29. Wand, Y., V. Storey, and R. Weber, "An Ontological Analysis of the relationship Construct in Conceptual Modelling", *ACM Transactions on Database Systems*, Vol. 24, No. 4, December 1999, pp. 494–528
30. Wand, Y. and Weber, R., "An Ontological Model of an Information System," *IEEE Transactions on Software Engineering*, 16(11), November 1990, 1282–1292.
31. Wand, Y. and Weber, R., "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars," *Journal of Information Systems*, 1993, 217–237.
32. Wand, Y. and Weber, R., "Towards a Theory of Deep Structure of Information Systems," *Journal of Information Systems*, 1995, 203–223.



# Data Quality in Web Information Systems

Barbara Pernici<sup>1</sup> and Monica Scannapieco<sup>2\*</sup>

<sup>1</sup> Dipartimento di Elettronica e Informazione,  
Politecnico di Milano  
Piazza Leonardo da Vinci 32, 20133 Milano, Italy  
`barbara.pernici@polimi.it`

<sup>2</sup> Dipartimento di Informatica e Sistemistica,  
Università di Roma “La Sapienza”  
Via Salaria 113, 00198 Roma, Italy  
`monscan@dis.uniroma1.it`

**Abstract.** Evaluation of data quality in web information systems provides support for a correct interpretation of the contents of web pages. Data quality dimensions proposed in the literature need to be adapted and extended to represent the characteristics of data in web pages, and in particular their dynamic aspects. The present paper proposes and discusses a model and a methodological framework to support data quality in web information systems. The design and a prototype implementation of a software module to publish quality information are also described.

## 1 Introduction

Web Information Systems (WIS's) [12] are characterized by the presentation to a wide audience of a large amount of data, the quality of which can be very heterogeneous. There are several reasons for this variety, but a significant reason is the conflict between the need of publishing high quality information and publishing information as soon as it becomes available. Providing information with a good quality level is a traditional goal of information systems: theoretical work in database design has focused on proposing appropriate data structures and design methods for guaranteeing the quality of data (e.g., [4]); in addition, more recent research work on data quality focuses on the quality of instances of data, by defining a number of data quality dimensions (e.g., [26]) and tools to support data quality improvement. On the other hand, information systems on the web need to publish information in the shortest possible time after it is available from information sources. These two requirements are in many aspects contradictory: accurate design of data structures, and in the case of web sites, of good navigational paths between pages, and certification of data to verify its correctness are costly and lengthy activities, while publication of data on web sites requires stringent times.

---

\* The work of Monica Scannapieco is supported by *Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche (IASI-CNR)*, Viale Manzoni 30, 00185 Roma, Italy.

Such a tradeoff between data quality and timeliness of information is a problem which has to be taken into account also in traditional publishing areas, such as newspapers and journals. However, WIS's present two peculiar aspects with respect to traditional paper-based information sources: first, a web site is a continuously evolving source of information, and it is not linked to a fixed release time of information; second, the process of producing information changes, additional information can be produced in different phases, and corrections to previously published information are possible. Such features lead to a different type of information with respect to traditional media.

Therefore, the persons in charge for information release on web sites are often faced with the need to decide, for single items of information, or for groups of data with similar characteristics, whether or not a given piece of information should be published at a given time.

The goal of this paper is to propose a data model and a methodological framework that support the task of data publication on the web, providing tools to associate quality information to data in web pages, both as static values, and as expected evolution of quality in time due to subsequent updates. The original contribution of the paper focuses mainly on the dynamic nature of quality information and on the need to define a methodological framework and tools to manage data quality information.

In the following of the paper, we discuss data quality in WIS's presenting examples taken from a university web site. Let us consider, for instance, the case in which a university has to present to perspective students new offerings in terms of curricula for the following academic year: the choice in terms of time of publication is between waiting that all curricula are defined in order to present the complete view on available degrees, or to present, maybe with different emphasis and details, new offerings and previously available degrees at different times. The second choice is more flexible, but it presents the disadvantage that perspective students could miss important information if they visit the university site before information they are looking for is available. On the other hand, the first option requires that all decisions are taken, and it will probably postpone the date of publication of information with respect to the second alternative; in some cases the final decisions could be taken in a time frame which is not compatible with the communication needed to publicize degree offerings for the following year.

In the previous example, a possible solution to miscommunication problems is to provide a meta-level information about the information being published. Quality information can be associated to each information item in a page (or groups of items, or pages), expressing properties related to the expected validity of each information, its completeness, and the quality of the information source. The evolution in time of quality information in WIS's is particularly important, since it provides the user of such information with a basis for deciding whether to use and how its contents. Therefore, the dynamic evolution of quality properties is also taken into account.

The structure of the paper is as follows. In Section 2, the data quality dimensions for WIS's are presented and discussed. In Section 3, a model for data

quality is proposed and in Section 4, a methodological framework for data quality is described. In Section 5, an architecture realizing the proposed framework is described. In Section 6, the design and implementation of a software module to publish quality information is presented. Finally, in Section 7, related research work is discussed and Section 8 concludes the paper by drawing future work and the applicability of the approach.

## 2 Definition of Data Quality Dimensions

In the literature, data quality is considered as a multidimensional concept [21], i.e., it is defined on the basis of a set of “dimensions”. Many proposals concerning the set of dimensions characterizing data quality have been made, a survey of which is given in [25]. One of the reasons for such a wide variety is that it is very difficult to define a set of data quality dimensions suitable for every kind of context. In the present paper, we focus only on a “core” set of dimensions, aiming at capturing the most important aspects of data quality in WIS’s, leaving to future work possible extensions of this set. The choice of this set has been guided by those that are used most frequently in the literature [23]. In the following, we give a definition for these core dimensions, and, in order to provide support for the evolving nature of information published in WIS’s, we also associate metrics to indicate the expected evolution of quality in time. We consider the following four data quality dimensions:

- **Expiration.** It is defined as the time until which data remain current. As an example, let us consider the case of a professor of a department of a university that is going to move to another university. If a list of the members of the department is published, this situation can be represented by associating to the professor as a member of the department its expiration date, corresponding to the date when the professor leaves the university. Expiration is a time-related dimension; dimensions of this type are differently defined in the literature, and there is no agreement on them. As an example, in [2] timeliness is defined in terms of currency (how recent are data) and volatility (how long data remains valid). This definition of volatility recalls our definition of expiration: we have instead chosen to give a dynamic semantics to volatility, as detailed in section 2.1. In [21] only currency is defined, while in [26] and [18] only timeliness is considered as a time-related dimension.
- **Completeness.** It is defined as the degree to which the elements of an aggregated element are present in the aggregated element instance. Therefore, completeness is defined only for elements that are an aggregation of other elements, such as lists, tables, and so on. Let us consider a list of courses published on the university site, and the situation in which it is known that new courses will be added to the list. A completeness measure can be associated to the list to make it explicit that the list is still partial.

A recent work [20] distinguishes three kinds of completeness, namely: schema completeness, column completeness and population completeness. The measures of such completeness types can give very useful information for a “general” assessment of the data completeness of an organization, especially if data are relational. Instead, we propose a different semantics for completeness, taking into account the homogeneous nesting structure of some information available on the Web.

- **Source reliability.** It is defined as the credibility of the source that provides data.

Let us suppose that the starting date for a semester is available on the university web site. Before having official information (e.g., a rector’s decree), it typically happens that such information is already circulating, e.g., in administrative offices. If there is a need to make this information visible before it is finally approved by decree, we can associate a **LOW** value of source reliability to the date. Conversely, if the date is provided by the official source for such information in the university, i.e., it has been validated and officially approved, it will have a **HIGH** source reliability value. The same data source may have different source reliability values for different data it provides. As an example, the school administrative office may have a **LOW** source reliability value for starting dates of semesters; instead it may have a **HIGH** source reliability value for course programs.

The proposed definition is the same as provided in [26] for source reputation.

- **Correctness.** It is the distance between the data value  $v$  and  $v'$ , being  $v'$  the value considered as correct.

Let us consider the following example. A professor published on the university web site has an attribute name, the value of which is **JHN**, while the correct value is **JOHN**: this is a case of low correctness, as **JHN** is not an admissible value according to a dictionary of English names.

Notice that, we have adopted for correctness the definition proposed in [21].

## 2.1 Dynamics of Data Quality Dimensions

One of the aspects of data quality in WIS’s is the possible evolution in time of some data. For instance, if an incomplete course list is published, the completeness dimension presented above provides only a static measure of the completeness of such a list. An important information is also the expected evolution of the list (“how and when it will be completed?”). In this section we show how to associate temporal dynamics to the expiration and completeness dimensions.

We assume instead that source reliability and correctness are constant in time for a certain instance of data, i.e., if their values vary, it is not anymore the same data. Considering the previous example of the starting date for a semester provided by an administrative office and thus having a **LOW** source reliability value, this date may be confirmed or not by the official source; in both cases, the date communicated by the official source is a new data item with a **HIGH** source reliability value; similar considerations can be also made for correctness.

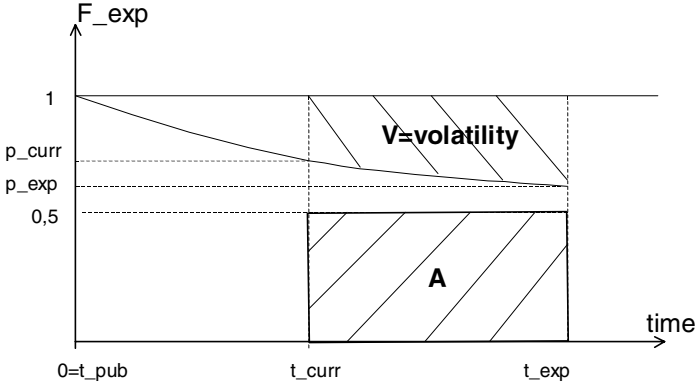
**Temporal Dynamics of Expiration: Volatility.** With reference to the temporal dynamics of expiration, we consider a function corresponding to the probability that the expiration time will change in the time interval starting from the instant of publication and ending with the expiration time. More formally, let us consider the following values:

- $t_{pub}$ : instant of publication of data,
- $t_{exp}$ : expiration time.

We consider the event  $E_t$  defined as “ $t_{exp}$  is the actual expiration time at the instant  $t$ ”. Then we introduce a function  $F_{exp}(t)$  = probability distribution of  $E_t$ . Starting from the function  $F_{exp}(t)$ , we can define a **volatility** of the published data as the probability at time  $t$  that  $t_{exp}$  will be the real expiration time, with  $t \in [t_{pub}, t_{exp}]$ .

**Definition:** Volatility is defined as  $V = (t_{exp} - t_{curr}) - \int_{t_{curr}}^{t_{exp}} F_{exp}(t) dt$ , where  $t_{curr}$  is the time at which volatility is evaluated and  $t_{curr} < t_{exp}$ .

For data that are stable in time, we assume that  $F_{exp}(t)$  is not definable, therefore no volatility is associated to them. In Figure 1, we show an example where  $p_{curr}$  and  $p_{exp}$  are the values of  $F_{exp}(t)$  in the instants  $t_{curr}$  and  $t_{exp}$  respectively.



**Fig. 1.** A graphical representation of volatility.

The volatility of the data published in the instant  $t_{curr}$  can be graphically depicted as an area; it can range between:

- a minimum value equal to 0, when  $F_{exp}(t) = 1$  at all times after  $t_{curr}$ ;
- a maximum value equal to  $(t_{exp} - t_{curr})$ , when  $F_{exp}(t) = 0$  at all times after  $t_{curr}$ . Notice that this value should be considered as a theoretical limit, as it indicates that the probability that the expiration time  $t_{exp}$  will be the actual expiration time is 0.

We define a range **LOW**, **MEDIUM**, **HIGH** for volatility; if we consider as a reference the area  $A = \frac{(t_{exp} - t_{curr})}{2}$  (see Figure 1), volatility is:

- **HIGH**, if  $V > A$ ;
- **MEDIUM**, if  $V = A$ ;
- **LOW**, if  $V < A$ .

As an example of the usefulness of such kind of information, let us consider again the case of a professor moving to another university. It is possible to know that the date when the professor will leave is fixed or that it “may” change or that it is “very probable” that it will change. A **LOW** value of volatility corresponds to the fact that the date is definitive, a **MEDIUM** value to the fact that the date may change and a **HIGH** value to a very probable change of the date.

**Temporal Dynamics of Completeness: Completability.** In a similar way, we consider the temporal dynamics of completeness. We consider how completeness evolves in time, that is how a given aggregation of elements evolves towards completion. Let us consider the following given values:

- **t\_max**: the maximum time within which the aggregation of elements will be completed;
- **t\_pub**: the instant of publication of data,

We consider a function  $C(t)$  = estimate of the value of completeness in the instant  $t$ , where  $t \in [t_{pub}, t_{max}]$ . Starting from the function  $C(t)$ , we can define the **completability** of the published data.

**Definition:** Completability is  $C = \int_{t_{curr}}^{t_{max}} C(t) dt$ , where  $t_{curr}$  is the time at which completability is evaluated and  $t_{curr} < t_{max}$ .

In Figure 2, an example of completability is shown. Notice that the value corresponding to **t\_curr** is indicated as **c\_curr**; **c\_max** is the value for completeness estimated for **t\_max**. The value **c\_max** is a real reachable limit that can be specified for the completeness of an aggregation of elements; if this real limit does not exist, **c\_max** is equal to 1.

As in the case of volatility, we can define ranges for completability (**HIGH**, **MEDIUM**, **LOW**). We consider as a reference the area  $A = \frac{(t_{max} - t_{curr}) * (c_{max} - c_{pub})}{2}$  (see Figure 2). Then completability is:

- **HIGH**, if  $C > A$ ;
- **MEDIUM**, if  $C = A$ ;
- **LOW**, if  $C < A$ .

As an example, we can consider the list of courses published on the university web site; the completeness dimension gives the information about the current degree of completeness; the completability information gives the information about how fast such degree will grow in time, i.e., how fast the list of courses will be completed.

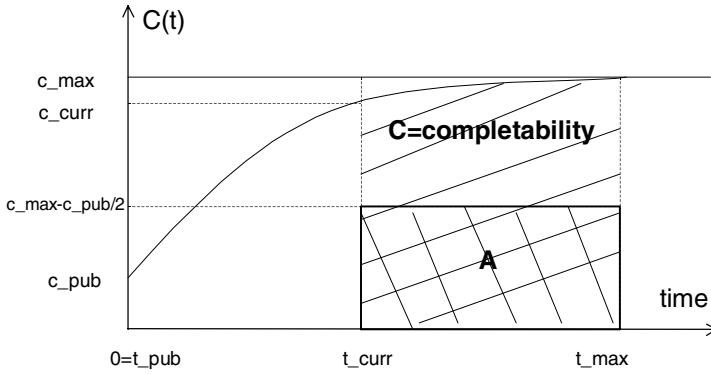


Fig. 2. A graphical representation of completability.

### 3 Modeling Data Quality

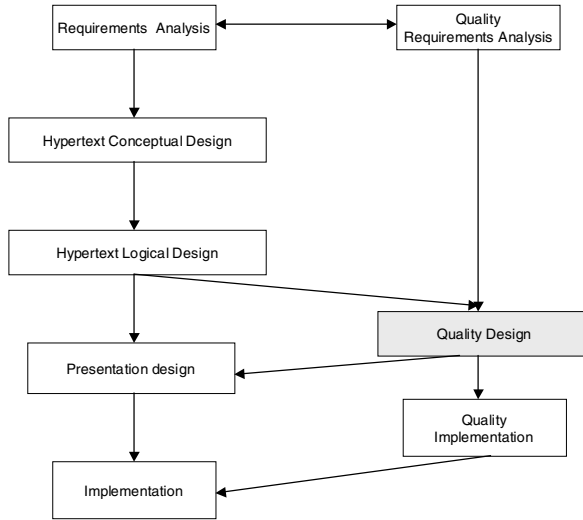
#### 3.1 WIS Design Methodology

Several methodologies for WIS design (e.g., ARANEUS [16] and RMM [14,13]) are based on the following sequence of steps (see Figure 3, left side):

- *Requirements Analysis*: it aims at understanding the information domain, what the WIS has to provide and who will be the users.
- *Hypertext Conceptual and Logical Design*: it implies the conceptual modeling of two aspects: (i) data that will be part of the hypertext, and (ii) paths to navigate among published data. This step also includes the design of some “logical” constructs, related to the ways to access information (i.e. indexes, tours, etc.) and to the structure of pages.
- *Presentation design*: it is focused on the interface design of the WIS.
- *Implementation*: it implies the generation of web pages.

To support the association of data quality information to web pages, we propose to introduce additional steps, related to the previously described ones, specifically devoted to data quality design (see Figure 3, right side):

- *Quality Requirements Analysis*: it is focused on the definition of the required quality dimensions to be associated to publishing data and of criteria and metrics associated to each dimension. The data quality dimensions in our approach are based on the definitions provided in the previous section.
- *Quality design*: the goal of this phase is to associate data quality information to pages and their contents.
- *Quality Implementation*: associating quality values to the previously defined dimensions.



**Fig. 3.** The process of realization of a WIS (left side) and activities for data quality design (right side). The grey task is further considered in Section 3.2.

### 3.2 A General Hyperbase Data Model for Data Quality

In this section, we show how to realize the step of quality design, shown in Figure 3, by proposing a model to associate quality information to web-publishing data.

Specializing a terminology introduced in [10], in the following we call as **hyperbase data model** a specific data model for hypertext representation that includes data structures, while being “purged” from the strictly navigational elements. An example of a hyperbase data model is the one of RMM [14,13], which includes constructs such as attributes, entities, slices (i.e., groups of attributes of an entity), etc.; other constructs of RMM, such as guided tours, are not included as they are navigational elements.

We propose a **General Hyperbase Data Model** (GHDM) in order to abstract the features of hyperbase data models which are more relevant from a data quality perspective.

With respect to the possibility of associating data quality information to data elements, an important feature is the “granularity” of a data element. To clarify this concept, consider the source reliability dimension: it can be associated to a list of elements, if each element of the list derives from the same source; conversely, it can be associated to each element of the list, when the sources for them are different, and possibly with different source reliability values.

Therefore, in our general hyperbase data model we distinguish between atomic and aggregated elements and we associate data quality information to both of them by defining a *quality atomic element* and a *quality aggregated element*.



**Definition:** a **quality atomic element** is defined as a tuple  $\langle \text{atomic element, Set of Quality Dimensions (SQD)} \rangle$ , in which:

- an atomic element is the schema of the smallest piece of information to be published on the Web (e.g., an attribute);
- a possible SQD is the one proposed in Section 2, and includes: (i) expiration, (ii) volatility, (iii) source reliability, (iv) correctness.

**Definition:** a **quality aggregated element** is a tuple  $\langle \text{aggregated element (of level i), Set of Quality Dimensions (SQD)} \rangle$ , in which:

- aggregated elements can have different levels of aggregation, i.e., they can be the aggregation of atomic elements or the aggregation of aggregated elements; examples are entities in the ER-model, that are an aggregation of attributes;
- a possible SQD includes: (i) expiration, (ii) volatility, (iii) source reliability, (iv) correctness, (v) completeness and (vi) completability.

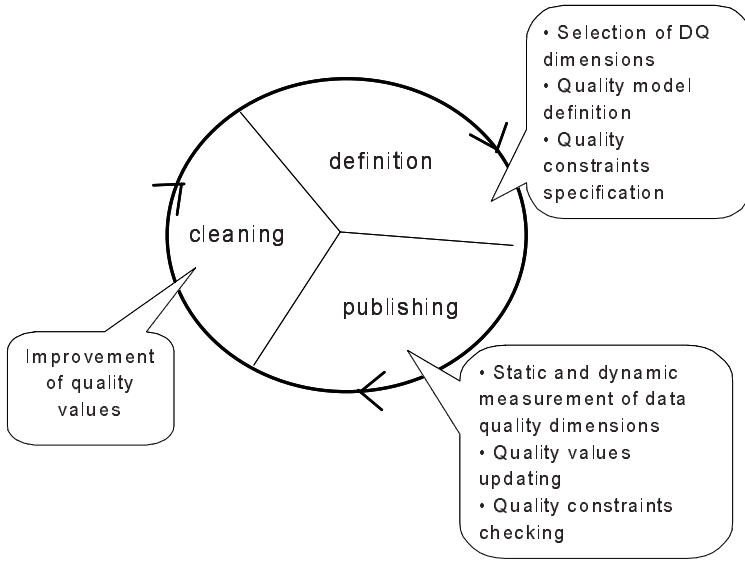
## 4 A Methodological Framework for Data Quality Management in WIS's

When introducing data quality in a WIS, in addition to data quality design, further issues must be considered, concerning data quality management: as an example, problems related to the updating of dynamically changing quality values and to quality improvement should be taken into account. In the present section, we describe a methodological framework for managing data quality in WIS's, discussing the principal activities that must be supported to manage data quality.

The main activities that should be made in order to cope with the requirements of an effective data quality management include:

- the definition of a set of data quality dimensions to associate to information to make available on the web;
- the definition of a conceptual model that defines quality elements according to which modeling data quality dimensions and associating quality information to data elements;
- the specification of a set of quality requirements to be satisfied by published data;
- the measurement of quality dimensions values for information to be published;
- the updating of quality dimension values that dynamically change;
- the improvement of the quality of published data.

We propose a methodological cycle to manage data quality in a WIS; the cycle consists of three iterative phases, namely: *Definition*, *Publishing* and *Cleaning* (see Figure 4). In the following subsections, each of the phases will be described.



**Fig. 4.** A methodological cycle for data quality management in WIS's.

#### 4.1 Definition Phase

In this phase, the three following elements are to be considered:

- Data quality dimensions. The data quality dimensions that are interesting for the WIS should be chosen. A comprehensive choice should include both static and dynamic aspects of quality dimensions. The set of dimensions proposed in Section 2, and including expiration, completeness, source reliability, correctness to catch static features and volatility and completability to catch the dynamic ones, is a potential set.
- Quality Model. A quality model has the role of modeling the chosen data quality dimensions as well as their associations to data elements in the data model. In Section 3.2, the GHDM has been proposed with the aim of addressing these features.
- Quality Constraints. It could be defined a set of quality constraints that data published in a WIS need to satisfy. A quality constraint can be seen as an expression  $\langle \text{dimension} \rangle \langle \text{operator} \rangle \langle \text{value} \rangle$ , where  $\langle \text{dimension} \rangle$  is any data quality dimension,  $\langle \text{operator} \rangle$  can be  $=, <, >, \leq$  or  $\geq$ , and  $\langle \text{value} \rangle$  is defined in the domain of the data quality dimension.

As an example, a data quality constraint that it is possible to state is that Course Programs are published on the university web site only if **correctness**  $\geq 6$ , if measuring correctness over a domain  $\{1..10\}$ .

A set of quality constraints can be specified: (i) for data which are going to be published, i.e., they will be published only if they satisfy such quality

constraints; *(ii)* for published data in order to enact data cleaning actions when a fixed percentage of published data does not satisfy the specified constraints.

## 4.2 Publishing Phase

This phase generally includes the measurement of data quality values to publish as well as their updating. Moreover it is also possible to fix some “thresholds” for data quality dimensions values according to which to decide whether or not to publish some data. As an example, in the university web site one can choose that the information about the dates of examinations must be published only if their correctness is **HIGH**.

Such thresholds are managed in the monitoring activity and can also trigger cleaning actions when the quality of published data goes down acceptable values.

The publishing phase is structured according to the following steps:

- static measurement;
- dynamic measurement;
- monitoring.

**Static Measurement.** This step implies the evaluation of the static values of data quality dimensions for data values which are going to be published. This step has to be made when data to be published have been decided; it is made once for a given set of data. Methods that can be used for measuring static data quality dimensions depend on each specific dimension chosen in the definition phase.

In the following, we describe possible methods to measure the set of quality dimensions proposed in Section 2. The expiration dimension is specified by the expiration date. For all data that expire, the expiration date should be directly associated to them. Moreover, when the expiration date for a data item is reached, two situations can occur: *(i)* expired data remain valid or *(ii)* expired data become not valid. To model this situation, expiring data have not only an expiration date associated to them, but also a **validity interval**, specifying the instant from which and the instant until which data remain valid [22].

As an example of such a situation, consider the case of a timetable of a course for a semester; supposing that the course is taught during the first semester of the year, the timetable expires at the end of the first semester but remain valid until the end of the year. Therefore, the timetable of the course has a validity interval [**Start\_1st\_Sem**, **End\_Year**].

Note that managers of the WIS can choose to maintain on-line not yet valid data or not, depending on the peculiarity of data themselves.

Completeness values can be calculated by considering the percentage of non-existent data values in aggregated elements. As an example, when publishing the list of the home pages of the professors of a university, all the URL’s may not be available; by publishing only the available ones, the completeness of the list is a percentage of all the professors’ home pages.

Source reliability values should be evaluated for each data type managed by the WIS. Data types, the values of which can be published only when an official source provide them (i.e., calls for tenders), are associated to HIGH source reliability values.

correctness can be checked by comparing data values with reference dictionaries (e.g., name dictionaries, address lists, domain related dictionaries such as product or commercial categories lists).

**Dynamic Measurement.** This step has to evaluate the dynamic values of data quality dimensions for publishing data. Differently from the static measurement, the dynamic measurement step is typically made many times, in specific instants that may be of two types: (i) pre-fixed, such as the publication instant (i.e.,  $t_{pub}$ ), and (ii) random, i.e., depending on unpredictable changes of parameters on which dynamic dimensions depend. In the data quality literature, as of our knowledge, there are no examples of dynamic data quality dimensions, therefore there are no proposals about measurement methods.

With reference to the dynamic dimensions proposed in Section 2.1, they can be measured as follows:

- volatility can be evaluated as HIGH, MEDIUM or LOW, according to the formulas described in Section 2.1. The pre-fixed instants of evaluation can be, beside the publication instant (i.e.,  $t_{pub}$ ), all other instants in the interval  $[t_{pub}, t_{exp}]$ . The random evaluation instants depend on anticipations or postponements of  $t_{exp}$ ;
- completability can also be evaluated similarly to volatility. The random instants may depend on eventually changes of the maximum time within which the aggregation of elements will be completed (i.e.,  $t_{max}$ ) and/or the corresponding maximum value for completeness (i.e.,  $c_{max}$ ).

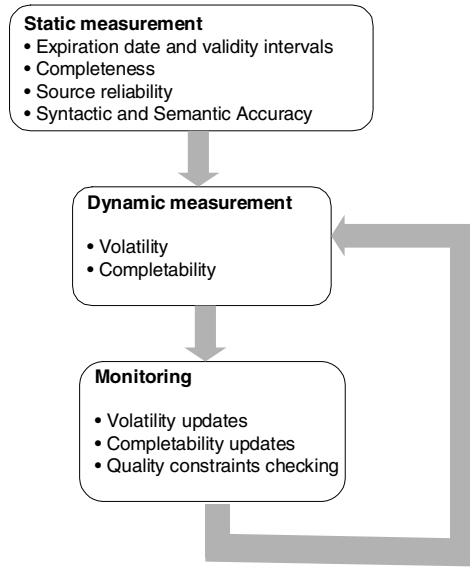
**Monitoring.** This step consists of controlling a set of parameters, on which data quality values associated to published web pages depend, and of enacting updates of such values. As an example, at the pre-fixed instants the evaluation (or re-evaluation) of dynamic values for data quality dimensions are enacted by monitoring conditions.

A further activity included in this step consists of the checking of the set of quality dimensions specified in the definition phase. The checking should regard both data which are going to be published and already published data, in order to enact cleaning actions.

The sequence of the publishing steps is shown in Figure 5.

### 4.3 Cleaning Phase

The cleaning phase concerns the improvement of the quality of source databases for published data. We propose to engage cleaning actions when the monitoring activity alerts that a specified set of quality constraints is not satisfied by



**Fig. 5.** Different steps in the publishing phase.

published data. There are many research results concerning the cleaning of data (e.g., [9,8]); but they only address correctness and completeness, among the dimensions proposed in the present paper, and consistency, among the dimensions that have not been considered. No attention is paid to the improvement of source reliability and dynamic dimensions that we plan to consider in future work. Note that the cleaning activity may not succeed in improving data as much as it is necessary to satisfy the defined quality constraints. In such a case, the definition phase should be newly engaged.

## 5 An Architecture

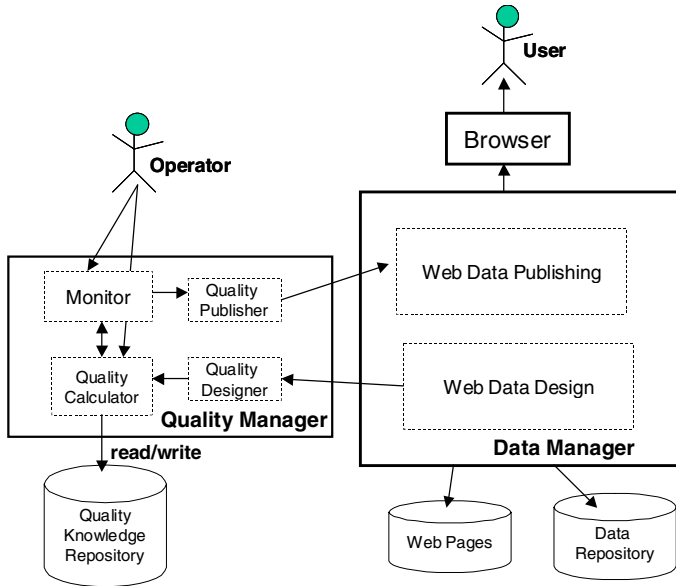
The methodological framework described in Section 4 enables to manage data quality in a WIS. Such a framework can be realized by the software architecture shown in Figure 6.

The elements specifically introduced for data quality management are:

- a *quality knowledge repository* and
- a *quality manager*.

Data values to publish are stored in the *data repository*; the *quality knowledge repository* contains, for each data value both of atomic and aggregated type, the static values of quality dimensions. Moreover, it also stores historical values for the estimation of dynamic values of quality dimensions.

The *quality manager* consists of four modules, namely *monitor*, *quality calculator*, *quality designer* and *quality publisher*. In order to describe how these



**Fig. 6.** An architecture supporting the quality management framework.

modules interact, we distinguish: (i) a publication mode and (ii) an updating mode.

In the publication mode, the *quality designer* receives the conceptual elements representing data to be published together with references to data values from the module *web data design*. It also receives the specification of the quality dimensions to associate to such elements. On the basis of a quality model, such as GHDM, it associates quality elements to data elements, and produces a quality schema.

At this point, the *quality calculator* receives the produced quality schema and the set of data values references according to which: (i) it reads static data quality values from the *quality knowledge repository*; (ii) it calculates dynamic values. The calculation of dynamic dimensions should be made by interacting with an operator who provides the module with the real-time information it needs for.

In the publication mode, the *monitor* has only the task of checking the set of quality constraints that have been previously provided by the operator, against the quality values resulting from the activity of the *quality calculator*. If the checking is successful, data and the associated quality values are passed to the *quality publisher*, which acts as a bridge towards the external *web data publishing* module (that has the task of presenting both data and quality data).

If the checking is not successful, an alert is sent to the operator who has to take the opportune actions.

In the updating mode, the *monitor* plays the most important role. It has some triggers coded, such as the ones related to pre-fixed instants when dynamic dimensions should be re-calculated; in this case, it requests to the *quality calculator* to compute again dynamic dimensions, and after the quality constraint check, it gives new quality values to the *quality publisher*.

Note that when dynamic dimensions need to be re-calculated at random instant, the *monitor* is alerted by the operator. Moreover, the *monitor* has also some rules to be checked in order to eventually enact cleaning actions; such rules are derived from quality constraints on the overall quality of published information. The quality controls aiming at cleaning should be periodically realized by the *monitor*.

## 6 Implementing the Quality Publisher

In order to publish quality information associated to web-available data, a major choice is related to which language can easily represent quality data. From a conceptual point of view, quality data represent a specific kind of *metadata*, thus we have chosen Resource Description Framework (RDF)[15,7] to represent quality data. RDF is currently recommended as a language for the Semantic Web [5]. The usage of RDF allows for quality data to be queryable not only by humans, such as web site users, but also by machines; in this way, smart services for searching or even integrating data could be based on quality data.

This section is structured as follows. We first describe how representing quality information through RDF in Section 6.1. Then, in Section 6.2 we show some implementation details of the *quality publisher* module, introduced in the previous section .

### 6.1 Designing the Data Quality File (DQF)

We suppose to publish data as HTML files. In a separate RDF file, called Data Quality File (DQF), we specify the quality data. An example of a Data Quality File is shown in the following.

```
(1) <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
(2)   xmlns:dq="http://DataQualitySchema">
(3)   <rdf:Description about = "Scannapieco Monica HomePage#page1">
(4)     <dq:DQproperties>
(5)       <dq:property>
(6)         <dq:Name>Author</dq:Name>
(7)         <dq:Info>Monica Scannapieco</dq:Info>
(8)         <dq:Comment>Student in Computer Engineering</dq:Comment>
(9)         <dq:Image>c:\demo\homepage_file\foto.gif</dq:Image>
(10)      </dq:property>
(11)      <dq:property>
(12)        <dq:Name>Last Update</dq:Name>
(13)        <dq:Info>29/07/2002</dq:Info>
(14)        <dq:Comment>Publications updated</dq:Comment>
(15)      </dq:property>
(16)    </dq:DQproperties>
(17)   <rdf:Description about="Curriculum Vitae#_page2">
```

```

(18)         <dq:DQproperties>
(19)             <dq:property>
(20)                 <dq:Name>Completeness</dq:Name>
(21)                 <dq:Info>70%</dq:Info>
(22)                 <dq:Comment>Completed within 12/31/2002</dq:Comment>
(23)                 <dq:Image>C:\Demo\HomePage_file\Completeness.jpg</dq:Image>
(24)             </dq:property>
(25)         </dq:DQproperties>
(26)     </rdf:Description>
(27)</rdf:Description>
(28) </rdf:RDF>

```

We have chosen to represent quality information as RDF properties. Specifically, in our DQF files, a property is defined by:

- a *name*, representing the name of the property;
- an *info*, representing the value of the property;
- a *comment* (optionally), informally describing the property;
- an *image* (optionally), specifying the name or the path of an image associated to the property.

In the shown DQF file, some general information are associated to the home page of a web site related to a researcher (lines 5-10). The web page contains information related to teaching activities, publications as well as a curriculum vitae of the researcher. In lines 19-24 a data quality property related to completeness is defined. Specifically, completeness information is associated to a curriculum vitae element; such an information is represented as an image showing the evolution in time of the completeness of the curriculum vitae. In the next section, a tool for visualizing DQF files is presented and a visualization of the described DQF file will be also shown.

## 6.2 Implementation Details

The *quality publisher* module, introduced in Section 5, has the role of visualizing quality information associated to Web pages. To such a scope, it has to interact with a *web data publishing module*, that has the specific task of visualizing data (see Figure6).

We have developed a prototype of the *quality publisher* module, which has been implemented by a software module called Data Quality Viewer (DQViewer). DQViewer has been designed as a plug-in of Microsoft's Internet Explorer (IE), which is a possible implementation of the *web data publishing module*.

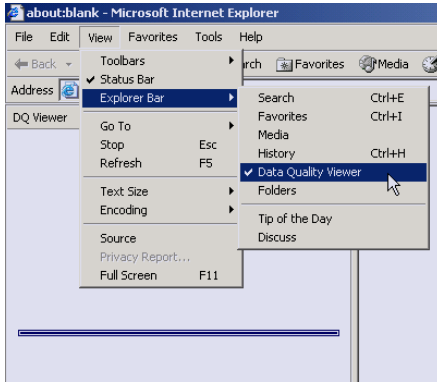
Once DQViewer is installed, it appears in the list of IE's plug-ins, as shown in Figure 7(a). Each time an HTML page is loaded, DQViewer is notified by an event, and a message is visualized in a window like the one shown in Figure 7(b).

Once the HTML page is completely loaded, DQViewer asks IE for the HTML source code, and verifies if the page has a Data Quality File associated to it. This verification is made by looking for a string:

```
<META DataQualityInfo='http://Nomefile.DQF'>
```

If the Data Quality File does not exist, a message is shown. Otherwise the DQF is loaded. At this point, the file is parsed in order to create the HTML





(a) IE's list of plugins



(b) Starting DQViewer

file that will finally visualize quality information. Specifically, the DQViewer performs the following steps:

- The DQF is read in order to identify which elements in the HTML page have quality information associated.
- On the basis of the different nesting levels in the original HTML page, a new HTML page is constructed containing a tree representing such elements.
- For each element in the tree, a table (**Dimension**, **Description**) is constructed. In this table, for each quality dimension, a value as well as optional comments are visualized.
- By selecting each of the tree elements, it is possible to visualize the associated quality information.

As an example, in Figure 7, the visualization of the DQF file described in the previous section is shown. In the figure, general information are associated to the home page element, which is the element at the higher granularity level. Such information are represented as a table; specifically, the table has two columns, namely: **Dimension** and **Description**. The **Dimension** values are the author and the last update of the home page; instead the **Description** values correspond to the info, comment and image values associated to the described properties (i.e., author and last update).

In the lower part of the DQViewer window, the tree structure of the page with reference to quality associations is shown. Specifically, the curriculum vitae element is nested within the home page element. By clicking on such an element, the associated quality information is visualized, as shown in Figure 8. The visualized dimension is **completability**; the **completability** value is 70%, the **t\_max** value (see Section 2) is January 31th 2001 and the evolution in time of the function  $C(t)$  is shown by an image.

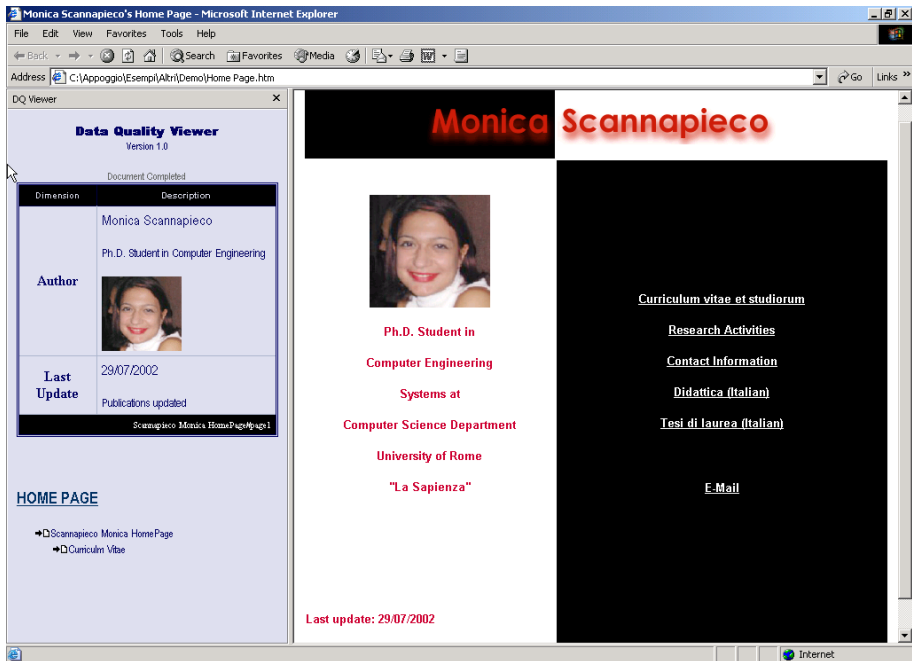


Fig. 7. Visualizing metadata with DQViewer.

## 7 Related Work

In the literature, methodological frameworks to manage data quality within single information systems [24] and cooperative information systems [6] have already been proposed; instead, as of our knowledge, a methodological proposal to manage data quality in WIS's has not yet been addressed.

Most of the work that is relative to both data quality and web regards the assessment of the quality of web sites [1,3,11]. In [19], a Quality Evaluation Method (QEM) is proposed to assess the quality of academic web sites, by proposing more than a hundred characteristics. This approach allows an evaluation of the accomplishment of the characteristics required by web sites. Our perspective is different, as we give a method to *declare* the quality of the published information, more than to evaluate it "a-posteriori".

In [17], the problem of the quality of web available information has been faced in order to select data with high quality coming from distinct sources: every source has to evaluate some pre-defined data quality parameters, and to make their values available through the exposition of metadata. The approach proposed in this article is complementary to the one proposed in [17]. In fact, on one hand, our approach suggests a framework for the management of quality information associated to data. On the other hand, our focus is not on selecting sources on the basis of the quality of the provided data, but on considering

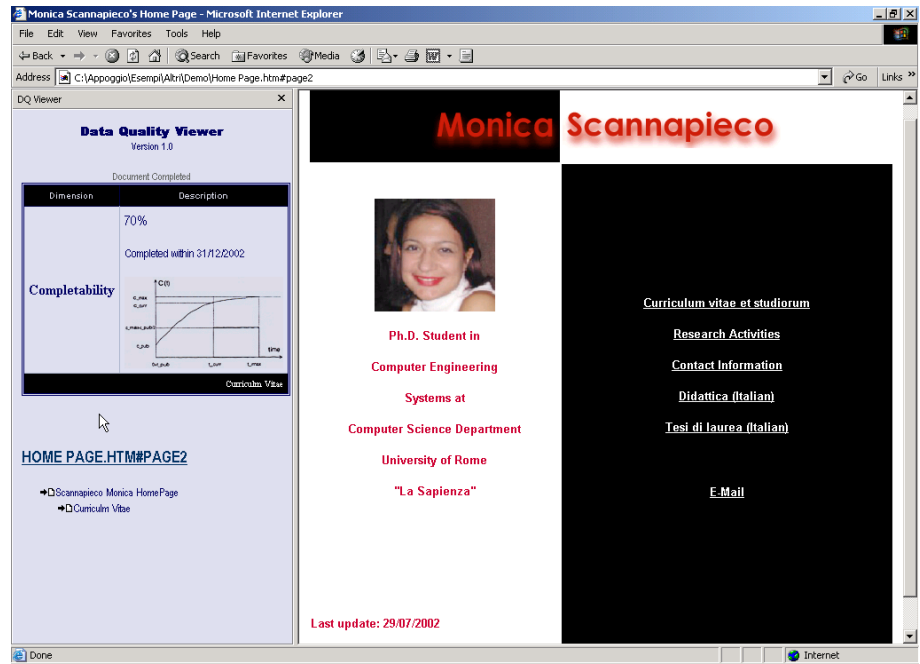


Fig. 8. Visualizing quality metadata with DQViewer.

the quality information of data published within a single site; in this case the published quality information can be detailed in a more specific way, including a dynamic characterization of dimensions.

## 8 Concluding Remarks

The data quality dimensions for data in web information systems proposed in the present paper are on one hand a subset of the most typical dimensions in data quality literature, and on the other hand provide an original view on these dimensions, since the dynamic evolution of web data is taken into account with information about the volatility and completability of web data.

The proposed data quality dimensions may be associated both to atomic and aggregated data in web pages. The main purpose is to provide clients and users accessing contents of web pages with a way to evaluate such an information with respect to its quality. The evaluation of data quality in WIS's may provide a way to guide users to select pages and to decide when to retrieve and make use of given information.

As for WIS's categories that could better benefit from the proposed ideas and architecture, we refer to the classification of Web sites give in [16]. According to this classification, four quadrants can be identified by drawing complexity of applications vs. complexity of data, and distinguishing HIGH and LOW complexity

levels for both applications and data. Specifically, the categories that could better benefit from the proposed approach are the two ones corresponding to HIGH data complexity, namely: (i) catalogue site (also called data intensive), with HIGH data complexity and LOW applications complexity and (ii) Web-Based Information Systems, with HIGH data and applications complexities. In fact, when dealing with huge amounts of data, it is more relevant and may become crucial to have an automated framework for the evaluation of their quality.

The proposed framework needs to be further refined in some directions. First of all, there is a need to study whether other dimensions traditionally proposed in data quality literature are applicable and useful in WIS's. In addition, the proposed approach associate quality attributes either to atomic elements or to aggregates. It is interesting to provide algorithms to derive quality information in aggregate web data from quality information associated to atomic elements. Another aspect which needs further investigation is the periodical nature of publication of information. For instance, the publication of course programs is periodical (i.e., yearly, or semester based), information about new degrees is published on a yearly base. Further research is needed to relate the periodical nature of information and data quality information about expiration and volatility of information.

**Acknowledgments.** This work is supported by MIUR, COFIN 2001 Project “DaQuinCIS - Methodologies and Tools for Data Quality inside Cooperative Information Systems” (<http://www.dis.uniroma1.it/~dq/>).

The authors would like to thank Massimo Mecella for important discussions about many issues addressed in this work.

## References

1. Atzeni P., Merialdo P., Sindoni G.: Web Site Evaluation: Methodology and Case Study. In Proceedings of DASWIS 2001: International Workshop on Data Semantics in Web Information Systems, Yokohama, Japan, 2001.
2. Ballou D.P., Pazer H.L.: Modeling data and process quality in multi-input, multi-output information systems, *Management Science*, vol.31, no.2, 1985.
3. Barnes S.J., Vidgen R.T.: Assessing the Quality of Auction Web Sites. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34), Maui, Hawaii, 2001.
4. Batini C., Lenzerini M., Navathe S.B.: A Comparative Analysis of Methodologies for Database Schema Integration, *ACM Computing Survey*, vol. 15, no.4, 1984.
5. Berners-Lee T., Hendler J., Lassila O.: The Semantic Web, *Scientific American*, May 2001.
6. Bertolazzi P., Scannapieco M.: Introducing Data Quality in a Cooperative Context. In Proceedings of the 6th International Conference on Information Quality (IQ'01), Boston, MA, USA, 2001.
7. Brickley D., Guha R.V. (eds.): RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, 23 January 2003.

8. Dasu T., Johnson T., Muthukrishnan S., Shkapienyuk V.: Mining Database Structure or How to Build a Data Quality Browser. In *Proceedings of the 2002 ACM/SIGMOD Conference*, Madison, WI, USA, 2002.
9. Gallhardas H., Florescu D., Shasha D., Simon E.: An Extensible Framework for Data Cleaning. In *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, San Diego, California, CA, 2000.
10. Garzotto F., Mainetti L., Paolini P.: Hypermedia Design, Analysis and Evaluation Issues. *Communication of the ACM*, vol. 58, no. 8, 1995.
11. Katerattanakul P., Siau K.: Measuring Information Quality of Web Sites: Development of an Instrument. In *Proceedings of the International Conference on Information Systems (ICIS 1999)*, Charlotte, North Carolina, USA, 1999.
12. Isakowitz T., Bieber M., Vitali F. (eds.): *Web Information Systems (Special Issue)*. *Communications of the ACM*, vol.41, no.7, 1998.
13. Isakowitz T., Kamis A., Koufaris M.: The Extended RMM Methodology for Web Publishing. Working Paper IS-98-18, Center for Research on Information Systems, University of Pennsylvania, Philadelphia, PA, USA, 1998.
14. Isakowitz T., Stohr E.A., Balasubramanian P.: RMM: a Methodology for Structured Hypermedia Design. *Communications of the ACM*, vol. 58, no. 8, 1995.
15. Lassila O., Swick R. R. (eds.): *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation, 22 February 1999.
16. Mecca G., Merialdo P., Atzeni A., Crescenzi V.: The (short) ARANEUS Guide to Web-Site Development. In *Proceedings of the Second International Workshop on the Web and Databases (WebDB'99) in conjunction with SIGMOD'99*, Philadelphia, Pennsylvania, USA, 1999.
17. Mihaila G., Raschid L., Vidal M.: Querying Quality of Data Metadata. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.
18. Naumann F.: *Quality-Driven Query Answering for Integrated Information Systems*, LNCS 2261, 2002.
19. Murugesan S., Deshpande Y. (eds.): *Web Engineering, Software Engineering and Web Application Development*. *Lecture Notes in Computer Science* vol. 2016, Springer Verlag, 2001.
20. Pipino L., Lee Y., Wang R. : Data Quality Assessment. *Communications of the ACM*, vol. 45, no. 4, 2002.
21. Redman T.C.: *Data Quality for the Information Age*. Artech House, 1996.
22. Tansell A., Snodgrass R., Clifford J., Gadia S., Segev A. (eds.): *Temporal Databases*. Benjamin-Cummings, 1993.
23. Wand Y., Wang R.Y.: Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM*, vol.39, no.11, 1996.
24. Wang R.Y.: A Product Perspective on Total Data Quality Management. *Communication of the ACM*, vol. 41, no. 2, 1998.
25. Wang R.Y., Storey V.C., Firth C.P.: A Framework for Analysis of Data Quality Research. *IEEE Transaction on Knowledge and Data Engineering*, vol. 7, no. 4, 1995.
26. Wang R.Y., Strong D.M.: Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, vol. 12, no. 4, 1996.

# Reasoning about Anonymous Resources and Meta Statements on the Semantic Web

Guizhen Yang<sup>1</sup> and Michael Kifer<sup>2</sup>

<sup>1</sup> Department of Computer Science and Engineering  
University at Buffalo, The State University of New York  
Buffalo, NY 14260-2000, U.S.A.  
`gzyang@CSE.Buffalo.EDU`

<sup>2</sup> Department of Computer Science  
Stony Brook University  
Stony Brook, NY 11794-4400, U.S.A.  
`kifer@CS.StonyBrook.EDU`

**Abstract.** Anonymous resources and meta statements are two of the more interesting features of RDF — an emerging standard for representing semantic information on the Web. Ironically, when RDF was standardized by W3C over three years ago [24], it came without a semantics. There is now growing understanding that a Semantic Web language without a semantics is an oxymoron, and a number of efforts are directed towards giving RDF a precise semantics [17,15,11]. In this paper we propose a simple semantics for anonymous resources and meta statements in F-logic [23] — a frame-based logic language, which is a popular formalism for representing and reasoning about semantic information on the Web [29,14,16,13,12].

The choice of F-logic (over RDF) as a basis for our semantics is motivated by the fact that F-logic provides a comprehensive solution for the problem of integrating frames, rules, inheritance, and deduction, and it has been shown to provide an effective inference service for RDF [13,28].

## 1 Introduction

RDF [24] was proposed as a standard for representing semantic information on the Web. Ironically, the specification of RDF did not formally define a semantics. Fortunately this peculiar situation is currently being rectified by a number of efforts [17,15,11].

While much of the RDF syntax is first-order in nature, *reification*, which is involved in expressing *meta statements* like “Tom believes that Alice said that RDF is a good idea”, is not. In fact, when left to its own devices, reification can lead to logical paradoxes. In Section 8 we explain why this problem does not arise in our framework. Related to this is the issue of anonymous resources, which in RDF are invoked to express statements such as “A person, called Ora Lassila, is the creator of RDF.” In this paper, we propose a simple model-theoretic semantics for both of these issues using F-logic [23] as the underlying formalism.

The choice of F-logic is motivated by several considerations. First, it has been a popular vehicle for ontology-based information mediation and representing semantic information on the Web whenever complex inference is required [20,22,29,14,16,12]. As such applications grow in complexity, the need for an inferencing service will increase. For instance, [13] envisions F-logic precisely as such a service for RDF. A number of initiatives (*e.g.*, RuleML<sup>1</sup>) have sprung up to promote the use of rule-based reasoning for processing semantic information on the Web, and F-logic and its derivatives are some of the languages being considered.

Our contention is that a model theory for RDF must be considered as part of a more general framework, because experience shows that semantics developed for a limited language like RDF might not generalize. For instance, rules and inheritance are some of the issues whose subtle influence is not apparent in the restricted setting of RDF [31]. In fact, we argue that the current proposal for the RDF model theory [15] has several weaknesses, such as *non-compositionality* (see Section 6), which might cause problems down the road. We also point out that there are at least two different useful notions of entailment for RDF graphs (see Section 5), but only one is currently reflected in the RDF model theory document [15].

The idea of embedding RDF into a larger theory is, of course, not new. Embedding RDF into F-logic was proposed in [13], and in [17] the same was done for KIF [19]. Both proposals are incomplete, however, as they do not address reification and anonymous resources — the main subject of the present paper. The embedding into F-logic *would have been* complete if F-logic, as described in [23], had support for these two features. We are rectifying this situation in the present work. Our proposed semantics is conceptually very simple<sup>2</sup> and is inspired by HiLog [10] — a logic language that provides a foundation for tractable higher-order logic programming — and by the treatment of anonymous object identities in  $\mathcal{F}$ LORA-2 [30,33] — a powerful frame-based language for knowledge representation and reasoning, which is based on F-logic, HiLog, and Transaction Logic [4,5]. By incorporating reification into the F-logic model theory we provide a model theory for reification in RDF and extend it with powerful meta-programming and inferencing capabilities, which F-logic is known for.

Embedding into F-logic also provides an immediate practical benefit. There are already F-logic based systems, such as  $\mathcal{F}$ LORA-2 [30,33] and TRIPLE [28], which support reification and have RDF handling capabilities. In particular,  $\mathcal{F}$ LORA-2 implements the proposed semantics and provides full support for frame-based representation, rules, inheritance, meta-programming, database updates, and more — all in a clean logical fashion. It has already been used in a number of projects ranging from data integration in neuroscience [21] to processing semistructured and semantic information on the Web [12].

This paper is organized as follows. Section 2 surveys the necessary background from F-logic and HiLog. Section 3 motivates the proposed extensions to

<sup>1</sup> <http://www.ruleml.org>

<sup>2</sup> While it is simple, it is not obvious, as a number of authors believed that F-logic does not generalize to deal with reification [7,13,28].

F-logic from the point of view of modeling using RDF. Section 4 formally treats the semantics of the proposed extensions. Sections 6 and 7 discuss the properties of the semantics introduced in Sections 4 and 5 and point out some problems with the current proposal for the RDF model theory [15]. Section 8 discusses the paradoxes that are often lurking in logical theories that support reification and explains why these problems are avoided in our framework. Section 9 concludes the paper.

## 2 Preliminaries

In this section we review the main ideas behind F-logic [23] and HiLog [10] — the two formalisms that form the basis for our proposed semantics.

### 2.1 F-Logic

F-logic is an extension of classical predicate logic which allows frame-based (or object-oriented) syntax, and has a natural model-theoretic semantics, and sound and complete proof theory.

F-logic uses Prolog *ground* (i.e., variable-free) terms to represent object identities (abbr., oids), e.g., `john` and `father(mary)`. Objects can have functional (single-valued), multivalued, or Boolean attributes, for example:

```
mary[spouse → john, children → {alice, nancy}].
mary[children → jack].
mary[married].
```

Here `spouse → john` is a *single-valued* attribute specification; it says that `mary` has an attribute `spouse`, whose value is a singleton oid `john`. The specification `children → {alice, nancy}` says that `children` is a *multivalued* attribute; its value in the context of the object `mary` is a set that *includes* the oids `alice` and `nancy`. We emphasize “includes” because a set does not need to be specified all at once. For instance, the second fact above says that `mary` has one additional child, `jack`. Note also that we usually omit the braces while specifying a singleton set. The last clause is an example of a Boolean attribute: it states that the value of `married` is true for the object `mary`.

While some attributes of an object can be specified explicitly as facts, other attributes can be defined using inference rules. For instance, we can derive `john[children → {alice, nancy, jack}]` with the help of the following inference rule:

$$X[\text{children} \rightarrow \{C\}] :- Y[\text{spouse} \rightarrow X, \text{children} \rightarrow \{C\}].$$

Here we adopt the usual Prolog convention that capitalized symbols denote variables, while symbols beginning with a lowercase letter denote constants.

F-logic objects can also have *methods*, i.e., functions that return a value or a set of values when appropriate arguments are provided. For instance,

```
john[grade(cs305,f2002) → 100, courses(f2002) → {cs305, cs306}].
```



says that `john` has a single-valued method, `grade`, whose value on the arguments `cs305` and `f2002` is 100, and a multivalued method `courses`, whose value on the argument `f2002` is a set of oids that contains `cs305` and `cs306`. As attributes, methods can also be defined using rules.

In addition, *class memberships* (e.g., `john:student`), *subclass relationships* (e.g., `student::person`), *types* (e.g., `person[name⇒string]`), and many other things can also be specified — both statically, as facts, and dynamically, via rules.

In the sequel, we will consider only multivalued attributes and ignore the rest of the features — the results of this paper extend straightforwardly to include class membership, subclass relationship, methods, types, etc.

## 2.2 HiLog

HiLog was introduced in [9,10] to provide a convenient syntax and tractable model theory to higher-order logic programming. The main highlights of this language are the variables that can range over both function and predicate symbols and a complete elimination of the barrier between predicate formulas and first-order terms. In this way, HiLog provides a natural syntax and semantics for reification: a statement can be a formula and an object at the same time.

We illustrate HiLog through examples. The simplest yet most unusual one is the definition of the standard Prolog meta-predicate `call`:

`call(X) :- X.`

In this example, HiLog does not distinguish between function terms and atomic formulas: the same variable can range over both. Therefore, one can reify statements and reason about them in the same language. For instance, we can state that Bob believes (among other things) that Mary likes RDF as follows:

`believes(bob,likes(mary,rdf)).`

and then state that whatever Bob believes is true:

`X :- believes(bob,X).`

Variables can also range over function symbols, as in `X(Y,a)`. A query of the form `?- p(X),X,X(Y,X)` is well within the boundaries of HiLog. The syntax for HiLog terms also extends that of classical logic. For instance, `g(X)(f(a,X),Y)(b,Y)` is perfectly fine. Of course, such powerful syntax should be used sparingly, but people have found many important uses for these features. For instance, the following simple program defines transitive closure of *any* binary relation. The program defines a higher-order predicate constructor `closure`, which takes a binary predicate as a parameter and yields a first-order predicate:

`closure(Pred)(X,Y) :- Pred(X,Y).`  
`closure(Pred)(X,Y) :- Pred(X,Z), closure(Pred)(Z,Y).`

Here  $\text{Pred}$  is a variable that ranges over predicate symbols. When it is bound to a particular symbol, say  $\text{parent}$ , the above program would compute the relation  $\text{closure}(\text{parent})$ , *i.e.*, the ancestor relation. More examples of this kind of programming are found in [10].

When combined with F-logic, HiLog enables powerful meta-programming features [23,30,33]. For instance, we can define an attribute, `methods`, whose value for any object is the set of the names of unary and binary single-valued methods defined for that object:

$$\begin{aligned} X[\text{methods} \rightarrow \{M\}] &:- X[M(A) \rightarrow V]. \\ X[\text{methods} \rightarrow \{M\}] &:- X[M(A1,A2) \rightarrow V]. \end{aligned}$$

Since the main focus of this paper is reification and anonymous identity, in the rest of this paper we will consider only the subset of HiLog that enables reification and combine it with F-logic. Indeed, the use of HiLog to enhance meta-programming in F-logic was discussed in [23,30], but its use for supporting reification has not been considered.

### 3 Supporting RDF in F-Logic

It was argued in [13] that F-logic is a natural formalism to provide semantics and inference service for RDF(S) [24]. However, some important aspects, such as anonymous resources, containers, and reification were left out because the original F-logic [23] did not support them. In this section we illustrate on a number of examples that all these features can be supported by slightly extending the logic with *anonymous ID symbols* and *reified statements*. Formal treatment of this extension is given in Sections 4 and 5.

#### 3.1 RDF Data Model

First, we briefly recall the RDF data model, which can be represented as triples, as a graph, or in XML [24]. These representations have equivalent meaning but here we will mainly use triple syntax and XML to present RDF data (see [24] for more details). An RDF document is a finite set of statements of the form  $\{\text{predicate, subject, object}\}$  [24], where *predicate* is a *property*, *subject* is a *resource*, and *object* is a resource or a literal. A set of RDF statements can also be viewed as a *directed labeled graph*, where the vertexes are the resources and the literals, and a triple  $\{p, s, o\}$  represents an arc from  $s$  to  $o$ , labeled with  $p$ .

A resource describes a real or conceptual entity (*e.g.*, John Doe). Typically, resources are represented as URIs [2]. But they can also be *anonymous* (*e.g.*, *some-one*). For instance, the URI, <http://www.w3.org/TR/REC-rdf-syntax>, represents the abstract concept of RDF itself. URIs are considered as logical constants referring to entities. In this paper when we use triple syntax to present RDF data, we will enclose URIs using a pair of square brackets, *e.g.*, [<http://foo.org/TheBulb>]. For simplicity we will ignore aspects of meaning encoded in particular URI forms and just treat URI references as simple names.

A property is a predicate that specifies a *binary* relationship (e.g., *parent*). In RDF, properties form a subset of resources and are also represented using URIs. Property names must be associated with a schema or vocabulary. Usually we use QName syntax (see XML Namespaces specification [6] for more details) to denote property names, e.g., *rdf:type*, where *rdf* can be defined as an XML namespace prefix referring to <http://www.w3.org/1999/02/22-rdf-syntax-ns#>, the URI of the (conceptual) vocabulary of RDF properties. For simplicity we sometimes omit the namespace prefix and just write the property name in triple syntax, e.g., [*inventor*]. How XML namespaces are resolved is orthogonal to the results of this paper.

Finally, literals are constants in some primitive data types, such as *string* or *number*. The meaning of a literal is principally determined by its character string: it either refers to the value mapped from the string by the associated data type, or the literal itself when no data type is provided. In all of our examples here literals are just character strings. We normally denote literals using a pair of double quotes in triple syntax, e.g., "Thomas Edison".

However, there is slightly a deviation to represent RDF in F-logic. For simplicity in illustration, we will use atomic constants (sometimes enclosed in single quotes) to denote URIs, properties, and literals, e.g., '<http://foo.org/TheBulb>', '*rdf:type*', '*Thomas Edison*'. In real implementation the semantics of these names and notations can be resolved by a separate procedure.

### 3.2 Anonymous Object Identity

Representation of RDF statements with named resources in F-logic is straightforward. For instance, the following sentence

*Thomas Edison is the inventor of the bulb (represented by a resource with the URI <http://foo.org/TheBulb>).*

can be represented as a triple

{[*inventor*], [<http://foo.org/TheBulb>], "Thomas Edison" }

where the notation [ref] denotes the resource identified by the URI ref and a string enclosed in double quotes denotes a literal. In F-logic, the same statement is written like this:

'<http://foo.org/TheBulb>'[*inventor*  $\rightarrow$  '*Thomas Edison*'].

One difficulty arises when we try to translate RDF statements that include *anonymous resources*, i.e., resources that are not named explicitly. This kind of resources are involved in expressing statements such as

*Someone, named Thomas Edison, born in 1847, is the inventor of the resource <http://foo.org/TheBulb>.*

The intent here is to make a structured resource *without a known object ID* and state that it has two properties, *name* and *born*, with the above values. In RDF, this sentence would be represented using triple syntax as follows:

```
{[name], [X], "Thomas Edison" }
{[born], [X], "1847" }
{[inventor], [http://foo.org/TheBulb], [X]}
```

Here [X] represents an anonymous resource.

Objects with anonymous IDs were not envisioned in the original work on F-logic [23], but were introduced in  $\mathcal{F}$ LORA-2 [33,30] — our implementation of F-logic that extends it with many additional concepts. To represent such objects,  $\mathcal{F}$ LORA-2 uses a special symbol,  $\_ \#$ , called an *unnumbered anonymous ID symbol*, and another countable set of symbols,  $\_ \#1$ ,  $\_ \#2$ , ..., etc., called *numbered anonymous ID symbols*. The intended meaning (which is formalized in Section 4) is that each occurrence of  $\_ \#$  denotes a distinct object ID that does not occur anywhere else in the program. All occurrences of the same numbered anonymous ID symbol, e.g.  $\_ \#1$ , within the same scope are treated as representing the same object ID, but this ID is distinct from any other ID used elsewhere in the program (including the occurrences of  $\_ \#1$  in a different scope). The notion of scope is formalized in Section 4, but for our current purposes let us assume that the scope of numbered anonymous ID symbols extends over the entire clause (where each clause is terminated with a “.” and comma represents the conjunction). Thus, the above statement can be represented in  $\mathcal{F}$ LORA-2 as follows:

```
'http://foo.org/TheBulb'[inventor  $\rightarrow$   $\_ \#1$ ],
 $\_ \#1$ [name  $\rightarrow$  'Thomas Edison', born  $\rightarrow$  '1847'].
```

Note that here the two occurrences of  $\_ \#1$  are within the same clause and thus the same scope. So they refer to the same object. If we want to state that *someone invented the bulb and someone called Thomas Edison was born in 1847*, then we could write

```
'http://foo.org/TheBulb'[inventor  $\rightarrow$   $\_ \#$ ],
 $\_ \#$ [name  $\rightarrow$  'Thomas Edison', born  $\rightarrow$  '1847'].
```

Here we use unnumbered anonymous ID symbols and, even though they occur within the same scope, they represent different objects.

Anonymous resources are also frequently used to represent *containers* in RDF. For example, the following sentence

*The committee of Fred, Wilma, and Dino approved the resolution.*

can be expressed using the *Bag* container of RDF and would be written in the RDF syntax as follows:

```

<rdf:RDF>
  <rdf:Description about="http://xyz.org/resolution">
    <approvedBy>
      <rdf:Bag>
        <rdf:li resource="http://xyz.org/members/Fred" />
        <rdf:li resource="http://xyz.org/members/Wilma" />
        <rdf:li resource="http://xyz.org/members/Dino" />
      </rdf:Bag>
    </approvedBy>
  </rdf:Description>
</rdf:RDF>

```

In  $\mathcal{FLORA-2}$ , the same sentence can be represented as follows:<sup>3</sup>

```

_#1[
  'rdf:type' → 'rdf:Bag',
  'rdf:_1' → 'http://xyz.org/members/Fred',
  'rdf:_2' → 'http://xyz.org/members/Wilma',
  'rdf:_3' → 'http://xyz.org/members/Dino'
],
'http://xyz.org/resolution'[approvedBy → _#1].

```

Again, here the two occurrences of `_#1` are within the same scope and thus represent the same object. The first occurrence represents a *Bag* object and the second occurrence refers to this object.

### 3.3 Reified Statements

Reification in RDF is used to make meta statements, *i.e.*, statements about statements. Since statements are formulas, making statements about them means that formulas must be somehow treated as objects. To represent the following statement

*Someone named John Doe believes that a person, called Thomas Edison, invented the bulb (resource `http://foo.org/TheBulb`).*

using RDF triple syntax one would have to write a rather convoluted expression below:

<sup>3</sup> F-logic provides other, more natural ways of expressing the same thing, like, for instance, `'http://xyz.org/resolution' [approvedBy → { 'http://xyz.org/members/Fred', 'http://xyz.org/members/Wilma', 'http://xyz.org/members/Dino' }]`. The cumbersome statement above is intended to mimic the structure of the corresponding RDF statement.

```

{[type], [X], [RDF:Statement]}
{[predicate], [X], [inventor]}
{[subject], [X], [http://foo.org/TheBulb]}
{[object], [X], [Y]}
{[name], [Y], "Thomas Edison"}
{[name], [Z], "John Doe"}
{[believes], [Z], [X]}

```

Here a new, anonymous resource  $X$  is used as a *referent* to the following reified statement

*A person, called Thomas Edison, invented the bulb.*

This is expressed by the first four triples, which say that: (1)  $X$  represents an RDF statement; (2) its predicate is *inventor*; (3) its subject is the URI <http://foo.org/TheBulb>; and (4) its object is another anonymous object  $Y$ . In the fifth triple we say that this latter object has property *name* with the value "Thomas Edison". The sixth statement says that there is an anonymous object  $Z$  with property *name* whose value is "John Doe". Finally, the last statement says that object  $Z$  has property *believes* with value  $X$  — the anonymous resource that represents the reified statement that *a person, called Thomas Edison, invented the bulb*.

In our extension to F-logic this statement can be modeled in the following much more natural way:

```

-#[
  name → 'John Doe',
  believes →
  ('http://foo.org/TheBulb'[inventor → #1], #1[name → 'Thomas Edison'])
].

```

Note that here the formula `'http://foo.org/TheBulb'[inventor → #1]` *itself* is an object — not some other object ID that refers to this formula. We will argue in Section 7 that this syntax and its corresponding semantics is superior to that of the current proposal for RDF model theory [15]. It also permits more interesting reasoning to be easily performed over reified statements (see Section 7).

We should note that the above syntax is *not* the actual syntax of  $\mathcal{FLORA-2}$  [33] and is also quite different from the F-logic syntax as described in [23]. We decided to depart from that syntax in this paper in order to avoid the distraction that would result from the introduction of additional features of F-logic and  $\mathcal{FLORA-2}$ , and in order to simplify the formal development of the model theory in Section 4. In fact, the actual syntax of  $\mathcal{FLORA-2}$  is much richer, which allows to write the above and the earlier sentences more succinctly:

```

-#[
  name → 'John Doe',
  believes →
  ${'http://foo.org/TheBulb'[inventor → #1[name → 'Thomas Edison']}]
].

```

That is, in the actual  $\mathcal{F}\text{LORA-2}$  syntax, reification is specified using the  $\$\{...\}$  construct and the statement inside of  $\$\{...\}$  is a shorthand for the conjunction of two F-logic statements: `'http://foo.org/TheBulb'[inventor  $\rightarrow$  #1]` and `#1[name  $\rightarrow$  'Thomas Edison']`. The interested reader is referred to [33] for the details.

In Section 4, we discuss the notions of RDF graph entailment and equivalence and show that there are at least two such notions, both useful, but only one is currently considered by the RDF model theory proposal [15].

## 4 Formal Syntax and Semantics

In this section we formally define the syntax and semantics of an F-logic language extended with anonymous identity and reification. We will continue to call this extension “F-logic” in order to avoid introducing yet another name.

To simplify the exposition, we focus on a subset of the new F-logic syntax. The only kind of atoms we consider here is in the form of  $\mathbf{o}[\mathbf{m} \rightarrow \mathbf{v}]$ , which specifies that the object  $\mathbf{o}$  has a multivalued method,  $\mathbf{m}$ , that returns some set of objects, which contains  $\mathbf{v}$  as a member. The symbols  $\mathbf{o}$ ,  $\mathbf{m}$ , and  $\mathbf{v}$  are F-logic terms (to be defined below); they represent the ID of an object, a method, and a value of the method, respectively. In a program, these terms can contain variables in which case they would represent a parameterized collection of objects — one object per variable instantiation. This design makes meta-programming in F-logic as natural as querying.

An F-logic language  $\mathcal{L}$  consists of a set of *constants*,  $\mathcal{C}$ ; a set of *variables*,  $\mathcal{V}$ ; an *unnumbered anonymous ID* symbol, `_#`; *numbered anonymous ID* symbols, `_#1`, `_#2`, ... (for each positive integer); *connectives* including  $\neg$ ,  $\vee$ ,  $\wedge$ , and  $\leftarrow$ ; *quantifiers* including  $\exists$  and  $\forall$ ; and *auxiliary symbols*, such as comma, parentheses, and brackets. In defining the semantics for F-logic programs, we will assume an *a priori* fixed F-logic language  $\mathcal{L}$ .

Intuitively, an occurrence of an unnumbered anonymous ID symbol implies a *distinct* object that is different from any object represented by any other term. Moreover, two occurrences of `_#` represent two distinct objects. Numbered anonymous ID symbols are similar, but with a twist. Different occurrences of `_#N` and `_#M`, where  $N \neq M$ , represent distinct objects. However, different occurrences of `_#N` (with the same number  $N$ ) within *the same scope* refer to the same object. This meaning of “scope” will be made precise later, when we give a formal semantics.

Formally, F-logic *terms* and *atoms* are constructed inductively as follows. The idea of this construction is borrowed from HiLog [9,10] and is extended to accommodate reification of F-logic atoms (statements).

**Definition 1 (Terms and Atoms).** *Given an F-logic language  $\mathcal{L}$ , the terms and atoms are defined inductively as follows:*

- Any constant  $c \in \mathcal{C}$  is a term.
- Any variable  $X \in \mathcal{V}$  is a term.

- Unnumbered or numbered ID symbols,  $\_ \#$ ,  $\_ \#1$ ,  $\_ \#2$ , ..., are terms.
- If  $t$  is a term and  $t_1, \dots, t_n$  are terms, then  $t(t_1, \dots, t_n)$  is a term.
- Any term in any of the above forms is called a HiLog term. Note that expressions like  $p(a, p(X)(Y))(W, q)(f(X))$  are HiLog terms according to this definition.
- If  $o$ ,  $m$ , and  $v$  are terms, then  $o[m \rightarrow v]$  is a term, also called an F-logic term.
- If  $A_1$  and  $A_2$  are terms, then their conjunction,  $A_1 \wedge A_2$ , is a term.

### Definition 2 (Flat Formula).

**Reification:** Any term is also a formula. In particular, any HiLog atom or F-logic atom is called an atomic flat (HiLog or F-logic) formula.

**Composition:** If  $\phi$  and  $\psi$  are flat formulas, then so are

- $\neg \phi$
- $\phi \vee \psi$  and  $\phi \wedge \psi$
- $\phi \leftarrow \psi$ , which is defined to be just a shortcut for  $\phi \vee \neg \psi$
- $\exists X \phi$  and  $\forall X \phi$ , where  $X \in \mathcal{V}$  is a variable.

Note that both F-logic and predicate terms (or, more precisely, HiLog terms) are atomic formulas in our language. In this way, both relational and object-oriented programming are supported. The last two cases in Definition 1 and the first case in Definition 2 state that atomic formulas as well as their conjunctions are terms and so are first-class objects in the language. In particular, as we shall see, variables can range over such formulas. This makes the syntax higher-order and provides support for reification. However, the semantics needs to be carefully defined so as to stay tractable and first-order in the sense of [9,10].

Now we will define interpretations, *i.e.*, the semantic structures that give meanings to F-logic formulas and programs. Our definitions follow the standard convention except that we need to take special care of the anonymous ID symbols and reified statements. To this end, we first have to define the domain of interpretations. In classical logic programming, the domain is typically the set of all ground terms in the language, which is called the Herbrand universe. In our case, however, the domain must also include the constants that are used to interpret the anonymous ID symbols. This idea is formalized next.

**Definition 3 (Augmented Herbrand Universe).** Let  $\mathcal{L}$  be an F-logic language,  $\mathcal{C}$  be the set of constants in  $\mathcal{L}$ , and  $\mathcal{D}$  be a countable set of constants that is disjoint from  $\mathcal{C}$ . We shall call  $\mathcal{D}$  an anonymous domain, since it will be used to interpret the anonymous ID symbols. The augmented Herbrand universe of  $\mathcal{L}$  with respect to  $\mathcal{D}$ , denoted  $\mathcal{HU}(\mathcal{D})$ , is the set of all terms (see Definition 1) constructed using the constants in  $\mathcal{C} \cup \mathcal{D}$ . Note that in constructing the terms for  $\mathcal{HU}(\mathcal{D})$ , variables and anonymous ID symbols are excluded. Such variable-free terms are called ground.

**Definition 4 (Interpretation).** Given an F-logic language  $\mathcal{L}$ , an interpretation  $\mathcal{I}$  is a pair  $(\mathcal{D}, \mathcal{S})$ , where



- $\mathcal{D}$  is an anonymous domain, i.e., a countable set of constants that is disjoint from  $\mathcal{C}$  (the set of all constants in  $\mathcal{L}$ ).
- $\mathcal{S}$  is a subset of  $\mathcal{HU}(\mathcal{D})$ , the augmented Herbrand universe of  $\mathcal{L}$  with respect to  $\mathcal{D}$ . Moreover,  $\mathcal{S}$  is allowed to contain atomic formulas only. Intuitively,  $\mathcal{S}$  represents “what is true” in  $\mathcal{I}$ .<sup>4</sup>

The above definition differs from those used in classical logic programming, HiLog, and the original F-logic in its use of the anonymous domain. One significant impact of this domain is that the Herbrand universes of different models are different when they use different anonymous domains. To be more precise, the ground terms that are constructed using the constants in  $\mathcal{C}$  are the same in all Herbrand universes, but the terms that involve the constants from anonymous domains may not be shared. In classical theory of logic programming all interpretations have the same domain — the Herbrand universe. Our definitions reduce to classical ones when there are no anonymous ID symbols and thus no anonymous domains.

Since interpretations can have different domains, there are many ways to compare interpretations. One possibility is by set inclusion. However, it only makes sense for interpretations over the same domain. Another alternative involves domain mapping and domain isomorphism. Formally, we have the following definitions.

**Definition 5 (Domain Mapping).** Let  $\mathcal{D}_1$  and  $\mathcal{D}_2$  be two anonymous domains with respect to an F-logic language  $\mathcal{L}$  and let  $\mathcal{C}$  be the set of constants in  $\mathcal{L}$ . Then any function  $\tau: \mathcal{D}_1 \rightarrow \mathcal{D}_2$  is called an anonymous domain mapping. Any function  $\lambda: \mathcal{D}_1 \rightarrow \mathcal{D}_2 \cup \mathcal{C}$  is called an augmented domain mapping.

For any  $\mathbf{t} \in \mathcal{HU}(\mathcal{D}_1)$ ,  $\tau(\mathbf{t})$  is a term obtained from  $\mathbf{t}$  by simultaneously replacing every constant  $\mathbf{d} \in \mathcal{D}_1$  with  $\tau(\mathbf{d})$  and leaving the constants in  $\mathcal{C}$  intact. For any  $\mathcal{S} \subseteq \mathcal{HU}(\mathcal{D}_1)$ ,  $\tau(\mathcal{S}) = \{\tau(\mathbf{x}) \mid \mathbf{x} \in \mathcal{S}\}$ .  $\lambda(\mathbf{t})$  and  $\lambda(\mathcal{S})$  are defined similarly.

**Definition 6 (Ordering and Isomorphism).** Let  $\mathcal{D}_1$  and  $\mathcal{D}_2$  be two anonymous domains, and  $\mathcal{I}_1 = (\mathcal{D}_1, \mathcal{S}_1)$  and  $\mathcal{I}_2 = (\mathcal{D}_2, \mathcal{S}_2)$  be two interpretations.

- We write  $\mathcal{I}_1 \preceq \mathcal{I}_2$  iff there is an 1-1 anonymous domain mapping  $\tau: \mathcal{D}_1 \rightarrow \mathcal{D}_2$  such that  $\tau(\mathcal{S}_1) \subseteq \mathcal{S}_2$ . We will write  $\mathcal{I}_1 \prec \mathcal{I}_2$  if  $\tau(\mathcal{S}_1) \subsetneq \mathcal{S}_2$ .
- $\mathcal{I}_1$  is isomorphic to  $\mathcal{I}_2$ , denoted  $\mathcal{I}_1 \simeq \mathcal{I}_2$ , iff  $\mathcal{I}_1 \preceq \mathcal{I}_2$  and  $\mathcal{I}_2 \preceq \mathcal{I}_1$ .

Before we can give semantics to formulas that contain anonymous ID symbols, we need to introduce one more notion, the *scoped formulas*. Recall from Section 3 that the intended meaning of a numbered anonymous ID symbol is that two different occurrences of the same symbol within the scope of the same

<sup>4</sup> Note that in classical logic programming an interpretation would contain a subset of the *Herbrand base* — a set of atomic formulas (which is distinct from the set of terms that comprises the Herbrand universe). However, in our case (as in HiLog), atomic formulas are reified and thus the Herbrand base and the Herbrand universe are the same.

rule denote the same object; otherwise, they potentially refer to different objects. The notion of scoped formulas allows us to extend this idea to more general types of formulas.

**Definition 7 (Scoped Formula).**

- If  $\phi$  is a flat formula, then  $\{ \phi \}$  is a scoped formula.
- If  $\psi$  and  $\xi$  are scoped formulas, then so are  $\psi \vee \xi$  and  $\psi \wedge \xi$ .

Note that our definition of scoped formulas is such that the scoping braces,  $\{ \dots \}$ , always apply only to the top level conjuncts and disjuncts. For instance,  $\{ \_ \#1[\text{loves} \rightarrow \text{mary}] \} \wedge \{ \exists X(\_ \#1[\text{child} \rightarrow X]) \}$  is a valid scoped formula whereas  $\{ \_ \#1[\text{loves} \rightarrow \text{mary}] \} \wedge \exists X(\_ \#1[\text{child} \rightarrow X])$  is not. Although we could define even more general scoping rules, including nested scoping, the practical utility of this complication is unclear and we will not introduce it in this paper.

An *F-logic program* is a finite collection of *scoped rules* where all variables are universally quantified. A rule has the following form:

$$\{ \forall (A_1 \wedge \dots \wedge A_m \leftarrow B_1 \wedge \dots \wedge B_n) \}$$

where  $m \geq 1, n \geq 0$ ,  $A_i$  ( $1 \leq i \leq m$ ) and  $B_j$  ( $1 \leq j \leq n$ ) are atoms, and only the rule head (the  $A_i$ 's) is allowed to have anonymous ID symbols. Note that each rule is a scoped formula where the scope is the entire rule. Thus an F-logic program can be thought of as a scoped formula formed by conjoining all the scoped formulas corresponding to the rules.

Before proceeding, we would like to point out that our theory of anonymous identity, which will be introduced below, is not restricted to clauses of the above form. We choose to limit our attention to rules without negation in the body in order to focus on the problem of modeling of blank nodes in RDF. The theory presented herein can be readily extended to F-logic programs with negation and inheritance by combining it with the semantics described in [31,32].

Following the standard convention, we will omit universal quantifiers in the rules and since the scope is the entire rule we will omit the scoping braces as well. Thus, rules will be simply written as follows:

$$A_1, \dots, A_m \leftarrow B_1, \dots, B_n$$

We will continue to use the convention from Section 2 whereby uppercase names denote variables and lowercase names denote constants. A rule with an empty body is called a *fact*. When writing down the facts, we will omit the implication symbol and simply show the head.

**Definition 8 (Skolemization).** Let  $\phi$  be a scoped formula and  $\mathcal{D}$  be an anonymous domain. A *skolemization* of  $\phi$  with respect to  $\mathcal{D}$ , denoted  $\Pi_{\mathcal{D}}(\phi)$ , is a formula obtained as follows:

- If  $\phi = \psi \vee \varphi$  such that  $\phi$  and  $\varphi$  are scoped formulas, then  $\Pi_{\mathcal{D}}(\phi) = \Pi_{\mathcal{D}_1}(\psi) \vee \Pi_{\mathcal{D}_2}(\varphi)$ , where  $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$  and  $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$ .

- If  $\phi = \psi \wedge \varphi$  such that  $\phi$  and  $\varphi$  are scoped formulas, then  $\Pi_{\mathcal{D}}(\phi) = \Pi_{\mathcal{D}_1}(\psi) \wedge \Pi_{\mathcal{D}_2}(\varphi)$ , where  $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$  and  $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$ .
- If  $\phi = \{ \psi \}$ , where  $\psi$  is a flat formula, then  $\Pi_{\mathcal{D}}(\phi) = \text{skolem}(\mathcal{D}, \psi)$ , where  $\text{skolem}(\mathcal{D}, \psi)$  is defined as follows:
  - Different occurrences of the same numbered ID symbol are mapped into the same constant in  $\mathcal{D}$ , but different numbered anonymous ID symbols ( $\_ \# \mathbf{M}$  and  $\_ \# \mathbf{N}$ , where  $\mathbf{M} \neq \mathbf{N}$ ) in  $\psi$  are mapped to distinct constants in  $\mathcal{D}$ .
  - Every occurrence of the unnumbered anonymous ID symbol,  $\_ \#$ , in  $\psi$  is mapped to a distinct constant in  $\mathcal{D}$  (i.e., different from the constants chosen for the numbered ID symbols).

Note that, in a flat formula, each occurrence of the *unnumbered* anonymous ID symbol  $\_ \#$  is mapped to a different element of  $\mathcal{D}$ , but all occurrences of the same *numbered* anonymous ID symbol, say  $\_ \# 1$ , are mapped to the same distinct element of  $\mathcal{D}$ . Also observe that two occurrences of the same numbered ID symbol, say  $\_ \# 1$ , under different scopes are mapped to different elements. For instance, consider a scoped formula  $\{ \phi \} \wedge \{ \psi \}$ . Then occurrences of  $\_ \# 1$  in  $\phi$  are going to be skolemized differently from those in  $\psi$ . This fact is a consequence of partitioning  $\mathcal{D}$  into disjoint  $\mathcal{D}_1$  and  $\mathcal{D}_2$  in the first two cases of Definition 8.

Another way of looking at  $\Pi_{\mathcal{D}}(\phi)$  is that it is just like a flat formula, with no scope and no anonymous ID symbols, but some of the symbols in that formula might come from the anonymous domain  $\mathcal{D}$ . The following example illustrates skolemization.

Let  $\mathcal{D} = \{\mathbf{o1}, \mathbf{o2}, \mathbf{o3}, \mathbf{o4}, \dots\}$  be an anonymous domain and  $P$  be the following simple F-logic program:

$\_ \# [\_ \# \rightarrow \_ \# 1], \_ \# 1[\text{name} \rightarrow \text{mary}].$   
 $\_ \# 1[\text{name} \rightarrow \text{'Ora Lassila'}], \text{'RDF'}[\text{creator} \rightarrow \_ \# 1].$

Note that according to our definition of F-logic programs, the above program is a shortcut for the following scoped formula:

$$\begin{aligned} & \{ \_ \# [\_ \# \rightarrow \_ \# 1] \wedge \_ \# 1[\text{name} \rightarrow \text{mary}] \} \\ & \quad \wedge \\ & \{ \_ \# 1[\text{name} \rightarrow \text{'OraLassila'}] \wedge \text{'RDF'}[\text{creator} \rightarrow \_ \# 1] \} \end{aligned}$$

If we map the first occurrence of  $\_ \#$  to  $\mathbf{o1}$ , its second occurrence to  $\mathbf{o4}$ , both occurrences of  $\_ \# 1$  in the first rule to  $\mathbf{o2}$ , and both occurrences of  $\_ \# 1$  in the second rule to  $\mathbf{o3}$ , then we obtain the following skolemization  $\Pi_{\mathcal{D}}(P)$ :

$\mathbf{o1}[\mathbf{o4} \rightarrow \mathbf{o2}], \mathbf{o2}[\text{name} \rightarrow \text{mary}].$   
 $\mathbf{o3}[\text{name} \rightarrow \text{'Ora Lassila'}], \text{'RDF'}[\text{creator} \rightarrow \mathbf{o3}].$

A different way of mapping anonymous ID symbols to the constants in  $\mathcal{D}$  (e.g., where the first occurrence of  $\_ \#$  is mapped to  $\mathbf{o4}$  and the second to  $\mathbf{o1}$ ) would lead to a different skolemization of  $P$ . Both mappings are valid skolemizations, however, as they satisfy the conditions of Definition 8.

We can now define what it means to be a model of a given scoped formula.

**Definition 9 (Model).** Let  $\mathcal{I} = (\mathcal{D}, \mathcal{S})$  be an interpretation and  $\phi$  be a scoped formula. Then  $\mathcal{I}$  is a model of  $\phi$ , denoted  $\mathcal{I} \models \phi$ , if and only if there is a skolemization,  $\psi = \Pi_{\mathcal{D}}(\phi)$ , of the formula  $\phi$  such that  $\mathcal{S} \models \psi$ , where  $\mathcal{S} \models \psi$  is defined in the classical sense:

- If  $\psi$  is an atom, then  $\mathcal{S} \models \psi$  iff  $\psi \in \mathcal{S}$ .
- If  $\psi = \neg \varphi$ , then  $\mathcal{S} \models \psi$  iff it is not the case that  $\mathcal{S} \models \varphi$ .
- If  $\psi = \varphi \vee \xi$ , then  $\mathcal{S} \models \psi$  iff either  $\mathcal{S} \models \varphi$  or  $\mathcal{S} \models \xi$ .
- If  $\psi = \varphi \wedge \xi$ , then  $\mathcal{S} \models \psi$  iff  $\mathcal{S} \models \varphi$  and  $\mathcal{S} \models \xi$ .
- If  $\psi = \exists X \varphi$ , then  $\mathcal{S} \models \psi$  iff there is  $t \in \mathcal{HU}(\mathcal{D})$  (a term in the augmented Herbrand universe) such that  $\mathcal{S} \models \psi[X/t]$ , where  $\psi[X/t]$  denotes the formula obtained from  $\psi$  by substituting  $t$  for all free occurrences of the variable  $X$ .
- If  $\psi = \forall X \varphi$  then  $\mathcal{S} \models \psi$  iff for all  $t \in \mathcal{HU}(\mathcal{D})$ ,  $\mathcal{S} \models \psi[X/t]$ .

**Lemma 1.** If  $\mathcal{I} \models \phi$  and  $\mathcal{I} \simeq \mathcal{J}$ , then  $\mathcal{J} \models \phi$ . Thus, the set of models of a formula is closed under isomorphism.

*Proof.* Let  $\mathcal{I} = (\mathcal{D}_1, \mathcal{S}_1)$  and  $\mathcal{J} = (\mathcal{D}_2, \mathcal{S}_2)$ . Since  $\mathcal{I} \models \phi$ , it follows that there is a skolemization,  $\psi_1 = \Pi_{\mathcal{D}_1}(\phi)$ , of the formula  $\phi$  such that  $\mathcal{I} \models \psi_1$ . Because  $\mathcal{I} \simeq \mathcal{J}$ , there is an 1-1 anonymous domain mapping  $\tau: \mathcal{D}_1 \rightarrow \mathcal{D}_2$  such that  $\tau(\mathcal{S}_1) = \mathcal{S}_2$ . Let  $\psi_2 = \tau(\psi_1)$ . By structural induction, we can show that if  $\mathcal{I} \models \psi_1$ , then  $\mathcal{J} \models \psi_2$ . It can be easily shown using Definition 8 that  $\psi_2$  is a valid skolemization,  $\psi_2 = \Pi_{\mathcal{D}_2}(\phi)$ , of the formula  $\phi$ . Therefore  $\mathcal{J} \models \phi$ .

We can now develop a fixpoint model theory analogously to the classical theory of logic programming. This would provide one computational model for the proposed semantics. Let  $P$  be an F-logic program. We will show that  $P$  has a property similar to the least model property of logic programs. Of course, given that different models of  $P$  can have different anonymous domains, there can be no unique “least” model. However, such models are all isomorphic. These notions are made precise in the following definitions and lemmas.

**Definition 10 (Initial Model).** Given an F-logic program  $P$  and an interpretation  $\mathcal{I}$ ,  $\mathcal{I}$  is an initial model of  $P$ , iff

- $\mathcal{I}$  is a model of  $P$ , i.e.,  $\mathcal{I} \models P$ ; and
- $\mathcal{I}$  is minimal, i.e., there is no  $\mathcal{J} \models P$  such that  $\mathcal{J} \prec \mathcal{I}$ .

To construct initial models, we will adapt the classical fixpoint theory for Horn programs to the case of F-logic programs with anonymous ID symbols and reified statements. Namely, we will define a program consequence operator whose least fixpoint computes an initial model of a given F-logic program.

Note that the F-logic programs described here can contain reified statements and a reified statement can be a conjunction of atomic formulas. Therefore, such conjunctions of atomic formulas can be deduced. The consequence of inferring a conjunction is that each constituent atomic formula in the conjunction is also deduced and added to the model. Let  $A$  be a conjunction of atomic formulas. We will use  $flatten(A)$  to denote the set of atomic formulas that appear in  $A$ .

**Definition 11 (Program Consequence Operator).** Let  $P$  be an  $F$ -logic program,  $\mathcal{D}$  be an anonymous domain, and  $\Pi_{\mathcal{D}}(P)$  be a skolemization of  $P$  with respect to  $\mathcal{D}$ . Similarly to [25], we can define a program consequence operator,  $\mathcal{T}_{\Pi_{\mathcal{D}}(P)}$ , which maps an interpretation,  $\mathcal{I} = (\mathcal{D}, \mathcal{S})$ , over  $\mathcal{D}$  to another interpretation over  $\mathcal{D}$  as follows:  $\mathcal{T}_{\Pi_{\mathcal{D}}(P)}(\mathcal{I}) = \mathcal{J}$ , where  $\mathcal{J} = (\mathcal{D}, \mathcal{R})$  and  $\mathcal{R}$  is the following set of terms:

$$\left\{ A \mid \begin{array}{l} \text{There is a ground instance } A_1, \dots, A_m \leftarrow B_1, \dots, B_n \\ \text{of a rule in } \Pi_{\mathcal{D}}(P) \text{ such that:} \\ - A \in \text{flatten}(A_i), \text{ for some } 1 \leq i \leq m; \text{ and} \\ - B_j \in \mathcal{S} \text{ for all } B_j, 1 \leq j \leq n. \end{array} \right\}$$

Let  $\mathcal{I} = (\mathcal{D}, \mathcal{S})$  and  $\mathcal{J} = (\mathcal{D}, \mathcal{R})$  be two interpretations. We write  $\mathcal{I} \subseteq \mathcal{J}$  iff  $\mathcal{S} \subseteq \mathcal{R}$ . Thus  $\subseteq$  defines a partial order among all interpretations on the same anonymous domain. It follows from the standard results in logic programming [25] that  $\mathcal{T}_{\Pi_{\mathcal{D}}(P)}$  is monotonic and thus it has a *unique* least fixpoint, denoted  $\text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$ .

**Theorem 1.** Let  $P$  be an  $F$ -logic program,  $\mathcal{D}$  be an anonymous domain, and  $\Pi_{\mathcal{D}}(P)$  be a skolemization of  $P$ . Then  $\mathcal{I} = (\mathcal{D}, \text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)}))$  is an initial model of  $P$ .

*Proof.* Similarly to [25], we can show that  $\text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$  is a model of  $\Pi_{\mathcal{D}}(P)$ . By Definition 9, it then follows that  $\mathcal{I}$  is a model of  $P$ . To show that it is an initial model of  $P$ , we need to show that there is no  $\mathcal{J} \models P$  such that  $\mathcal{J} \prec \mathcal{I}$ .

Let  $\mathcal{J} \models P$ , where  $\mathcal{J} = (\mathcal{D}_1, \mathcal{S}_1)$ . Since  $\mathcal{I} \models P$ , there must exist  $\Pi_{\mathcal{D}}(P)$  such that  $\mathcal{I} \models \Pi_{\mathcal{D}}(P)$ . Similarly, there must exist  $\Pi_{\mathcal{D}_1}(P)$  such that  $\mathcal{J} \models \Pi_{\mathcal{D}_1}(P)$ . Next we will show that  $\mathcal{I} \preceq \mathcal{J}$  and hence finish the proof.

If the program  $P$  does not contain any anonymous ID terms, then the two programs,  $\Pi_{\mathcal{D}}(P)$  and  $\Pi_{\mathcal{D}_1}(P)$ , must be the same. It follows from the classical results [25] that  $\text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)}) \subseteq \mathcal{S}_1$ . Thus  $\mathcal{I} \preceq \mathcal{J}$ .

If  $P$  contains anonymous ID terms, then there is an 1-1 mapping  $\tau: \mathcal{D} \rightarrow \mathcal{D}_1$  such that  $\tau(\Pi_{\mathcal{D}}(P))$ , the program obtained by applying  $\tau$  to every term in the program  $\Pi_{\mathcal{D}}(P)$ , is the same as  $\Pi_{\mathcal{D}_1}(P)$ . Then by transfinite induction on each iteration of applying the program consequence operator to generate  $\text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$ , we can show that  $\tau(\text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})) \subseteq \mathcal{S}_1$ . It follows that  $\mathcal{I} \preceq \mathcal{J}$ .

Of course, a program can have different initial models, since there can be different anonymous domains and different skolemizations. However, all initial models turn out to be isomorphic and all are least fixpoints of the  $\mathcal{T}_{\Pi_{\mathcal{D}}(P)}$  operator. This will be seen from the following two corollaries.

**Corollary 1.**  $\mathcal{I} = (\mathcal{D}, \mathcal{S})$  is an initial model of an  $F$ -logic program  $P$  iff there is a skolemization,  $\Pi_{\mathcal{D}}(P)$ , of  $P$  such that  $\mathcal{S} = \text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$ .

*Proof.* By Theorem 1, if  $\mathcal{S} = \text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$  then  $\mathcal{I} = (\mathcal{D}, \mathcal{S})$  is an initial model of  $P$ . Conversely, if  $\mathcal{I} = (\mathcal{D}, \mathcal{S})$  is an initial model of  $P$ , then there is a skolemization,

$\Pi_{\mathcal{D}}(P)$ , of  $P$  such that  $\mathcal{S} \models \Pi_{\mathcal{D}}(P)$ , by Definition 9. Since  $\text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$  is the minimal model of  $\Pi_{\mathcal{D}}(P)$ , it follows that  $\mathcal{S} \supseteq \text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$ . Therefore, we must have  $\mathcal{S} = \text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}}(P)})$ , for otherwise we get a contradiction with the assumption that  $\mathcal{I}$  is an initial model.

**Corollary 2.** *All initial models of an F-logic program are isomorphic.*

*Proof.* Let  $\mathcal{I} = (\mathcal{D}_1, \mathcal{S}_1)$  and  $\mathcal{J} = (\mathcal{D}_2, \mathcal{S}_2)$  be two initial models of an F-logic program  $P$ . It follows that there are two skolemizations,  $\Pi_{\mathcal{D}_1}(P)$  and  $\Pi_{\mathcal{D}_2}(P)$ , such that  $\mathcal{S}_1 = \text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}_1}(P)})$  and  $\mathcal{S}_2 = \text{lfp}(\mathcal{T}_{\Pi_{\mathcal{D}_2}(P)})$ , by Corollary 1. We can construct an 1-1 mapping  $\tau: \mathcal{D}_1 \rightarrow \mathcal{D}_2$  such that  $\tau(\Pi_{\mathcal{D}_1}(P))$  is the same as  $\Pi_{\mathcal{D}_2}(P)$ . It can be shown that  $\tau(\mathcal{S}_1) = \mathcal{S}_2$  by a transfinite induction on the program consequence operator. Therefore  $\mathcal{I}$  and  $\mathcal{J}$  are isomorphic.

## 5 Two Flavors of Entailment

To answer queries and to reason about logical statements we need to define the notion of *logical entailment* of one scoped formula by another. We show that two different notions of entailment are possible for RDF: *strict entailment*, denoted  $\models$ , and *relaxed entailment*, denoted  $\approx$ . In general these two entailments have rather different properties, but they coincide for certain classes of programs. Only one of these notions, *i.e.*, relaxed entailment, is currently reflected in the proposal for the RDF model theory [15].

The first type of entailment is defined similarly to classical logic.

**Definition 12 (Strict Entailment).** *Let  $\phi$  and  $\psi$  be two scoped formulas. We write  $\phi \models \psi$  iff for every interpretation  $\mathcal{I}$ , whenever  $\mathcal{I} \models \phi$  then it is the case that  $\mathcal{I} \models \psi$ .*

This definition has the normally expected properties, *e.g.*, if  $\phi \models \psi$  then  $\phi \models \psi \vee \xi$ , and if  $\phi \models \psi \wedge \xi$  then  $\phi \models \psi$  and  $\phi \models \xi$ . As a special case, if  $\phi$  represents an RDF graph, *i.e.*,  $\phi$  is just a conjunction of scoped atomic F-logic formulas, and  $\psi$  represents a subgraph of  $\phi$ , then  $\phi \models \psi$ . This property is analogous to the one exhibited by the current proposal for the RDF model theory [15]. However, the difference is that strict entailment does not hold between a *proper instance*<sup>5</sup> of an RDF graph and the graph itself. For instance,  $\{ \text{john}[\text{likes} \rightarrow \text{food}] \}$  is a proper instance of both  $\{ \text{\_}\#[\text{likes} \rightarrow \text{food}] \} \wedge \{ \text{\_}\#[\text{likes} \rightarrow \text{food}] \}$  and  $\{ \text{john}[\text{likes} \rightarrow \text{\_}\#] \} \wedge \{ \text{john}[\text{likes} \rightarrow \text{\_}\#] \}$ , but

$$\begin{aligned} \{ \text{john}[\text{likes} \rightarrow \text{food}] \} &\not\models \{ \text{\_}\#[\text{likes} \rightarrow \text{food}] \} \wedge \{ \text{\_}\#[\text{likes} \rightarrow \text{food}] \} \\ \{ \text{john}[\text{likes} \rightarrow \text{food}] \} &\not\models \{ \text{john}[\text{likes} \rightarrow \text{\_}\#] \} \wedge \{ \text{john}[\text{likes} \rightarrow \text{\_}\#] \} \end{aligned}$$

<sup>5</sup> A proper instance of an RDF graph is obtained by replacing one or more anonymous resources with named resources [15] (which in our case means replacing them with the constants in  $\mathcal{C}$ ).

Indeed, when applied to RDF graphs, strict entailment corresponds to isomorphic embedding of graphs (the entailed graph is the one that is embedded). In isomorphic embedding, named resource nodes in one graph are mapped to identically named nodes in another graph, while blank nodes are mapped to blanked nodes. Moreover, this mapping is 1-1.

In contrast, the notion of entailment defined in the proposed RDF model theory [15] corresponds to a more relaxed notion of embedding — one where blank nodes can be mapped to anything. In this notion, two blank nodes can even be spliced into the same node.

It turns out that this second notion of entailment, *i.e.*, relaxed entailment, can also be captured in our framework. First, we need the following transformation, which replaces anonymous ID symbols in a scoped formula with existential quantifiers. Let  $\psi$  be a scoped formula:

1. If  $\psi$  has an occurrence of  $\_ \#$ , then replace  $\psi$  with  $\exists X(\psi')$ , where  $X$  is a new variables that does not occur anywhere else and  $\psi'$  is obtained from  $\psi$  by replacing this particular occurrence of  $\_ \#$  with  $X$ .
2. If  $\psi$  has an occurrence of a numbered anonymous ID symbol, say  $\_ \#1$ , then rewrite  $\psi$  into  $\exists X(\psi')$ , where  $X$  is a new variable and  $\psi'$  denotes  $\psi$  where all occurrences of  $\_ \#1$  within the same scope are replaced with  $X$ .

Let  $\text{strip}(\phi)$  denote the first order formula obtained by an exhaustive application of the above rewrite rules to a given scoped formula  $\phi$ . Note that the new existential quantifiers generated by the rewrite rules are placed at the beginning of the newly created formula. For instance, let  $\phi = \forall Y(o[m(Y) \rightarrow \_ \#] \leftarrow o[f(Y) \rightarrow v])$ , then  $\text{strip}(\phi) = \exists X \forall Y(o[m(Y) \rightarrow X] \leftarrow o[f(Y) \rightarrow v])$ .

**Definition 13 (Relaxed Entailment).** *Let  $P$  and  $Q$  be two  $F$ -logic programs. We write  $P \approx Q$  iff  $P \models \text{strip}(Q)$ .*

We will now investigate the relationship between strict and relaxed entailment.

**Theorem 2.** *Let  $P$  and  $Q$  be two  $F$ -logic programs. If  $P \models Q$  then  $P \approx Q$ .*

*Proof.* Suppose  $P \models Q$ . Let  $\mathcal{I} \models P$ . Then it must be the case that  $\mathcal{I} \models Q$ . This means that there is a skolemization of  $Q$  such that  $\mathcal{I} \models \Pi_{\mathcal{D}}(Q)$ , where the last  $\models$  denotes entailment in the classical sense (note that  $\Pi_{\mathcal{D}}(Q)$  has no anonymous ID symbols). Let  $\nu_Q$  denote the mapping defined by the skolemization: it assigns a constant to each occurrence of an anonymous ID symbol in accordance with the scoping rules. Let  $\eta_Q$  denote the 1-1 and onto mapping from the occurrences of the anonymous ID symbols in  $Q$  to the occurrences of the newly generated variables in  $\text{strip}(Q)$ , which is defined by the  $\text{strip}$  transformation above. Then it is easy to see that  $\nu_Q \circ \eta_Q^{-1}$  is a variable assignment for the existential variables in  $\text{strip}(Q)$ , which converts  $\text{strip}(Q)$  into  $\Pi_{\mathcal{D}}(Q)$  (after removal of the now defunct existential quantifiers). Therefore,  $\mathcal{I}$  is a model of  $\text{strip}(Q)$ .

For another connection between these two entailments, it is easy to see that if the consequent is an F-logic program that does not use anonymous ID symbols then the two notions of entailment coincide.

**Theorem 3.** *Let  $P$  and  $Q$  be two F-logic programs and suppose  $Q$  does not contain anonymous ID symbols. Then  $P \models Q$  iff  $P \approx Q$ .*

*Proof.* The result follows immediately, since in this case  $\text{strip}(Q) = Q$ .

Recall that under strict entailment, an instance of an RDF graph does not necessarily entail the graph. This property holds, however, for relaxed entailment, which makes it behave similarly to the RDF model theory [15].

**Theorem 4.** *Given two F-logic programs  $P$  and  $Q$ , if  $Q$  represents an RDF graph, i.e., a conjunction of scoped atomic F-logic formulas, and  $P$  is a proper instance of  $Q$ , then  $P \approx Q$ .*

*Proof.* Recall that a proper instance of an RDF graph is obtained by replacing one or more anonymous resources with arbitrarily named resources [15], which are constants of  $\mathcal{C}$  in our logic.

Since  $P \models P$ , it follows from Theorem 2 that  $P \approx P$ , i.e.,  $P \models \text{strip}(P)$ . But  $\text{strip}(P)$  is nothing but  $\text{strip}(Q)$  with some constants substituted for existential quantifiers. By classical results about entailment, we conclude that  $\text{strip}(P) \models \text{strip}(Q)$ . Combined with  $P \models \text{strip}(P)$ , we get  $P \models \text{strip}(Q)$ , i.e.,  $P \approx Q$ .

Let us revisit the previous example where  $\{ \text{john}[\text{likes} \rightarrow \text{food}] \}$  is a proper instance of both  $\{ \_ \# [\text{likes} \rightarrow \text{food}] \} \wedge \{ \_ \# [\text{likes} \rightarrow \text{food}] \}$  and  $\{ \text{john}[\text{likes} \rightarrow \_ \#] \} \wedge \{ \text{john}[\text{likes} \rightarrow \_ \#] \}$ . We have

$$\begin{aligned} \{ \text{john}[\text{likes} \rightarrow \text{food}] \} &\approx \{ \_ \# [\text{likes} \rightarrow \text{food}] \} \wedge \{ \_ \# [\text{likes} \rightarrow \text{food}] \} \\ \{ \text{john}[\text{likes} \rightarrow \text{food}] \} &\approx \{ \text{john}[\text{likes} \rightarrow \_ \#] \} \wedge \{ \text{john}[\text{likes} \rightarrow \_ \#] \} \end{aligned}$$

We should note that it is not clear which notion of entailment,  $\models$  or  $\approx$ , is more appropriate for RDF graph entailment. Perhaps, both should be used, as they give rise to different notions of equivalence for these graphs. We say that  $P \equiv Q$  iff  $P \models Q$  and  $Q \models P$ . Similarly,  $P \cong Q$  iff  $P \approx Q$  and  $Q \approx P$ . It is easy to show that if  $P \equiv Q$  then the initial models of  $P$  and  $Q$  are isomorphic and in a well-defined sense they correspond to the RDF graph represented by these programs. In contrast, if  $P \cong Q$ , then  $P$  and  $Q$  can still have non-isomorphic initial models and their RDF graphs can be different. For instance, if  $P$  has only one fact,  $\{ a[b \rightarrow c] \}$ , and  $Q$  is  $\{ a[b \rightarrow c], a[b \rightarrow \_ \#] \}$ , then  $P \cong Q$ . However, the RDF graph of  $P$  has only one arc and two nodes, while the graph of  $Q$  has three nodes and two arcs. But each graph can be (non-isomorphically) embedded into the other. The relationship between relaxed entailment and non-isomorphic graph embedding is explored in the following theorem.

**Theorem 5.** *Let  $P$  and  $Q$  be F-logic programs representing RDF graphs. Let  $\mathcal{I} = (\mathcal{D}_1, \mathcal{R})$  and  $\mathcal{J} = (\mathcal{D}_2, \mathcal{S})$  be the initial models of  $P$  and  $Q$ , respectively. Then  $P \approx Q$  iff there is an augmented domain mapping  $\lambda: \mathcal{D}_2 \rightarrow \mathcal{D}_1 \cup \mathcal{C}$  such that  $\lambda(\mathcal{S}) \subseteq \mathcal{R}$ .*



*Proof.* Suppose  $P \approx Q$ . Then  $\mathcal{I} \models \text{strip}(Q)$  by Definition 13. This means that there is an assignment,  $\rho$ , of the existential variables in  $\text{strip}(Q)$  to the elements in  $\mathcal{C} \cup \mathcal{D}_1$  such that  $\mathcal{R} \models Q'$ , where  $Q'$  is  $\rho(\text{strip}(Q))$  ( $\text{strip}(Q)$  with the now defunct existential quantifiers removed).

Let  $\eta_Q$  denote the 1-1 and onto mapping from the occurrences of the anonymous ID symbols in  $Q$  to the occurrences of the newly generated variables in  $\text{strip}(Q)$ , which is defined by the  $\text{strip}$  transformation. Then  $\rho(\eta_Q(Q)) = Q'$ .

Since  $\mathcal{J} = (\mathcal{D}_2, \mathcal{S})$  is an initial model of  $Q$ , let  $\nu_Q$  denote the mapping defined by the skolemization: it assigns a constant to each occurrence of an anonymous ID symbol in accordance with the scoping rules. Because  $Q$  represents an RDF graph, *i.e.*, a set of facts, it follows that  $\rho(\eta_Q(\nu_Q^{-1}(\mathcal{S}))) = Q'$ . Finally, since  $\mathcal{R} \models Q'$  and both  $\mathcal{R}$  and  $Q'$  are sets of facts, it follows that  $Q' \subseteq \mathcal{R}$ . Therefore,  $\rho \circ \eta_Q \circ \nu_Q^{-1}$  is the required augmented domain mapping.

By a similar argument we can also prove the opposite direction.

To conclude our discussion of entailment, we would like to remark on the relationship between equality and the semantics presented in Sections 4 and 5. It is a rather common misconception that a Herbrand style semantics, such as the one presented here, implies a closed world assumption or a unique name assumption. In actuality, resolution-based proof theory for full classical predicate calculus relies on Herbrand models. This semantics can be extended to include any kind of equality theory by superimposing an equivalence relation over the Herbrand universe [8]. Likewise, our semantics for anonymous identity implies neither the unique name nor the closed world assumption. Equality theories can be incorporated in the same way as in [8] and the semantics can be made nonmonotonic if required. These extensions are implemented in the *FLORA-2* system [33].

## 6 Compositionality of the Semantics

It is our contention that the semantics for Semantic Web languages should have the *semantic compositionality* property. Informally, this means that the result of putting together, say, two RDF documents should be another valid RDF document. In other words, if  $P_1$  and  $P_2$  are two RDF documents then each model of  $P_1 \cup P_2$  should be some sort of a union of the models of  $P_1$  and  $P_2$  separately.

It turns out that the F-logic language with anonymous ID symbols presented in the previous section satisfies this requirement, whereas the N3 notation for RDF, similar to RDF triple syntax (see [3] for an introduction to N3), and the current proposal for the RDF model theory [15] do not. To make this requirement precise, we first need to define what we mean by the “union” of two interpretations.

First, we recall the notion of a disjoint union of sets. Let  $\mathcal{D}_1$  and  $\mathcal{D}_2$  be two sets. Their *disjoint union*, denoted  $\mathcal{D}_1 \uplus \mathcal{D}_2$ , is obtained by “renaming apart” the common elements of  $\mathcal{D}_1$  and  $\mathcal{D}_2$  (thus making the sets disjoint) and then taking the union. The set  $\mathcal{D}_1 \uplus \mathcal{D}_2$  has the following properties:

- There are 1-1 mappings  $\iota_1 : \mathcal{D}_1 \rightarrow \mathcal{D}_1 \uplus \mathcal{D}_2$  and  $\iota_2 : \mathcal{D}_2 \rightarrow \mathcal{D}_1 \uplus \mathcal{D}_2$ ;
- $\mathcal{D}_1 \uplus \mathcal{D}_2 = \iota_1(\mathcal{D}_1) \cup \iota_2(\mathcal{D}_2)$ ; and
- $\iota_1(\mathcal{D}_1) \cap \iota_2(\mathcal{D}_2) = \emptyset$ .

**Definition 14 (Disjoint Union of Interpretations).** Let  $\mathcal{I}_1 = (\mathcal{D}_1, \mathcal{S}_1)$  and  $\mathcal{I}_2 = (\mathcal{D}_2, \mathcal{S}_2)$  be two interpretations. A disjoint union of  $\mathcal{I}_1$  and  $\mathcal{I}_2$ , denoted  $\mathcal{I}_1 \uplus \mathcal{I}_2$ , is an interpretation of the form  $(\mathcal{D}_1 \uplus \mathcal{D}_2, \iota_1(\mathcal{S}_1) \cup \iota_2(\mathcal{S}_2))$ , where  $\iota_1$  and  $\iota_2$  are the 1-1 embeddings  $\iota_1 : \mathcal{D}_1 \rightarrow \mathcal{D}_1 \uplus \mathcal{D}_2$  and  $\iota_2 : \mathcal{D}_2 \rightarrow \mathcal{D}_1 \uplus \mathcal{D}_2$  that are associated with  $\mathcal{D}_1 \uplus \mathcal{D}_2$ . In other words, a disjoint union of two interpretations makes sure that anonymous constants that happen to occur in both anonymous domains  $\mathcal{D}_1$  and  $\mathcal{D}_2$  are renamed apart (both in these domains and in  $\mathcal{S}_1$  and  $\mathcal{S}_2$ ) prior to taking the union.

We can now state the following simple result:

**Theorem 6.** Let  $P_1$  and  $P_2$  be two F-logic programs, and let  $\mathcal{I}_1$  and  $\mathcal{I}_2$  be the initial models of  $P_1$  and  $P_2$ , respectively. Suppose that both  $P_1$  and  $P_2$  represent an RDF graph (i.e., a conjunction of scoped atomic formulas). Then  $\mathcal{I}_1 \uplus \mathcal{I}_2$  is an initial model of  $P_1 \cup P_2$ .

*Proof.* Since  $P_1$  and  $P_2$  represent RDF graphs, they may contain anonymous ID symbols but no variables. Let  $\mathcal{I}_1 = (\mathcal{D}_1, \mathcal{S}_1)$  and  $\mathcal{I}_2 = (\mathcal{D}_2, \mathcal{S}_2)$ . To simplify the exposition, let us assume that  $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$ .

There must be skolemizations  $\Pi_{\mathcal{D}_1}(P_1)$  and  $\Pi_{\mathcal{D}_2}(P_2)$ , such that  $\mathcal{S}_1$  and  $\mathcal{S}_2$  are, respectively, the minimal models of  $\Pi_{\mathcal{D}_1}(P_1)$  and  $\Pi_{\mathcal{D}_2}(P_2)$ . Since  $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$ , it follows that  $\Pi_{\mathcal{D}_1}(P_1) \cup \Pi_{\mathcal{D}_2}(P_2)$  is a valid skolemization of  $P_1 \cup P_2$ . Since  $P_1$  and  $P_2$  represent RDF graphs, the skolemizations  $\Pi_{\mathcal{D}_1}(P_1)$  and  $\Pi_{\mathcal{D}_2}(P_2)$  have no variables and thus  $\mathcal{S}_1$  and  $\mathcal{S}_2$  are identical to  $\Pi_{\mathcal{D}_1}(P_1)$  and  $\Pi_{\mathcal{D}_2}(P_2)$ , respectively. It is therefore easy to see that  $\mathcal{S}_1 \cup \mathcal{S}_2$  is the minimal model of  $\Pi_{\mathcal{D}_1}(P_1) \cup \Pi_{\mathcal{D}_2}(P_2)$ . Thus,  $\mathcal{I}_1 \uplus \mathcal{I}_2$  is an initial model of  $P_1 \cup P_2$ .

The above result could be extended to programs that contain inference rules. The main difference is that  $P_1 \cup P_2$  can derive more facts than the union of the facts that  $P_1$  and  $P_2$  can derive in separation. Consequently,  $\mathcal{I}_1 \uplus \mathcal{I}_2$  may not be an initial model of  $P_1 \cup P_2$ , but rather a congruent sub-interpretation of an initial model.

It appears that the current proposal for the RDF model theory does not have the compositionality property. To see this, consider the sentence *someone loves Mary* represented using RDF triple syntax:

$\{[\text{loves}], [X], [\text{Mary}]\}$

Let  $P_1$  denote this document. Let  $P_2$  denote the document that represents the sentence *someone invented the bulb*:

$\{[\text{inventor}], [X], [\text{Bulb}]\}$

By composing the two documents, we get  $P_1 \cup P_2$ :

$$\begin{aligned} &\{[\text{loves}], [X], [\text{Mary}]\} \\ &\{[\text{inventor}], [X], [\text{Bulb}]\} \end{aligned}$$

Although the interpretation  $\{ \{[\text{loves}], [\text{John}], [\text{Mary}]\} \}$  is a model for the document  $P_1$  and  $\{ \{[\text{inventor}], [\text{ThomasEdison}], [\text{Bulb}]\} \}$  is a model for  $P_2$ , their (disjoint) union,  $\{ \{[\text{loves}], [\text{John}], [\text{Mary}]\}, \{[\text{inventor}], [\text{ThomasEdison}], [\text{Bulb}]\} \}$  is *not* a model of  $P_1 \cup P_2$ .

This problem can be rectified by adding the notion of scope, as introduced in this paper, to N3 and the RDF model theory. An alternative solution — changing the definition of the union of RDF graphs — would be undesirable, as it will only serve to highlight the inadequacy of the current syntax and semantics of N3.

## 7 Properties of Reification

In this section we discuss some properties of reification in F-logic and compare them to the current proposal for RDF model theory [15].

One important difference is that in F-logic reified statements are themselves objects, whereas in the RDF model theory they are referred to by names. In particular, one can give two names to the same statement and these would be completely unrelated objects. In our opinion, such a semantics is too weak.<sup>6</sup> To illustrate the problem, consider again the statement from Section 3 that *John believes that Thomas Edison (this time a known resource) invented the bulb*:

$$\begin{aligned} &\{[\text{type}], [X], [\text{RDF:Statement}]\} \\ &\{[\text{predicate}], [X], [\text{inventor}]\} \\ &\{[\text{subject}], [X], [\text{http://foo.org/TheBulb}]\} \\ &\{[\text{object}], [X], [\text{http://foo.org/ThomasEdison}]\} \\ &\{\text{believes}, [\text{http://xyz.com/John}], [X]\} \end{aligned}$$

In the RDF triple syntax, one can add another reference to the reified statement:

$$\begin{aligned} &\{[\text{type}], [Y], [\text{RDF:Statement}]\} \\ &\{[\text{predicate}], [Y], [\text{inventor}]\} \\ &\{[\text{subject}], [Y], [\text{http://foo.org/TheBulb}]\} \\ &\{[\text{object}], [Y], [\text{http://foo.org/ThomasEdison}]\} \end{aligned}$$

However, in RDF, the objects referred to by  $X$  and  $Y$  would have nothing to do with each other. For instance, John would believe  $X$  but not  $Y$ . Likewise, stating that  $X$  is a true statement or that it was made by *Encyclopedia Britannica* does not imply that the same holds for  $Y$ . In contrast, in F-logic we can say

<sup>6</sup> There has been lengthy discussion of this problem on the RDF mailing list <http://lists.w3.org/Archives/Public/www-rdf-interest/> and it has been proposed that reifications of RDF statements should be viewed as *statings* — which are “instances” of statements. (See <http://ilrt.org/discovery/2000/11/statements/>.) However, we remain unconvinced as to the desirability of this distinction.

```
'http://foo.org/TheBulb'[inventor → 'http://foo.org/ThomasEdison']
[
  veracity → true,
  authority → 'http://www.britannica.com/'
].
```

So anyone who believes this statement would be believing a statement whose veracity attribute has the value true and which is believed to be authorized by Britannica. In fact, we can even go further and specify the rule

Statement  $\leftarrow$  Statement[veracity  $\rightarrow$  true].

which will make any statement whose veracity property is “true” into a true statement in every model. Thus, for example, Mary, who also believes this statement and who might even be defined in a different document, would be believing a true assertion.

Note that in our semantics conjunctions of atomic formulas are terms and thus can be treated as objects. In fact, this idea can be further extended to allow reification of more general statements. We will show some simple uses of reified conjunctions. First, for some properties, such as beliefs, it is reasonable to assume that conjunctions can be broken apart, because if someone believes in a combined statement then she is likely to believe in all of its parts. This can be expressed by the following rule:

$X[\text{believes} \rightarrow S1], X[\text{believes} \rightarrow S2] \leftarrow X[\text{believes} \rightarrow (S1 \wedge S2)]$ .

On the other hand, in some contexts reified conjuncted statements cannot be broken up. For instance, the following might be considered as a true statement

$(\text{john}[\text{likes} \rightarrow \text{sally}] \wedge \text{sally}[\text{likes} \rightarrow \text{john}]) [\text{statementAbout} \rightarrow \text{friendship}]$ .

But

$(\text{john}[\text{likes} \rightarrow \text{sally}]) [\text{statementAbout} \rightarrow \text{friendship}]$

is not necessarily a true statement.

An important advantage of our semantics is that it is developed in a general framework, which provides a basis for reasoning about reified statements — not only for stating them as facts. We have already shown how one can state that certain beliefs are actually true. However, there are many more possibilities for non-trivial reasoning. For instance, *Bob always believes when Alice says that some statement is made by a trusted third party*. This knowledge can be encoded using the following rule:

$\text{'http://xyz.com/Bob'}$ [believes  $\rightarrow$  Statement[authority  $\rightarrow$  A]]  $\leftarrow$   
 $\text{'http://xyz.com/Alice'}$ [says  $\rightarrow$  Statement[authority  $\rightarrow$  A]],  
 $\text{'http://xyz.com/Bob'}$ [trusts  $\rightarrow$  A].

In fact, one can express a number of belief systems using rules and then combine them with reification to derive useful conclusions about reified beliefs.

## 8 Reification and Logical Paradoxes

Year 2001 marked a Centennial of the famous Russell’s paradox — a remarkable discovery of the inconsistency of Frege’s comprehension axioms. The basic reason for the existence of this paradox in Frege’s logic is the ability to reify logical sentences and make statements about these sentences. Ever since this discovery logics that permit reification were subject to strict and just scrutiny. In this section we will briefly explain the nature of Russell’s paradox as it pertains to reification and how it affects our theory. A nice discussion of this paradox and of the ways to avoid it appears in [27].

The simplest and most familiar form of Russell’s paradox is the so called Liar’s paradox — a statement,  $L$ , which says that  $L$  is false. If  $L$  is true then it must be false, and if it is false then it must be true. In order to be able to make such statements within a logic we need the ability to refer to logical sentences in the language of the logic. We will show how such a paradox can be constructed within F-logic extended with the reification mechanism. In fact, we shall see that paradoxes can be constructed using HiLog alone.

To obtain a paradox, we need the so called *comprehension axiom schema* of the following form

$$\exists P \forall X (P(X) \leftrightarrow \phi(X))$$

in addition to reification. Here  $\phi$  is some formula that does not have  $P$  as a free variable. This axiom says that we can always define a predicate that is true precisely of those things that have the property  $\phi$ . For instance,  $\phi(X)$  can be something like  $q(X) \wedge X$ .

Note that the comprehension axioms are valid HiLog formulas. If we now substitute  $\neg X(X)$  for  $\phi$  and choose  $X$  to be  $P$ , we get

$$\exists P (P(P) \leftrightarrow \neg P(P))$$

which is unsatisfiable. The paradox consists in the realization that a very natural set of axioms, such as comprehension axioms, is unsatisfiable.

We might still think that Russell’s paradox is no more than a curiosity, because the comprehension axioms are so general that they are, perhaps, of little use for knowledge representation on the Semantic Web. Unfortunately we are not out of the woods yet. Let **true** be a new predicate symbol and consider the following *truth axiom*:

$$\forall X (\text{true}(X) \leftrightarrow X) \tag{1}$$

This axiom states that everything that is asserted to be true in the meta-level (*i.e.*, using the truth-predicate applied to reified formulas) is, indeed, true in the logic. It turns out that this axiom is an instance of the comprehension axiom schema and it is almost as bad. Unlike the comprehension axioms, we cannot simply dismiss the truth axiom as too general because it is so close to routine things that people do in knowledge representation. For instance, one might want

to say that Alice does not know false facts ( $X : - \text{sallyKnows}(X)$ ) and that Bob knows every true fact believed by Sally ( $\text{bobKnows}(X) : - X, \text{sallyBelieves}(X)$ ).<sup>7</sup>

To see how the truth axiom leads to a paradox, we assume that we are allowed to define new concepts using previously defined ones. Specifically, let  $r$  be a completely new symbol defined as follows:

$$\forall X (r(X) \leftrightarrow \neg \text{true}(X(X)))$$

Such definitions are the staple of every logic-based knowledge representation language, although in some (like Prolog) the definition would be unidirectional.

By substituting  $r$  for  $X$ , we obtain the so called *heterological Liar's statement*:

$$r(r) \leftrightarrow \neg \text{true}(r(r)) \quad (2)$$

It is easy to see that the Liar's statement is inconsistent with the truth axiom, since we can derive a contradiction:

$$r(r) \leftrightarrow \neg \text{true}(r(r)) \leftrightarrow \neg r(r)$$

Since the truth axiom is simply a pair of Horn clauses and the Liar's statement is also not too far off from what an arbitrary logic program could have, we need to re-examine our reified F-logic to see if it is of any use at all. Is it possible that an innocuous rule set written by an unsuspecting provider of Semantic Web content can have a hidden inconsistency?

**Theorem 7.** *Horn programs in reified F-logic augmented with the truth axiom (1) are consistent.*

*Proof.* Let  $P$  be an F-logic Horn program. Since the truth axiom is just a pair of Horn clauses, we can consider them to be part of  $P$ . By Theorem 1,  $P$  has an initial model and therefore is consistent.

The above theorem can be extended to programs that have negative literals in the rule body under the well-founded semantics [18]. Therefore, we can be satisfied that for most practical purposes reified F-logic is an adequate knowledge representation formalism.

An interesting question is whether it is possible to extend the scope of reification in F-logic to include disjunction and negation. The answer to the former is yes, because disjunctive logic programming [26] can be adapted to F-logic. On the other hand, reifying negation would create problems unless the structure of the programs and the truth axiom are restricted in some way. Perlis [27] proposed an elegant way to restrict the truth axiom, which eliminates paradoxes from first-order logics.<sup>8</sup> To see how inconsistency can arise if negation is reified, consider the following program:

<sup>7</sup> Another example is the axiom  $\text{Statement} \leftarrow \text{Statement}[\text{veracity} \rightarrow \text{true}]$  of the previous section.

<sup>8</sup> Unfortunately, this solution does not quite apply to HiLog, because Perlis' syntax precludes the heterological Liar's statement.

$$\begin{aligned}\text{true}(\neg p(X)) &\leftarrow p(X). \\ p(X) &\leftarrow \neg p(X).\end{aligned}$$

Because of the truth axiom, we obtain  $p(X) \leftrightarrow \neg p(X)$ .<sup>9</sup>

To summarize our discussion, we have shown that reified F-logic avoids paradoxes through the following restrictions:

- Programs must be sets of extended Horn clauses (*i.e.*, no negation in the rule head); and
- Reification of negative facts is not permitted, which stops negation from appearing in the rule heads through the back door.

This is not the only useful set of restrictions, however. In the  $\mathcal{F}\text{LORA-2}$  system [33], reification of negative facts is allowed and thus the second condition above does not hold. Instead,  $\mathcal{F}\text{LORA-2}$  precludes negative literals in rule heads (like the first restriction above) and closes the remaining loophole with the following restriction:

*The head of a rule cannot be a variable term.*

Note that rules like  $X(Y) :- \text{body}$  or  $X[Y \rightarrow Z] :- \text{body}$  are still allowed. The excluded rules are of the form  $X :- \text{body}$ . This restriction effectively rules out the truth axiom and, since negative literals cannot occur in the rule head, it becomes impossible to derive such facts despite the fact that individual variables can be bound to negative reified facts.

## 9 Conclusion

In this paper we presented an extension of F-logic that supports anonymous resources and reification. Such an extension makes F-logic a suitable foundation for Semantic Web languages, such as RDF. The language has a simple and natural, paradox-free semantics and a proof theory, and has been implemented in the  $\mathcal{F}\text{LORA-2}$  system [33]. In particular, our semantics rectifies a number of drawbacks of the current proposal for the RDF model theory [15]: non-compositional semantics and weaker than necessary treatment of reification. We also pointed out that there are at least two different (and useful) meanings for the notion of RDF graph entailment. On top of this, our semantics is much more general than [15], as it is given in the framework of an expressive rule-based and frame-based language, which opens up many possibilities for encoding knowledge.

Anonymous objects and reification have uses outside of the Semantic Web context. In fact, they were originally introduced in  $\mathcal{F}\text{LORA-2}$  to satisfy database-specific needs. For instance, anonymous objects are related to the so-called *pure*

<sup>9</sup> It should be kept in mind that this contradiction arises only in classical logic, while logic programs (especially those that involve negation) are based on non-classical logic. However, this example shows that developing an appropriate semantics for reified negation has issues to overcome.

*values* in object-oriented databases [1,23] and they are also very convenient for object-oriented modeling of XML documents. In addition, they turned out to be valuable from the software engineering point of view. Reification was introduced in *FLORA-2* to provide a clean semantics (and an implementation) for aggregate operators, which take a query (*i.e.*, a reified formula) as an argument and perform summarization operations over the set of answers to the query.

**Acknowledgement.** The authors would like to thank the anonymous referees for their comments and suggestions that help improve this paper.

This work was supported in part by NSF grant IIS-0072927. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing those of the funding agency or the U.S. government.

## References

1. F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an Object-Oriented Database System: The Story of O2*. Morgan Kaufmann, San Francisco, CA, 1990.
2. T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (URI): Generic syntax, August 1998. <http://www.isi.edu/in-notes/rfc2396.txt>.
3. Tim Berners-Lee. Primer: Getting into RDF & Semantic Web using N3. <http://www.w3.org/2000/10/swap/Primer>.
4. A.J. Bonner and M. Kifer. An overview of transaction logic. *Theoretical Computer Science*, 133:205–265, October 1994.
5. A.J. Bonner and M. Kifer. A logic for programming database transactions. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 5, pages 117–166. Kluwer Academic Publishers, March 1998.
6. T. Bray, D. Hollander, and A. Layman. Namespaces in XML, January 1999. <http://www.w3.org/TR/REC-xml-names/>.
7. J. Broekstra, C. Fluit, and F. van Harmelen. The state of the art on representation and query languages for semistructured data. Technical report, Administrator, Nederland BV, August 2000.
8. C. L. Chang and R. C. T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.
9. W. Chen, M. Kifer, and D.S. Warren. HiLog: A first-order semantics for higher-order logic programming constructs. In *North American Conference on Logic Programming*, Cambridge, MA, October 1989. MIT Press.
10. W. Chen, M. Kifer, and D.S. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, February 1993.
11. W. Conen, R. Klapsing, and E. Köppen. RDF M&S revisited: From reification to nesting, from containers to lists, from dialect to pure XML. In *Semantic Web Working Symposium (SWWS)*, 2001.
12. H. Davulcu, G. Yang, M. Kifer, and I.V. Ramakrishnan. Design and implementation of the physical layer in webbases: The XRouter experience. In *First International Conference on Computational Logic, DOOD'2000 Stream*, July 2000.
13. S. Decker, D. Brickley, J. Saarela, and J. Angele. A query and inference service for RDF. In *QL'98 - The Query Languages Workshop*, December 1998.



14. S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al., editor, *Database Semantics, Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer Academic Publisher, Boston, 1999.
15. P. Hayes (editor). RDF Model Theory. Technical report, W3C, April 2002. <http://www.w3.org/TR/rdf-mt/>.
16. D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: Or how to enable intelligent access to the WWW. In *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 1998.
17. R. Fikes and D.L. McGuinness. An axiomatic semantics for RDF, RDF Schema, and DAML+OIL. Technical Report KSL-01-01, Knowledge Systems Laboratory, Stanford University, October 2001.
18. A. Van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620–650, July 1991.
19. M.R. Genesereth. Knowledge interchange format. Technical Report NCITS.T2/98-004, Knowledge Systems Laboratory, Stanford University, 1998. Draft proposed American National Standard, <http://logic.stanford.edu/kif/dpans.html>.
20. C. H. Goh, S. Bressan, S. E. Madnick, and M. D. Siegel. Context interchange: Representing and reasoning about data semantics in heterogeneous systems. Technical report, MIT, School of Management, 1996.
21. A. Gupta, B. Ludäscher, and M. E. Martone. Knowledge-based integration of neuroscience data sources. In *12th International Conference on Scientific and Statistical Database Management (SSDBM)*, Berlin, Germany, July 2000. IEEE.
22. G.-J. Houben. HERA: Automatically generating hypermedia front-ends for ad hoc data from heterogeneous and legacy information systems. In *Engineering Federated Information Systems*, pages 81–88. Aka and IOS Press, 2000.
23. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of ACM*, 42:741–843, July 1995.
24. O. Lasilla and R. R. Swick (editors). Resource description framework (RDF) model and syntax specification. Technical report, W3C, February 1999. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>.
25. J. W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1984.
26. J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, Cambridge, Massachusetts, 1992.
27. D. Perlis. Languages with self-reference i: Foundations. *Artificial Intelligence*, 25:301–322, 1985.
28. M. Sintek and S. Decker. TRIPLE – An RDF query, inference, and transformation language. In *Deductive Databases and Knowledge Management (DDL’2001)*, October 2001.
29. S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. AI for the Web — Ontology-based community web portals. In *9-th International World Wide Web Conference (WWW9)*, Amsterdam, The Netherlands, May 2000.
30. G. Yang and M. Kifer. Implementing an efficient DOOD system using a tabling logic engine. In *First International Conference on Computational Logic, DOOD’2000 Stream*, July 2000.
31. G. Yang and M. Kifer. Well-founded optimism: Inheritance in frame-based knowledge bases. In *International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE)*, 2002.

32. G. Yang and M. Kifer. Inheritance and rules in object-oriented semantic web languages. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML)*, 2003.
33. G. Yang, M. Kifer, and C. Zhao. FLORA-2: User's manual. <http://flora.sourceforge.net/>, June 2003.

# IF-Map: An Ontology-Mapping Method Based on Information-Flow Theory

Yannis Kalfoglou<sup>1</sup> and Marco Schorlemmer<sup>2,3</sup> \*

ADVANCED KNOWLEDGE TECHNOLOGIES

<sup>1</sup> Department of Electronics and Computer Science  
University of Southampton

<sup>2</sup> Centre for Intelligent Systems and their Applications  
School of Informatics  
The University of Edinburgh

<sup>3</sup> Escola Universitària de Tecnologies d'Informació i Comunicació  
Universitat Internacional de Catalunya

**Abstract.** In order to tackle the need of sharing knowledge within and across organisational boundaries, the last decade has seen researchers both in academia and industry advocating for the use of ontologies as a means for providing a shared understanding of common domains. But with the generalised use of large distributed environments such as the World Wide Web came the proliferation of many different ontologies, even for the same or similar domain, hence setting forth a new need of sharing—that of sharing ontologies. In addition, if visions such as the Semantic Web are ever going to become a reality, it will be necessary to provide as much automated support as possible to the task of mapping different ontologies. Although many efforts in ontology mapping have already been carried out, we have noticed that few of them are based on strong theoretical grounds and on principled methodologies. Furthermore, many of them are based only on syntactical criteria. In this paper we present a theory and method for automated ontology mapping based on channel theory, a mathematical theory of semantic information flow. We successfully applied our method to a large-scale scenario involving the mapping of several different ontologies of computer-science departments from various UK universities.

## 1 Introduction

The wide-spread use of ontologies by diverse communities and in a variety of applications is a commonality in today's knowledge-sharing efforts. They are the backbone for semantically-rich information sharing, a prerequisite for knowledge sharing. As systems become more distributed and disparate within and across organisational boundaries, there is a need to preserve the meaning of concepts used in everyday transactions of information sharing. The emergence of the Semantic Web [5] has made these transactions, arguably, easier to implement and

---

\* Last names of authors are in alphabetical order.

deploy on a large scale in a distributed environment like the Internet. However, at the same time poses some interesting challenges. For instance, we observe that the demand for knowledge sharing has outstripped the current supply. And even when knowledge sharing is feasible, this is only within the boundaries of a specific system, when certain assumptions hold, and within a specific domain. The reason for this shortcoming is, probably, the very environment and technologies that generated a high demand for sharing: The more ontologies are being deployed on the Semantic Web, the more the demand to share them for the benefits of knowledge sharing and semantic interoperability. The sharing of ontology though, is not a solved problem. It has been acknowledged and researched by the knowledge engineering community for years.

In fact, this problem is related to that of integration of heterogeneous databases conducted in the 1980s and early 1990s [4,39], as far as database schemas can be considered a particular kind of lightweight ontologies. These early efforts in information sharing run into the problem of semantic heterogeneity, which required the identification and representation of all semantics useful in performing schema translation and schema integration. Current efforts on ontology sharing build upon the previous experience, but in the light of more broader a concept of ontology, and of more expressive logical formalisms and theories that better capture the semantics and information flow.

One of the aspects in ontology sharing is to perform some sort of mapping between ontology constructs. That is, given two ontologies, one should be able to map concepts found in one ontology onto the ones found in the other. Further, some research suggest that we should also be able to combine ontologies where the product of this combination will be, at the very least, the intersection of the two given ontologies. These are the dominant approaches that have been studied and applied in a variety of systems (see, for example, [35]).

There are, however, some drawbacks that prevent engineers from benefitting from such systems. Firstly, the assumptions made in devising ontology mappings and in combining ontologies are not always exposed to the community and no technical details are disclosed. Secondly, the systems that perform ontology mapping are often either embedded in an integrated environment for ontology editing or are attached to a specific formalism. Thirdly, in most cases mapping and merging are based on heuristics that mostly use syntactic clues to determine correspondence or equivalence between ontology concepts, but rarely use the meaning of those concepts, i.e., their semantics. Fourthly, most, if not all approaches, do not treat ontological axioms or rules often found in formal ontologies. Finally, ontology mapping as a term has a different meaning in different works merely due to the lack of a formal account of what ontology mapping is. There is an observed lack of theory behind most of the works in this area.

Motivated by these drawbacks we started to work on a method and a theory for ontology mapping and merging. We were determined to tackle these drawbacks so our approach is based on the observation reported in [38] of an essential duality in knowledge sharing, namely that sharing occurs both at the token and at the type level. We draw on sound theoretical ground, but at the same time we

are providing a systematic approach for ontology mapping in mechanised and methodological steps.

In particular, we propose an *Information-Flow-based* method for ontology mapping (hereafter, IF-Map). We are mostly interested in mapping ontologies, but we could extend the approach to merge them, too. IF-Map draws from the works of Schorlemmer on aligning ontologies [38] and on the heuristics defined by Kalfoglou (in [20], pp.95–97), to analyse prospective mappings between ontologies. On the theoretical side, our method is based on the mathematical theory of information flow proposed by Barwise and Seligman [3] and on Kent’s Information Flow Framework for the IEEE standardisation activity of an upper-level ontology [23], and also on his proposed methodology for merging ontologies [22, 24]. The methodological part of IF-Map has also been influenced by the work of Stumme and Maedche on the FCA-Merge method [41].

We outline a scenario and describe the architecture we have built to perform ontology mapping in Section 2. We briefly provide mathematical preliminaries on information flow and channel theory in Section 3, before we proceed to describe our ontology-mapping method in Section 4, together with an example case of its use. In Section 5 we report on an application of our method and elaborate on scalability issues. We discuss related work in Section 6 and summarise the paper in Section 7.

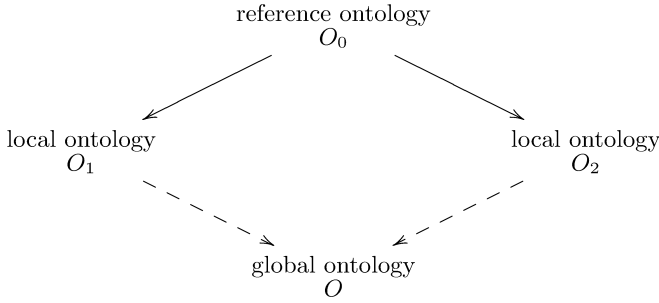
## 2 A Scenario and Architecture for Ontology Mapping

We should clarify some terminological issues before we continue with our architecture for ontology mapping. Some researchers view the mapping process as an integral part of alignment or merging. For example Noy and Musen view merging as the creation of “a single coherent ontology that includes the information from all the sources” and alignment as a process in which “the sources must be made consistent and coherent with one another but kept separately” [34]. Clearly, mapping is an essential aspect of alignment and could also be used to initiate merging. In this paper, we adopt a working view of ontology mapping. We do not intend to alter the original ontologies but rather look for possible mappings between them. To that extent we are compatible with Noy and Musen’s definition of alignment, however, we view alignment as a process that involves merging but without altering the original ontologies. This would be done by constructing a *virtual* ontology consisting of the mapped constructs of the two input ontologies. This view is more akin to those researchers working on the IEEE SUO initiative—the standardisation of an *upper level ontology*—where a meta-level framework for relating ontologies based on the Barwise-Seligman theory of information flow is currently being developed.<sup>1</sup> Although on our scenario we have not yet implemented this aspect, we could extend the existing IF-Map method to perform merging. In the remaining of the paper though, we will focus on ontology mapping only.

<sup>1</sup> We point the interested reader to <http://suo.ieee.org> for more information on this initiative.

## 2.1 Ontology Mapping Scenario

In Figure 1 we illustrate our approach to ontology mapping. In particular, the focus is on the use of information-flow theory as the underpinning mathematical foundation for establishing mappings between two ontologies. We shall formalise these mappings in terms of *local logics* and *logic infomorphisms*, which we introduce in Section 3.



**Fig. 1.** The scenario for ontology mapping.

We assume that, when two communities desire to share knowledge but each one has encoded knowledge according to its own *local ontology*— $O_1$  and  $O_2$  respectively—they have previously agreed upon a common understanding, a *reference ontology*  $O_0$ , in order to favour the sharing of knowledge. Typically each community will have its local ontology populated with its own instances, while the reference ontology will not have instances.

But, although having agreed upon a reference ontology, each community might prefer to communicate via its own local ontology, provided these ontologies  $O_1$  and  $O_2$  (or at least a significant fragment of them) conform to the reference ontology  $O_0$ . In this paper we provide a methodology to establish this conformance by looking for, and further generating, ontology mappings  $O_0 \rightarrow O_1$  and  $O_0 \rightarrow O_2$  from the reference ontology to local ontologies, denoted as solid arrows in Figure 1. The methodology is based upon formalising ontologies as *local logics* and ontology mappings as *logic infomorphisms*, as we shall see in Section 3.5.

The link  $O_1 \leftarrow O_0 \rightarrow O_2$  between local ontologies via the reference ontology, together with the generated ontology mappings from the latter to the former two, provides an *alignment structure* of ontologies that determine uniquely a *global ontology*  $O$ —an ontology that could be either constructed from the alignment structure or else generated ‘on-the-fly’ for the purpose of knowledge sharing. The dashed arrows denote the inclusion of local ontologies  $O_1$  and  $O_2$  into the global ontology  $O$ . This inclusions are also ontology mappings as formalised by logic infomorphisms, and hence our methodology IF-Map could be extended in future to include the merging of ontologies according to an alignment structure.

Figure 1 clearly resembles Kent’s proposed two-step process for conceptual knowledge organisation [22,24]. In fact, the existence of a unique global ontology  $O$  given an alignment  $O_1 \leftarrow O_0 \rightarrow O_2$  consisting of local ontologies and a reference ontology is the consequence of formalising ontologies and ontology mappings as local logics and logic infomorphisms.

2.2 Ontology Mapping Architecture

In Figure 2 we illustrate the process of IF-Map. We have built a step-wise process that consists of four major steps: (a) Ontology harvesting, (b) translation, (c) infomorphism generation, and (d) display of results. In the ontology harvesting step we perform our acquisition. We acquire ontologies by applying a variety of methods: Downloading existing ontologies from ontology libraries (for example, from the Ontolingua [12] or WebOnto [10] servers); editing them in ontology editors (for example, in Protégé [16]); or extracting them from the Web. The latter is ongoing research in the AKT project (<http://www.aktors.org>), where we are writing scripting programs to crawl the Web and harvest RDF-encoded resources to semi-automatically construct and populate ontologies. We will not expand on this topic here as it is peripheral to our theme of ontology mapping. As a result of our versatile ontology acquisition step, we have to deal with a variety of ontology representation languages ranging from KIF [15] and Ontolingua to OCML [31], RDF [27], Prolog, and native Protégé knowledge bases.

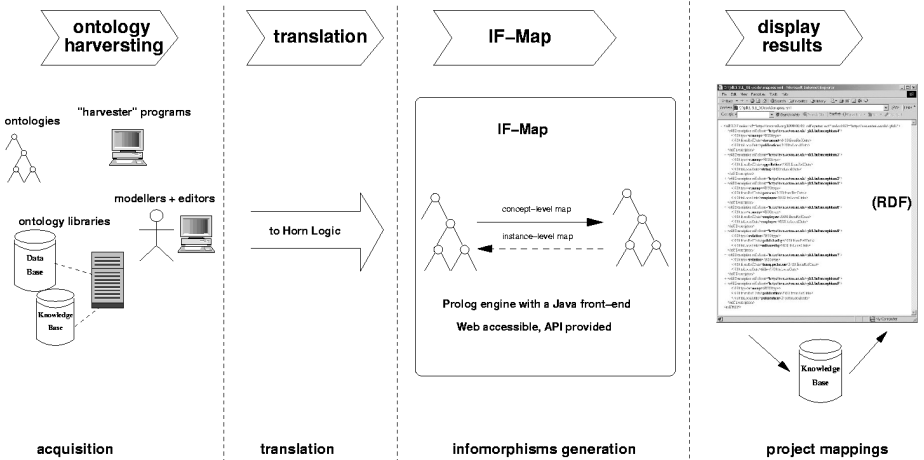


Fig. 2. The IF-Map architecture.

This introduces the second step in our process, that of translation. Since we have declaratively specified the IF-Map method in Horn logic and execute it with a Prolog interpreter, we partially translate the above formats to Prolog clauses.

Our translator programs are either written in-house, or whenever available, we use public translators. For example, there are public RDF-to-Prolog translators<sup>2</sup> as well as Ontolingua-to-Prolog translators. In most of these cases though, we found it practical to write our own ones. We did that to have a partial translation, customised for the purposes of ontology mapping. Furthermore, as it has been reported in a large-scale experiment with publicly available translators [7], the Prolog code produced is not elegant or even executable. Our own translators are customised to translate—from KIF, Ontolingua, and Protégé knowledge bases into Prolog clauses—those constructs that are needed for IF-Map: Class taxonomy, relations and representative instances for classes. Thus, we deliberately neglect constructs such as documentation slots, separation of own-slots and template-slots and other object-oriented modelling primitives used in Ontology languages (such as KIF or Ontolingua<sup>3</sup>) as they are not useful for IF-Map and their absence from the translated Prolog code does not invalidate their meaning. For Protégé knowledge bases we used the built-in Java API to obtain the constructs we wanted, and for RDF we used publicly available RDF to Prolog translators. The issue of a full-blown translation from one formalism to another is a difficult problem, and Corrêa da Silva et al. [7] offer an account on the effort involved.

The next step in our process is the main mapping mechanism: The IF-Map method, which we describe in Section 4. We have written a Java front-end to the Prolog-written IF-Map program so that we can access it from the Web, and we are currently in the process of writing a Java API to enable external calls to it from other systems. This step will find logic isomorphisms, if any, between the two ontologies under examination, and in the last step of our process we display them in RDF format. This step involves translating back from Prolog clauses to RDF triples by means of an intermediary Java layer, where RDF is being produced using the Jena RDF API [28]. Finally, we store the results in a knowledge base for future reference and maintenance.

Before proceeding to an example case of the deployment this architecture we shall introduce the theoretical background of IF-Map. Therefore, in the next section we expand on logic isomorphisms and other channel-theoretic notions, and give a formal account of ontology mapping.

### 3 Preliminaries

In order to give a formal characterisation of ontology mapping we start from the following basic assumption: If two communities desire to share information but their ontologies differ, we need to align and map their ontologies for the information to flow between them. Consequently, it would be desirable to base

---

<sup>2</sup> Like the one from Wielemaker downloadable from <http://www.swi-prolog.org/packages/rdf2pl.html>

<sup>3</sup> We briefly describe the principles we used to partially translate from Ontolingua to Prolog in [20], pp.105–107.



a theory of ontology mapping upon a mathematical theory that is capable of describing under which circumstances information flow occurs.

Although there is no such a theory yet, the most promising effort towards such a theory was initiated by Barwise and Perry with situation semantics [2], which was then been further developed by Devlin into a theory of information [8]. Barwise and Seligman's channel theory is currently the latest stage of this endeavour [3], and is probably the best place to start establishing a foundation for a theory of ontology mapping.

### 3.1 Information Flow

Barwise and Seligman propose a mathematical model that aims at establishing the laws that govern the flow of information. It is a general model that attempts to describe the information flow in any kind of distributed system, ranging from actual physical systems like a flashlight connecting a bulb to a switch and a battery, to abstract systems such as a mathematical proof connecting premises and hypothesis with inference steps and conclusions.

It is based on the understanding that information flow results from regularities in a distributed system, and that it is by virtue of regularities among the connections that information of some components of a system carries information of other components. The more regularities the system has, the more information flows; the more random the system is constituted the less information will be able to flow among its components.

As a notion of a component carrying information about another component, Barwise and Seligman follow Dretske's characterisation [11], in which assertions of the sort

*The rifle shot carried the information that the king was dead to the whole city.*

or

*The greyness of the sky carries the information that a storm is approaching.*

are abstractly characterised as following a common pattern

*a's being F carries/bears/conveys the information that b is G.*

In fact, it is a particular rifle shot that will inform us of the king's death, and hence it is a particular event *token e* of the *type* 'rifle shot' that will carry this information. A rifle shot 50 years later will definitely not carry the information that the present king is dead as will the present rifle shot not indicate that the king of the neighbouring city is dead. It is the fact that the event token of the rifle shot is connected spatio-temporally with the event of the king's death that we know that the information flows. Channel theory formalises this crucial aspect of information flow, namely that it is the tokens and its connections that carry information, so that information flow involves both types and tokens.

3.2 Aligning and Merging Ontologies

With respect to the problem that concerns us here—that of mapping ontologies—the same abstract pattern arises. Two communities with different ontologies will be able to share information when they are capable of establishing connections among their tokens in order to infer the relationship among their types. Let us develop an example taken from [40], which shows the issues one has to take into account when attempting to align the English concepts *river* and *stream* with the French concepts *fleuve* and *rivière*. According to Sowa,

In English, size is the feature that distinguishes *river* from *stream*; in French, a *fleuve* is a river that flows into the sea, and a *rivière* is either a river or a stream that runs into another river. [40]

This explains how the concepts need to be merged. Notice that the above quote requires an agreed understanding on how to distinguish between big and small rivers, and between rivers that run into a sea or into other rivers, yielding four types of instances of ‘water-flowing entities’: *big-into-sea*, *big-into-river*, *small-into-sea*, and *small-into-river*.

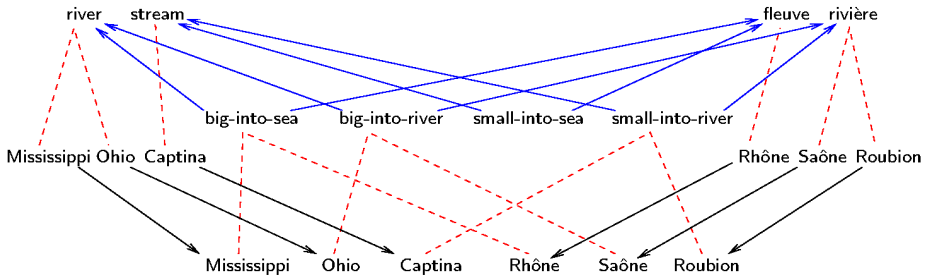


Fig. 3. Alignment by means of a pair of maps

Figure 3 shows how both, English and French speakers, base their concepts upon this agreed understanding, although English and French speakers don’t distinguish between some types of instances. For example, English speakers call both, *big-into-sea* and *big-into-river*, a *river*, while French speakers don’t distinguish between *big-into-river* and *small-into-river*, and call both types a *rivière*. The agreed understanding is materialised by two maps that form the alignment. It requires the classification of particular instances of *river*, *stream*, *fleuve*, and *rivière* according with the agreed understanding, since it is this agreed way of classification which will determine how the concepts *river*, *stream*, *fleuve*, and *rivière* are going to be related to each other.

The ultimate goal is to determine the connections that link particular instances of type *river* or *stream* with particular instances of type *fleuve* or *rivière*,

in a way that they respect the agreed understanding. This is done by connecting only those instances that conform to the same type according to the agreed understanding, as illustrated in Figure 4.

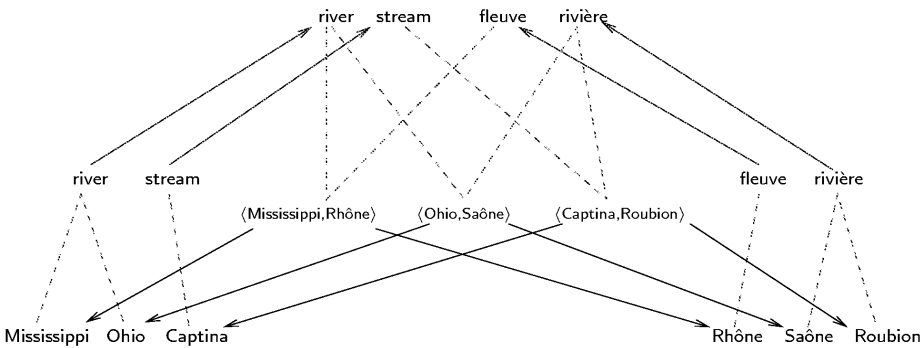


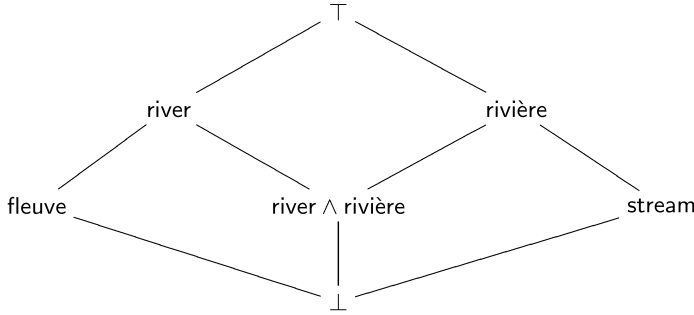
Fig. 4. Aligning ontologies through a pair of maps

The resulting classification of connections  $\langle \text{Mississippi, Rhône} \rangle$ ,  $\langle \text{Ohio, Saône} \rangle$ , and  $\langle \text{Captina, Roubion} \rangle$  into the four concepts *river*, *stream*, *fleuve*, and *rivière*, determines a theory of how these concepts are related (e.g., that a *fleuve* is also a *river*, or that a *stream* is also a *rivière*, but not vice versa). Figure 4 shows what in channel theory is known as a an *information channel* [3]. It captures, by means of two pairs of contra-variant functions, an existing duality between concepts and instances: Each pair consists of a map of concepts on the so called *type level* and map of instances on the so called *token level*, and pointing in the opposite direction. From a channel-theoretic perspective, Figure 4 actually illustrates us that sharing knowledge involves a flow of information that crucially depends on how the instances of different agents are connected together. The following table shows the *classification relation*, i.e., the connections as classified according to the concept types involved in the example:

	river	stream	fleuve	rivière
$\langle \text{Mississippi, Rhône} \rangle$	1	0	1	0
$\langle \text{Ohio, Saône} \rangle$	1	0	0	1
$\langle \text{Captina, Roubion} \rangle$	0	1	0	1

The merged set of concepts  $\{\text{river, stream, fleuve, rivière}\}$  has an additional structure that we can deduce from the way the connections of instances are classified with respect to these concepts. Through techniques from formal concepts analysis [14], for instance, we can make such structure explicit in the form of a *concept lattice*, as shown in Figure 5. The concept hierarchy represented in this

lattice depends on the choice of instances and its classification with respect to the agreed understanding. The fact that no instances were classified as of type *small-into-sea* was crucial in this example. Notice, also, that the resulting lattice has a node labelled with the concept  $\text{river} \wedge \text{rivière}$ , which is a formal concept that did not exist in the original vocabularies. It corresponds to the instances of ‘water-flowing entities’ that, although big, flow into other rivers, like Ohio and Saône.



**Fig. 5.** Concept lattice

### 3.3 Channel Theory

Channel theory provides the required mathematical machinery to describe the information flow between communities in terms of their connection by means of tokens and types. In order to specify the regularities of each component in a distributed systems channel theory uses the notion of a *local logic*. Separate interacting components will typically use different vocabularies, i.e., they will use different systems of *types*, and also *tokens* and of how these *classified* according to the types.

In addition, each community will have its own particular *constraints* that describe the local behaviour of their instances with respect to their system of types. A *local logic* brings all these ideas together:

**Definition 1.** A local logic is a quadruple  $\mathcal{L} = (I, T, \models, \vdash)$ , where

1.  $I$  is a set of instances (also called tokens);
2.  $T$  is a set of types;
3.  $\models$  is a classification relation, a binary relation between elements of  $I$  and  $T$ ;
4.  $\vdash$  is a consequence relation, a binary relation between subsets of  $T$ ;

There are two parts of a local logic that are of particular importance in the channel-theory framework. The first one is the triple  $(I, T, \models)$ , and is called

the *classification* of the local logic, because the binary relation  $\models$  determines a classification of instances in  $I$  with respect to types in  $T$ . Thus,  $x \models a$  means that instance  $x \in I$  is classified as of type  $a \in T$ .

The second important part is the pair  $(T, \vdash)$ , which is called the *theory* of the local logic. This theory is specified by a set of *sequents*  $\langle \Gamma, \Delta \rangle$ , i.e., pairs where  $\Gamma, \Delta \subseteq T$ . The set of types  $\Gamma$  is to be interpreted conjunctively, the set  $\Delta$  disjunctively, so that an instance  $x \in I$  *satisfies* a sequent  $\langle \Gamma, \Delta \rangle$  provided that, if  $x$  is of *every* type in  $\Gamma$ , then  $x$  is of *some* type in  $\Delta$ . Sequents that belong to the theory of a logic are called *constraints* and denoted  $\Gamma \vdash \Delta$ . Theories of local logics must satisfy the following conditions of *regularity*:<sup>4</sup>

1. Identity:  $a \vdash a$ , for all  $a \in T$ ;
2. Weakening: If  $\Gamma \vdash \Delta$  then  $\Gamma, \Gamma' \vdash \Delta, \Delta'$ , for all  $\Gamma, \Gamma', \Delta, \Delta' \subseteq T$ ;
3. Global Cut: If  $\Gamma, T'_0 \vdash \Delta, T'_1$  for each partition<sup>5</sup>  $\langle T'_0, T'_1 \rangle$  of any  $T' \subseteq T$ , then  $\Gamma \vdash \Delta$ , for all  $\Gamma, \Delta \subseteq T$ .

There is an additional element in local logics that we have deliberately left out in Definition 1. Ideally, instances of a local logic adhere to its constraints, although, we cannot presuppose this in general, and exceptions may occur. Local logics also distinguish a subset  $N \subseteq I$  of *normal instances* that must satisfy all constraints of the local logic. The idea of a normal instance is needed if we want to model reasonable but unsound flow of information. For the purposes of IF-Map, though, we shall assume that all instances are normal. Such logics are said to be *sound*.

For information to flow between separate components of a distributed system, we need to link local logics that characterise components in a sensible way. This will essentially affect the system of classifications and its associated theory, but in a way that allows the information to flow. This latter is captured with the idea of a *logic infomorphism*:

**Definition 2.** A logic infomorphism  $f : \mathcal{L} \rightleftharpoons \mathcal{L}'$  from local logic  $\mathcal{L} = (I, T, \models, \vdash)$  to local logic  $\mathcal{L}' = (I', T', \models', \vdash')$  is a contra-variant pair of functions  $f = \langle f^*, f_\star \rangle$ , where  $f^* : T \rightarrow T'$  and  $f_\star : I' \rightarrow I$ , such that,

1. for all  $x \in I'$  and  $a \in T$ ,  $f_\star(x) \models a$  if and only if  $x \models' f^*(a)$ ;
2. for all  $\Gamma, \Delta \subseteq T$ , if  $\Gamma \vdash \Delta$ , then  $f^*[\Gamma] \vdash' f^*[\Delta]$ .<sup>6</sup>

The restriction of a logic infomorphism to the classification part of local logics is called only an *infomorphism*.

<sup>4</sup> Regularity arises from the observation that, given a classification of instances to types, the set of all sequents that are satisfied by all instances do fulfil these properties.

<sup>5</sup> A partition of  $T'$  is a pair  $\langle T'_0, T'_1 \rangle$  of subsets of  $T'$ , such that  $T'_0 \cup T'_1 = T'$  and  $T'_0 \cap T'_1 = \emptyset$ ;  $T'_0$  and  $T'_1$  may themselves be empty (hence they form actually a quasi-partition).

<sup>6</sup>  $f^*[\Gamma]$  and  $f^*[\Delta]$  denote the set images of sets  $\Gamma$  and  $\Delta$  along function  $f^*$ , respectively.

### 3.4 Ontologies and Ontology Morphisms

For the purposes of IF-Map described in this paper, we adopt a definition of ontology that includes some of the core components that are usually part of an ontology: *Concepts*, organised in an *is-a hierarchy*, which we capture with a partial-order relation ' $\leq$ '; *relations* defined over these concepts; and notions of *disjointness* of two concepts—when no instance can be considered of both concepts—and *coverage* of two concept—when all instances are covered by two concepts.<sup>7</sup>

Disjointness and coverage are typically specified by means of ontological axioms. IF-Map takes these kind of axioms into account including disjointness and coverage into the hierarchy of concepts by means of two binary relations ' $\perp$ ' and ' $|$ ', respectively.

**Definition 3.** An ontology is a tuple  $\mathcal{O} = (C, R, \leq, \perp, |, \sigma)$  where

1.  $C$  is a finite set of concept symbols;
2.  $R$  is a finite set of relation symbols;
3.  $\leq$  is a reflexive, transitive and anti-symmetric relation on  $C$  (a partial order);
4.  $\perp$  is a symmetric and irreflexive relation on  $C$  (disjointness);
5.  $|$  is a symmetric relation on  $C$  (coverage); and
6.  $\sigma : R \rightarrow C^+$  is the function assigning to each relation symbol its arity; the functor  $(-)^+$  sends a set  $C$  to the set of finite tuples whose elements are in  $C$ .

When discarding binary relations  $\perp$  and  $|$ , this definition is equivalent to that of a *core ontology* in [41].

When an ontology  $\mathcal{O} = (C, R, \leq, \perp, |, \sigma)$  is used in some particular application domain, we need to populate it with instances. First, we will have to classify objects of a set  $X$  according to the concept symbols in  $C$  by defining a binary classification relation  $\models_C$ . This will determine a classification  $\mathbf{C} = (X, C, \models_C)$ . Next, we will have to specify over which instances the relations represented by the symbols in  $R$  are to hold, thus classifying finite tuples of objects of  $X$  to the relation symbols in  $R$  by defining a binary classification relation  $\models_R$ . This will determine a classification  $\mathbf{R} = (X^+, R, \models_R)$ . Both classifications will have to be defined in such a way that the partial order  $\leq$ , the disjointness  $\perp$ , the coverage  $|$ , and the arity function  $\sigma$  are respected:

**Definition 4.** A populated ontology is a tuple  $\tilde{\mathcal{O}} = (\mathbf{C}, \mathbf{R}, \leq, \perp, |, \sigma)$  such that  $\mathbf{C} = (X, C, \models_C)$  and  $\mathbf{R} = (X^+, R, \models_R)$  are classifications and  $\mathcal{O} = (C, R, \leq, \perp, |, \sigma)$  is an ontology; we say the ontology is correct when, for all  $x, x_1, \dots, x_n \in X$ ,  $c, d \in C$ ,  $r \in R$ , and  $\sigma(r) = \langle c_1, \dots, c_n \rangle$ ,

<sup>7</sup> Both disjointness and coverage can easily be extended to more than two concepts, although we stay with binary relations, for the ease of presentation.

1. if  $x \models_{\mathbf{C}} c$  and  $c \leq d$ , then  $x \models_{\mathbf{C}} d$ ;
2. if  $x \models_{\mathbf{C}} c$  and  $c \perp d$ , then  $x \not\models_{\mathbf{C}} d$ ;
3. if  $c \mid d$ , then  $x \models_{\mathbf{C}} c$  or  $x \models_{\mathbf{C}} d$ ;
4. if  $\langle x_1, \dots, x_n \rangle \models_{\mathbf{R}} r$  then  $x_i \models_{\mathbf{R}} c_i$ , for all  $i = 1, \dots, n$ .

Notice that we write  $\tilde{\mathcal{O}}$  for a populated ontology and  $\mathcal{O}$  for the respective unpopulated one.

Transformations of mathematical structures that preserve the structure that characterises them are usually described with homomorphism (or morphisms, for short). Thus, we study the mapping of ontologies through the morphisms of those mathematical structures we have defined for ontologies in Definition 3. The concept of ‘populated ontology’ is central to our approach to ontology mapping, and we shall use it later in Proposition 1 in order to justify the following definition of an ontology morphism:

**Definition 5.** *Given two ontologies  $\mathcal{O} = (C, R, \leq, \perp, \mid, \sigma)$  and  $\mathcal{O}' = (C', R', \leq', \perp', \mid', \sigma')$ , an ontology morphism  $\langle f^*, g^* \rangle : \mathcal{O} \rightarrow \mathcal{O}'$  is a pair of functions  $f^* : C \rightarrow C'$  and  $g^* : R \rightarrow R'$ , such that, for all  $c, d \in C$ ,  $r \in R$ , and  $\sigma(r) = \langle c_1, \dots, c_n \rangle$ ,*

1. if  $c \leq d$ , then  $f^*(c) \leq' f^*(d)$ ;
2. if  $c \perp d$ , then  $f^*(c) \perp' f^*(d)$ ;
3. if  $c \mid d$ , then  $f^*(c) \mid' f^*(d)$ ;
4. if  $\sigma'(g^*(r)) = \langle c'_1, \dots, c'_n \rangle$ , then  $c'_i \leq' f^*(c_i)$ , for all  $i = 1, \dots, n$ .

### 3.5 Information Flow between Ontologies

Our approach to ontology mapping is built upon the assumption that, in the context of channel theory, local logics characterise ontologies.

Hence, a populated ontology  $\tilde{\mathcal{O}} = (\mathbf{C}, \mathbf{R}, \leq, \perp, \mid, \sigma)$ —with  $\mathbf{C} = (X, C, \models_{\mathbf{C}})$ —determines a local logic  $\mathcal{L} = (X, C, \models_{\mathbf{C}}, \vdash)$  whose theory  $(C, \vdash)$  is given by the smallest regular consequence relation (i.e., the smallest relation closed under Identity, Weakening, and Global Cut) such that, for all  $c, d \in C$

$$\begin{array}{lll} c \vdash d & \text{iff} & c \leq d \\ c, d \vdash & \text{iff} & c \perp d \\ \vdash c, d & \text{iff} & c \mid d \end{array}$$

The characterisation of an ontology as a local logic justifies the IF-Map method presented in next section, which stems from our intention—explained in Section 2—to map an unpopulated ontology  $\mathcal{O} = (C, R, \leq, \perp, \mid, \sigma)$  to a populated one  $\tilde{\mathcal{O}} = (\mathbf{C}', \mathbf{R}', \leq', \perp', \mid', \sigma')$ , by looking at the information flow. For this reason we “artificially” populate the concept types given in  $C$  and the relation types given in  $R$  to obtain classifications  $\mathbf{C} = (Y, C, \models_{\mathbf{C}})$  and  $\mathbf{R} = (Z, R, \models_{\mathbf{R}})$  (unlike a populated ontology, the instances of  $R$  need not to be finite tuples of instances of  $C$ ), and further establish infomorphisms  $f : \mathbf{C} \rightleftharpoons \mathbf{C}'$  and  $g : \mathbf{R} \rightleftharpoons \mathbf{R}'$ , such that their type-level components  $f^*$  and  $g^*$  constitute an ontology morphism;

because in that case we know that the populated ontology  $\tilde{\mathcal{O}}'$  will be a *correct extension* of  $\mathcal{O}$ , in the sense that the images of  $\tilde{\mathcal{O}}'$ 's instances conform to  $\mathcal{O}$ , as stated in the following proposition:

**Proposition 1.** *Let  $\mathcal{O} = (C, R, \leq, \perp, |, \sigma)$  be an (unpopulated) ontology, and let  $\tilde{\mathcal{O}}' = (\mathbf{C}', \mathbf{R}', \leq', \perp', |', \sigma')$  be a populated ontology with classifications  $\mathbf{C}' = (X', C', \models_{\mathbf{C}'})$ ,  $\mathbf{R}' = (X'^+, R', \models_{\mathbf{R}'})$ . Let  $\mathbf{C} = (Y, C, \models_{\mathbf{C}})$  and  $\mathbf{R} = (Z, R, \models_{\mathbf{R}})$  be two classifications whose types are the concept and relation types of  $\mathcal{O}$ . If  $\tilde{\mathcal{O}}'$  is correct and  $f : \mathbf{C} \rightleftharpoons \mathbf{C}'$  and  $g : \mathbf{R} \rightleftharpoons \mathbf{R}'$  are infomorphisms, such that  $(f^*, g^*) : \mathcal{O} \rightarrow \mathcal{O}'$  is an ontology morphism, then, for all  $x, x_1, \dots, x_n \in X$ ,  $c, d \in C$ ,  $r \in R$ , and  $\sigma(r) = \langle c_1, \dots, c_n \rangle$ ,*

1.  $f_*(x) \models_{\mathbf{C}} c$  and  $c \leq d$  imply  $f_*(x) \models_{\mathbf{C}} d$ ;
2.  $f_*(x) \models_{\mathbf{C}} c$  and  $c \perp d$  imply  $f_*(x) \not\models_{\mathbf{C}} d$ ;
3.  $c | d$  implies  $f_*(x) \models_{\mathbf{C}} c$  or  $f_*(x) \models_{\mathbf{C}} d$ ;
4.  $g_*(\langle x_1, \dots, x_n \rangle) \models r$  implies  $f_*(x_i) \models c_i$ , for all  $i = 1, \dots, n$ .

*Proof.*

1. Suppose  $f_*(x) \models_{\mathbf{C}} c$  and  $c \leq d$ . Since  $f$  is an infomorphism,  $x \models_{\mathbf{C}'} f^*(c)$ . Furthermore,  $c \leq d$  implies  $f^*(c) \leq' f^*(d)$  because  $\langle f^*, g^* \rangle$  is an ontology morphism; consequently,  $x \models_{\mathbf{C}'} f^*(d)$ . Finally, since  $f$  is a infomorphism,  $f_*(x) \models_{\mathbf{C}} d$ .
2. Analogous to 1.
3. Analogous to 1.
4. Suppose  $g_*(\langle x_1, \dots, x_n \rangle) \models r$ . Because  $g$  is an infomorphism,  $\langle x_1, \dots, x_n \rangle \models g^*(r)$ . Let  $\sigma(g^*(r)) = \langle c'_1, \dots, c'_n \rangle$ . By the correctness of  $\tilde{\mathcal{O}}'$ ,  $x_i \models_{\mathbf{C}'} c'_i$ , for all  $i = 1, \dots, n$ , and since  $\langle f^*, g^* \rangle$  is an ontology morphism,  $x_i \models_{\mathbf{C}} f^*(c'_i)$ , for all  $i = 1, \dots, n$ . Consequently, and because  $f$  is a infomorphism,  $f_*(x_i) \models c_i$ , for all  $i = 1, \dots, n$ .

In the next section we describe the ontology mapping method based on the above characterisation of ontologies as local logics, and ontology morphisms as logic infomorphisms.

## 4 The IF-Map Method

We propose a method for mapping ontologies that draws on the mathematical foundations of information-flow, and we shall use a small easy-to-follow example to illustrate the core parts of IF-Map.

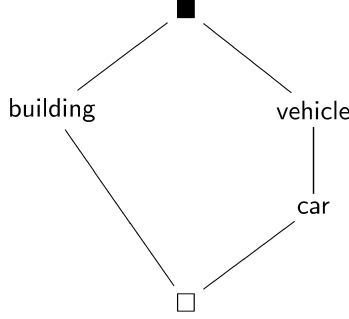
### 4.1 Reference and Local Ontology

Let us assume that we want to map two ontologies, a reference ontology with a local ontology. We follow the scenario given in Section 2 and assume that the reference ontology has no instances defined, just concept types and constraints over those types. The local ontology, however, has instances classified under its concept types according to a classification relation.



Let the reference ontology be  $\mathcal{O} = (C, R, \leq, \perp, |, \sigma)$ , with

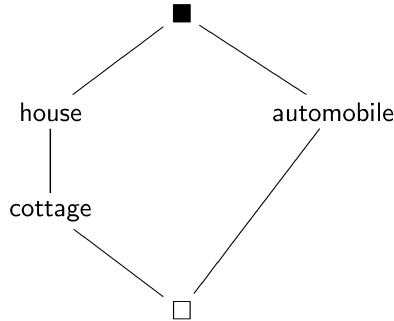
- concepts  $C = \{\text{building}, \text{vehicle}, \text{car}\}$ ;
- relations  $R = \{\text{hasParkingSpaceFor}\}$ ;
- arities  $\sigma(\text{hasParkingSpaceFor}) = \langle \text{building}, \text{vehicle} \rangle$ ; and
- partial order  $\leq$ , disjointness  $\perp$ , and coverage  $|$  as defined by the following lattice:



where ■ is the top and □ is the bottom of the lattice, i.e.,  $\text{building} | \text{vehicle}$  and  $\text{building} \perp \text{vehicle}$ .

Let the local ontology be  $\mathcal{O}' = (C', R', \leq', \perp', |', \sigma')$ , with

- concepts  $C' = \{\text{house}, \text{cottage}, \text{automobile}\}$ ;
- relations  $R' = \{\text{hasGarageFor}, \text{hasShelterFor}\}$ ;
- arities  $\sigma'(\text{hasGarageFor}) = \langle \text{house}, \text{automobile} \rangle$ ,  $\sigma'(\text{hasShelterFor}) = \langle \text{cottage}, \text{automobile} \rangle$ ; and
- partial order  $\leq'$ , disjointness  $\perp'$ , and coverage  $|'$  as defined by the following lattice:



where ■ is the top and □ is the bottom of the lattice, i.e.,  $\text{house} |' \text{automobile}$  and  $\text{house} \perp' \text{automobile}$ .

The local ontology, unlike reference ontology, is populated with instances  $X = \{cabrio, my-home, 4wd, new-inn, old-inn, coupe\}$ , which are classified as follows,

$\models_{C'}$	house	cottage	automobile
<i>cabrio</i>	0	0	1
<i>my-home</i>	1	0	0
<i>4wd</i>	0	0	1
<i>new-inn</i>	1	1	0
<i>old-inn</i>	1	1	0
<i>coupe</i>	0	0	1

This table contains the following information: *cabrio*, *4wd* and *coupe* are automobiles, *my-home* is a house, *new-inn* and *old-inn* are specific kinds of houses, namely cottages. It specifies the classification  $C' = (X, C', \models_{C'})$ .

## 4.2 Characterisation as Local Logics

In order to automatically find mappings between the reference and the local ontologies that conform to the definition of ontology morphism given in Definition 5, we will need to look for logic infomorphisms between the local logics that characterise these ontologies. First we shall concentrate on the concepts symbols and leave the relation symbols for Section 4.4.

The reference ontology, which is not populated, is characterised by the following local logic. Its regular theory  $(C, \vdash)$  has concept symbols as types, and  $\vdash$  is the smallest consequence relation closed by Identity, Weakening, and Global Cut that includes the following constraints:

$$\begin{aligned}
 &\text{building, vehicle} \vdash \\
 &\quad \text{car} \vdash \text{vehicle} \\
 &\quad \vdash \text{building, vehicle}
 \end{aligned}$$

Recall that the comma on the left-hand side of these constraints has conjunctive force whereas on the right-hand side it has disjunctive force. Following this, we can give a declarative reading of the above constraints: Nothing is both a building and a vehicle; all cars are vehicles; and everything is either a building or a vehicle.

Since the reference ontology does not have instances of its own, we will need to provide the theory with a set of instances and a classification of these instances with respect to the types. This can be achieved due to a Fundamental Representation Theorem (see [3]), which states that a local logic that is generated from the structure given in a classification is equivalent to the local logic constructed from its theory by generating formally created instances as follows:

1. We take as instances  $Y$  all those sequents  $\langle \Gamma, \Delta \rangle$  that
  - form a partition of the set of concepts ( $\Gamma \cup \Delta = C$  and  $\Gamma \cap \Delta = \emptyset$ ); and
  - are *not* constraints of the theory ( $\Gamma \not\vdash \Delta$ )

For the theory given above, these sequents are

$$\begin{aligned} &\langle \{\text{vehicle}, \text{car}\}, \{\text{building}\} \rangle \\ &\langle \{\text{building}\}, \{\text{vehicle}, \text{car}\} \rangle \\ &\langle \{\text{vehicle}\}, \{\text{building}, \text{car}\} \rangle \end{aligned}$$

2. We then classify these instances according to the concepts that occur in the left-hand side component of the sequent:

$\models_{\mathcal{C}}$	building	vehicle	car
$\langle \{\text{vehicle}, \text{car}\}, \{\text{building}\} \rangle$	0	1	1
$\langle \{\text{building}\}, \{\text{vehicle}, \text{car}\} \rangle$	1	0	0
$\langle \{\text{vehicle}\}, \{\text{building}, \text{car}\} \rangle$	0	1	0

The generation of instances by means of sequents and their classification to types may not seem obvious, but it is based on the fact that these sequents ‘code’ the content of the classification table (the left-hand sides of the these sequents indicate which columns of the table bear a ‘1’, while the right-hand sides indicate which columns bear a ‘0’). The local logic that characterises the reference ontology, i.e., the ontology given by  $\mathcal{O}$ , is  $\mathcal{L} = (Y, C, \models_{\mathcal{C}}, \vdash)$ .

The local ontology is populated, and hence has already instances and a classification relation. We only need to derive the theory of the local logic that characterises its concept hierarchy as specified in the lattice above. Therefore, its regular theory  $(C', \vdash')$  has concept symbols as types, and  $\vdash'$  is the smallest consequence relation closed by Identity, Weakening, and Global Cut that includes the following constraints:

$$\begin{aligned} \text{house}, \text{automobile} &\vdash' \\ \text{cottage} &\vdash' \text{house} \\ &\vdash' \text{house}, \text{automobile} \end{aligned}$$

The local logic that characterises the local ontology is, thus,  $\mathcal{L}' = (X, C', \models_{\mathcal{C}'}, \vdash')$ .

### 4.3 Generation of Ontology Morphisms via Infomorphisms

To map the ontologies, we must find an ontology morphism from  $\mathcal{O}$  to  $\mathcal{O}'$ , which means that there must exist a logic infomorphism  $f = \langle f^*, f_* \rangle$  from local logic  $\mathcal{L}$  to local logic  $\mathcal{L}'$ . This amounts to first look for an infomorphism between their respective classifications:

- A map of concepts  $f^* : C \rightarrow C'$  (concept-level);
- a map  $f_*$  from instances *cabrio*,  $\dots$ , *coupe* to the formally created instances of the reference ontology (instance-level);

Note that an ontology morphism, as defined in Definition 5, only captures the concept-level of the infomorphism, i.e.  $f^*$ . But  $f^*$  has to map the concepts in a way that it respects the hierarchy. One possible way would be:

$$\begin{aligned} f^*(\text{building}) &= \text{house} \\ f^*(\text{vehicle}) &= \text{automobile} \\ f^*(\text{car}) &= \text{automobile} \end{aligned}$$

However, we should point out that the automatic generation of these maps is growing exponentially. But we can use the constraint that the map has to respect the concept hierarchy and limit the number of possible maps. Once the map is fixed, there is at most one acceptable way to map the instances in order for  $f$  to be an infomorphism.

We do that by building the following table that represents an infomorphism:<sup>8</sup> We label rows by the instances in  $X = \{\text{cabrio}, \dots, \text{coupe}\}$  of the local ontology, and columns by the reference ontology's concepts  $C = \{\text{building}, \text{vehicle}, \text{car}\}$ . We put under each of these concepts the values of the column of the local ontology's classification table that corresponds to the image along the map of ontologies  $f^*$  (i.e., under **building** we put the column of **house**):

	building	vehicle	car
<i>cabrio</i>	0	1	1
<i>my-home</i>	1	0	0
<i>4wd</i>	0	1	1
<i>new-inn</i>	1	0	0
<i>old-inn</i>	1	0	0
<i>coupe</i>	0	1	1

Each row should identify (taking into account the classification table of the reference ontology) the formal instances to which each local instance should be mapped onto. Hence, we have the following instance-component of our infomorphism:

$$\begin{aligned} f_*(\text{cabrio}) &= \langle \{\text{vehicle}, \text{car}\}, \{\text{building}\} \rangle \\ f_*(\text{my-home}) &= \langle \{\text{building}\}, \{\text{vehicle}, \text{car}\} \rangle \\ f_*(\text{4wd}) &= \langle \{\text{vehicle}, \text{car}\}, \{\text{building}\} \rangle \\ f_*(\text{new-inn}) &= \langle \{\text{building}\}, \{\text{vehicle}, \text{car}\} \rangle \\ f_*(\text{old-inn}) &= \langle \{\text{building}\}, \{\text{vehicle}, \text{car}\} \rangle \\ f_*(\text{coupe}) &= \langle \{\text{vehicle}, \text{car}\}, \{\text{building}\} \rangle \end{aligned}$$

We can also interpret the above table (and its resulting mapping of instances) as follows: *cabrio* is classified as both a vehicle and a car, according to the reference ontology. No other classification is possible without violating the definition of infomorphism. If *cabrio* was a vehicle but not a car, the local ontology would have been classifying its instances in a way that does not conform to the reference ontology and the fixed map of concepts.

<sup>8</sup> Infomorphisms can themselves be represented by means of classification tables; this draws on theoretical work based on Chu spaces [17,1,36].

#### 4.4 Relations and Their Arities

In order to constrain the search space when infomorphisms are generated in an automated way, we use ontological relations to guide the classification process that will result in the ontology mapping, namely by looking for infomorphisms  $g : \mathbf{R} \rightarrow \mathbf{R}'$  in a similar fashion as before. So, in our example case, we have the following relation defined in the reference ontology:

**hasParkingSpaceFor** : building  $\times$  vehicle

that is, the binary relation **hasParkingSpaceFor** holds over **building** and **vehicle**. Similarly, in the local ontology we have the following two binary relations:

**hasGarageFor** : house  $\times$  automobile

**hasShelterFor** : cottage  $\times$  automobile

These **Local** relations could be used to classify pairs of local instances:

	hasShelterFor hasGarageFor	
$\langle my-home, cabrio \rangle$	0	0
$\langle my-home, 4wd \rangle$	0	0
$\langle my-home, coupe \rangle$	1	1
$\langle new-inn, cabrio \rangle$	1	0
$\langle new-inn, 4wd \rangle$	0	0
$\langle new-inn, coupe \rangle$	1	0
$\langle old-inn, cabrio \rangle$	0	0
$\langle old-inn, 4wd \rangle$	1	0
$\langle old-inn, coupe \rangle$	0	0

That is, my home has a garage (also considered a shelter) only for a coupe, the new inn has a shelter for a cabrio and a coupe, and the old inn has shelter for a 4wd. We then take these pairs and classify them according to the concepts of the reference ontology to determine the mapping of these ontologies:

1. Generate a classification of the above pairs with respect to the reference ontology's relation, by taking any of the two columns of the table above; this gives us two possibilities to explore, suppose we choose:

	hasParkingSpaceFor
$\langle my-home, cabrio \rangle$	0
$\langle my-home, 4wd \rangle$	0
$\langle my-home, coupe \rangle$	1
$\langle new-inn, cabrio \rangle$	1
$\langle new-inn, 4wd \rangle$	0
$\langle new-inn, coupe \rangle$	1
$\langle old-inn, cabrio \rangle$	0
$\langle old-inn, 4wd \rangle$	1
$\langle old-inn, coupe \rangle$	0

This is the column corresponding to **hasShelterFor**, and consequently  $g^*(\text{hasParkingSpaceFor}) = \text{hasShelterFor}$ .

2. The arity of relation `hasParkingSpaceFor` forces to classify the instances as follows:

	building vehicle car	
<i>cabrio</i>		1
<i>my-home</i>	1	
<i>4wd</i>		1
<i>new-inn</i>	1	
<i>old-inn</i>	1	
<i>coupe</i>		1

3. Then we need to complete the table according to the definition of infomorphism. This is done as follows: Columns have to correspond to columns of the local ontology's classification table. The only possible completion is:

	building vehicle car	
<i>cabrio</i>	0	1
<i>my-home</i>	1	0
<i>4wd</i>	0	1
<i>new-inn</i>	1	0
<i>old-inn</i>	1	0
<i>coupe</i>	0	1

Hence,

$$\begin{aligned} f^*(\text{building}) &= \text{house} \\ f^*(\text{vehicle}) &= \text{automobile} \end{aligned}$$

Rows have to correspond to rows of the reference ontology's classification table. The only possible completion is:

	building vehicle car		
<i>cabrio</i>	0	1	1
<i>my-home</i>	1	0	0
<i>4wd</i>	0	1	1
<i>new-inn</i>	1	0	0
<i>old-inn</i>	1	0	0
<i>coupe</i>	0	1	1

Hence,

$$f^*(\text{car}) = \text{automobile} ,$$

which completes one possible valid ontology mapping.

The steps described above constitute the core part of the IF-Map method. We complement it with heuristic-based techniques to help us initiate the infomorphism generation.

#### 4.5 Initiating the IF-Map Method

Our definition of ontology morphism (Definition 5) enforces an arity-compatibility check to ensure that the local instances are mapped onto appropriate reference types. When automating this step though, we have to be careful for undesired assignments. These arise when the prospective relations to be mapped share the same types but do not have the same semantics. For instance, assume that the reference ontology has relation `hasJobTitle` defined over concepts `employee` and `string` and the local ontology has relation `authoredBy` defined over `string` and `employee`.<sup>9</sup> The infomorphism generation will map the reference concept `employee` to the local concept `string` and the reference concept `string` to the local concept `employee`, which will inevitably map the `hasJobTitle` relation to `authoredBy` by virtue of sharing the same types.

To tackle this problem we are thinking of two possible ways: (a) We provide a partial map of concepts from one ontology to concepts of the other or (b) classify some instances from the local ontology to their counterparts in the reference ontology. This way we can say that the reference concept `employee` maps onto the local concept `employee` and the reference concept `string` maps onto the local concept `string` and only this mapping between these types is possible. This will constrain the infomorphism generation and the undesired infomorphisms will not appear. To do this partial mapping automatically we employ a set of heuristics (originally described in [20], pp.95–97, and enhanced for IF-Map). In particular, these heuristics are working on a purely syntactic match fashion but they use the *is-a* hierarchy and type checking to find types that are shared by relations in both ontologies. The algorithm goes like that:

1. Find relation names from both ontologies that are syntactically equivalent (i.e., `publishedBy` from the reference ontology matches `publishedBy` from the local ontology);
2. check if their argument types match (since we are dealing with binary relations, both argument types have to match, for instance `employee` for the reference and ontologies; `paper` for the reference and local ontologies);
3. use these types to fix a partial map to start the infomorphism generation;
4. if step 2 fails, then traverse the *is-a* hierarchy of types and find syntactically common types that subsume or are subsumed by the common relations' argument types (we traverse the *is-a* hierarchy in both directions: We check for parent and child nodes of the starting node);
5. those that are found syntactically equivalent will be used as in step 3 for partially fixing the initial map of the two ontologies;
6. if step 2 yields only one argument type match, use it and do step 4 for the other argument type.

Note that this algorithm relies on the existence of common relation names in both ontologies. These are syntactically invariant type names. We assume

---

<sup>9</sup> Note that here the reference and local ontologies are not the same ontologies used in the mapping example.

that since the role of reference ontologies within a community is to favour the sharing of knowledge expressed by means of different local ontologies, many of the names of concepts and relations used to express the reference ontology are syntactically equivalent to the ones used in the local ontologies to express the same (or similar) concepts and relations. If they are not, in which case we have syntactically variant type names, we use another algorithm which makes use of similar heuristics but the main difference is that step 2 above is now the first step and we don't employ step 1 at all. This will allow us to find syntactically variant type names that hold over the same argument types (or their dependants after the taxonomy of types has been traversed). However, this algorithm can yield irrelevant results as the types might be semantically different and not be a simple syntactic variation of the same concept.

In this case, the algorithms described cannot initiate IF-Map and then we turn to the second solution proposed above, which is to let the knowledge engineer classify instances manually. This solution though, requires familiarisation of the engineer with both the reference and local ontologies.

## 5 Application of IF-Map

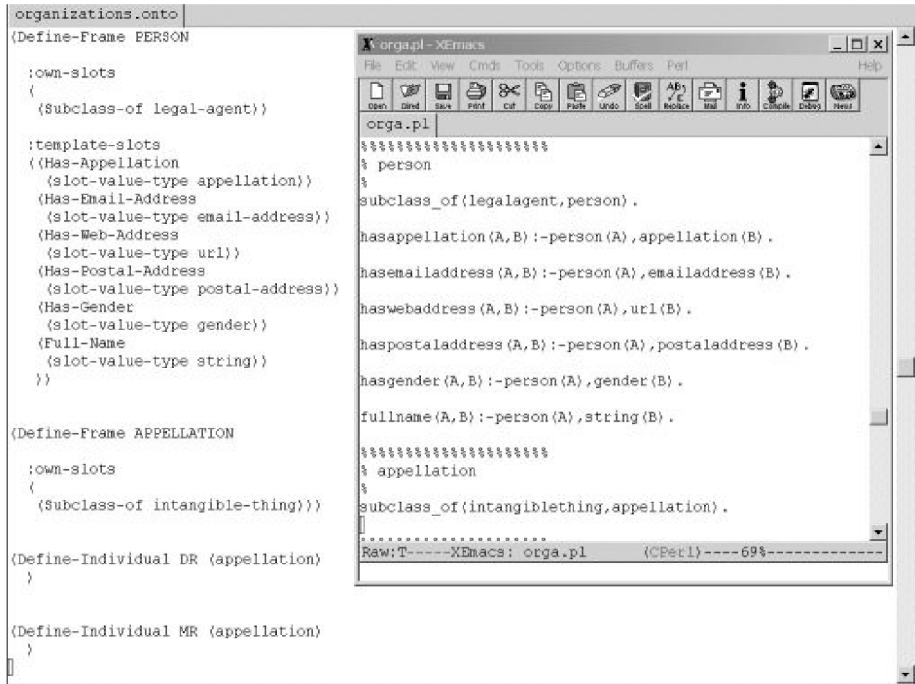
We applied the algorithmic steps described in the illustrative example of Section 4 in a large-scale experiment that we conducted in the context of the AKT project (<http://www.aktors.org>). We have not finished our experiments yet, but we have done enough to assess IF-Map. The setting of the experiment is as follows: In the AKT project, five participating universities are contributing their own ontologies representing their own important concepts in the domain of computer science departments in the UK. There is also a reference ontology, AKT REFERENCE, which was built collaboratively by interested participants from all five sites. So, we had to deal with five local ontologies and one reference ontology. The local ontologies were populated whereas the reference ontology was not. That is in-line with the IF-Map scenario we described in Section 4. Furthermore, since local ontologies are maintained locally by each participating site, it is anticipated to face a variety of formalisms and use different tools for ontology design, development, and deployment. IF-Map's architecture (see Section 2) however, allows for different formalisms to be used as input.

One of our case studies was to apply IF-Map to map AKT REFERENCE to Southampton's and Edinburgh's local ontologies, SOTON and EDIN. These local ontologies are populated with a few thousand instances (ranging from 5k to 18k) and a few hundreds of concepts. There are a few axioms defined, and both have relations. AKT REFERENCE is more compact, it has no instances and approximately 65 concepts with 45 relations. There are a few axioms defined as well. In Figure 6 we include two screenshots, one from Ontolingua-encoded AKT REFERENCE—in particular, the organisations sub-ontology—and the translated version of this fragment in Prolog clauses.<sup>10</sup> As we mentioned earlier (Section 2),

<sup>10</sup> The reader should bear in mind that clauses in the Prolog version denote the typing of the relations and not logical implications.



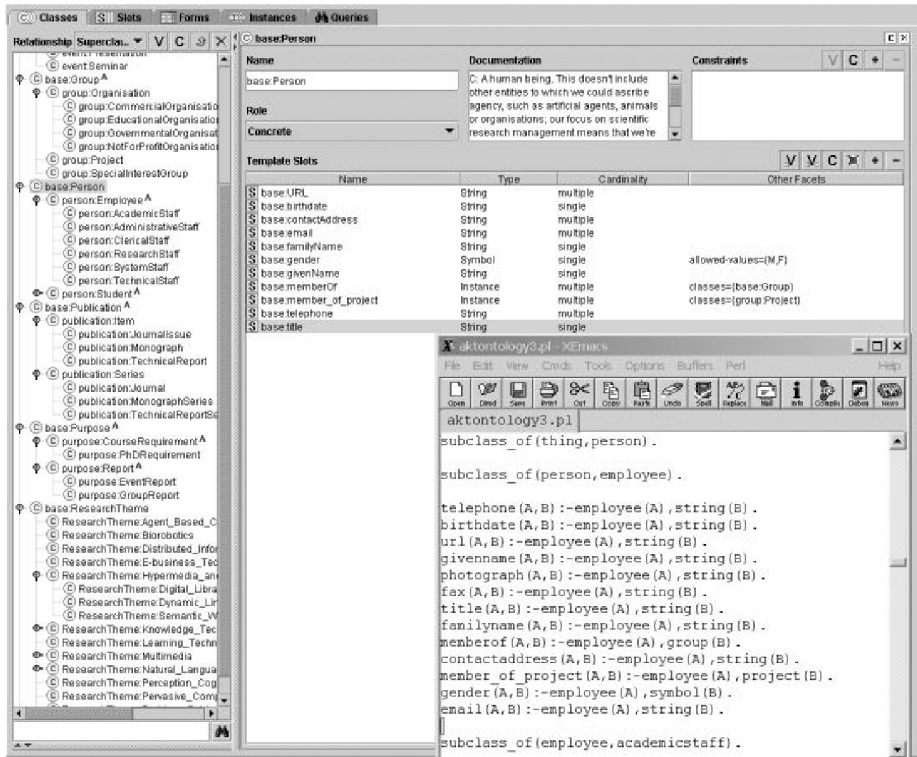
our translation is a partial and customised for the IF-Map application. Therefore, from the Ontolingua fragment shown in Figure 6 we only extract and transform to Prolog clauses the class hierarchy (by regarding frames as classes) and the relations along with their arities (which is used for type checking to bind appropriate tokens to types when using Prolog's backward chaining inference engine).



**Fig. 6.** A window with an Ontolingua fragment of AKT Reference overlapped by a smaller window showing a partial translation of this in Prolog clauses.

In Figure 7 we include two screenshots, one from the Protégé-edited SOTON—in particular, highlighting the class person and its template slots—and the translated version of this fragment in Prolog clauses. As with the reference ontology translation, we partially translate predefined fragments of the local ontology like class hierarchy and relations along with arities.

In Figure 8 we include a screenshot of our Web accessible RDF results page for some relations and concepts. In this page, we show a small fraction of the results from mapping concepts and relations from AKT REFERENCE to their counterparts in SOTON. The concepts shown can be traced back to the previous figures where the original format is shown. As we can see, apart from mapping concepts, like AKT REFERENCE's document to SOTON's publication we also map



**Fig. 7.** A window with a Protégé fragment of Soton overlapped by a smaller window showing a partial translation of this in Prolog clauses.

relations: AKT REFERENCE's *hasappellation* to SOTON's *title*. The arities of these relations allow this sort of mapping, whereas in other ontologies this would have been inappropriate, when for example *title* refers to title of a paper. These mappings were generated automatically, and IF-Map initiated these with the semantically-rich heuristics we described in 4.5.

The algorithms we have implemented so far are of exponential complexity in the number of concepts, because they are based on a declarative Prolog specification of the mapping principle explained in in Section 4. Consequently, we currently base the IF-Map method on an incremental construction of ontology morphisms, in order to tackle large-scale ontologies: First, only certain manageable fragments of the ontologies are mapped, and next, these fixed maps are used to guide the generation of larger fragments, in the manner explained in Section 4. We are currently investigating heuristics for the automatic identification of such fragments.



Fig. 8. Results of ontology mapping in Web accessible RDF format.

## 6 Related Work

IF-Map, amid its well-defined purpose of ontology mapping and, extensionally, merging, taps on a number of areas and uses techniques discussed in diverse communities. Therefore, it is impossible to compile an exhaustive list of references to related work, but we have deliberately expanded the scope of references to cover as many representative works as possible; for a more in-depth survey on

ontology mapping, see [21]. At the same time though, we were careful to identify works that are related somehow with IF-Map's core characteristics: Use of formal definitions of ontology mapping, use of information-flow theory, expressed in a declarative and executable language in a domain and tool independent manner, applied as a method and as a theory for ontology mapping, and being—under circumstances—fully automatic.

Not all of the references we cite here meet these criteria; some provide features that IF-Map does not support and others focus on a single criterion of the list given above. Nevertheless, the diversity of works reported in this section demonstrates the importance of the topic in a number of communities. This paper's scope, though, prevents us from getting into great detail when describing related work hereinafter, but we give a flavour of the current landscape in ontology mapping research across different communities. Among the few formal approaches in ontology mapping and merging is that of FCA-Merge [41]. It is based on Formal Concept Analysis [14] and it is aimed, mainly, at merging ontologies. Its developers, Stumme and Maedche, incorporate natural language techniques in their FCA-based method to derive a lattice of concepts. The lattice is then explored manually by a knowledge engineer who will build the merged ontology with semi-automatic guidance from FCA-Merge. In particular, FCA-Merge works as follows: The input to the method are a set of documents—from which concepts will be extracted—together with the ontologies that will be merged. These documents should be representative of the domain at question and be related to the ontologies. They also have to cover all concepts from both ontologies as well as separating them well enough. These strong assumptions have to be met in order to obtain good results from the FCA-Merge. As this method relies heavily on the availability of classified instances in ontologies to be merged, the authors argue that this will not be the case in most ontologies so they opt to extract instances from documents. In this respect, the first step of FCA-Merge could be viewed as an ontology population mechanism. This initial step in FCA-Merge could be skipped though, when there is a shared set of instances classified to the concepts in both ontologies. Once the instances will be extracted,<sup>11</sup> Stumme and Maedche construct the concept lattice and from there provide semi-automatic support for the knowledge engineer to derive the final merged ontology.

Formal Concept Analysis has also been used by the database community in their federated databases domain. In particular, Schmitt and Saake employ Formal Concept Analysis techniques to assist database schema integration [37]. The focus of their work is to merge different inheritance hierarchies by decomposing overlapping class extensions into base extensions and use Formal Concept Analysis techniques to inform algorithms for integrating the databases schemata. In the Scalable Knowledge Composition (SKC) project, Jannink et al. [19] presented the use of a rule-based algebra for encapsulating and composing ontologies. On-

---

<sup>11</sup> We do not refer to natural language techniques and methods used in this process by FCA-Merge developers, since they are peripheral to our interests in ontology mapping.

tologies are clustered in contexts, and the authors use a rule-based algebra to define interfaces to link the extracted contexts with the original ontologies.

Noy and Musen have developed two systems for performing ontology merging and alignment in the Protégé-2000 ontology development environment [16]: SMART [33] and its successor PROMPT [34]. These tools use linguistic similarity matches between concepts for initiating the merging or alignment process and then use the underlying ontological structures provided by the Protégé-2000 environment (classes, slots, facets) to inform a set of heuristics for identifying further matches between the ontologies. A similar tool has been developed by McGuinness et al. for the Ontolingua ontology editor: Chimaera [29]. As in PROMPT, this tool is interactive and the engineer is in charge of making decisions that will affect the merging.

From the machine learning perspective we report the works of Lacher and Groh [26] and Doan et al. [9] where their systems employ machine learning algorithms in conjunction with similarity measures to yield prospective mappings between ontology concepts. Other works worth citing here are Chalupsky's OntoMorph [6] translation system for symbolic knowledge, Kiryakov et al.'s OntoMap portal [25] for mapping linguistic ontologies, the OBSERVER system [30] by Mena et al. for information integration, Gangemi et al.'s [13] ONIONS methodology for medical ontologies, Visser and Tamma's heterogeneity categorisation [42], and the reports from Pinto et al. [35] and Noy and Hafner in [32].

## 7 Conclusion

In this paper we have presented a novel method and a theory for ontology mapping. We formalised the notion of ontology, ontology morphism and ontology mapping and linked them to the formal notions of local logic and logic informorphism stemming from IF theory. We then applied them in a mechanised manner—IF-Map—to map diverse ontologies. The first results are promising for the application of IF-Map to large-scale ontology mapping efforts and we continue researching fruitful extensions of it, such as ontology merging, reasoning about ontology evolution, and inclusion of ontological axioms.

The automation of ontology mapping as provided by IF-Map provides automated support in the alignment of ontologies by automatically generating mappings between a reference and various local ontologies. For instance, for the special case of data integration, this would correspond to automatic generation of source descriptions, e.g. formalised as views over a mediated schema [18]. The correspondence, though, is not exact, as views over a mediated schema could be more expressive than IF-Maps current concept-to-concept and relation-to-relation mapping.

In this sense it would be interesting to work on the integration of previous approaches to semantic integration by the database community with recent efforts carried out by the ontology community (see, e.g., [21]) in the context of

our approach to ontology mapping based on the Barwise-Seligman theory of information flow, as it might further illuminate the long road lying ahead.

**Acknowledgements.** An earlier version of this paper was presented under the title “Information-Flow based Ontology Mapping” at the First International Conference on Ontologies, Databases and Applications of Semantics (ODBASE’02), Irvine CA, USA, October 2002. We are grateful to Karl Aberer for selecting an extended version of that paper for this special volume, and we would like to thank the audience of ODBASE’02 and the reviewers for their valuable comments.

This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

Marco Schorlemmer is also supported by a ‘Ramón y Cajal’ Fellowship from the Spanish Ministry of Science and Technology.

## References

1. M. Barr. The Chu construction. *Theory and Applications of Categories*, 2(2):17–35, 1996.
2. J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983.
3. J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.
4. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
6. H. Chalupsky. OntoMorph: A translation system for symbolic knowledge. In *7th International Conference on Principles of Knowledge Representation and Reasoning*, Breckenridge, Colorado, USA, Apr. 2000.
7. F. Corrêa da Silva, W. Vasconcelos, D. Robertson, V. Brilhante, A. de Melo, M. Finger, and J. Agustí. On the insufficiency of ontologies: Problems in knowledge sharing and alternative solutions. *Knowledge-Based Systems*, 15(3):147–167, 2002.
8. K. Devlin. *Logic and Information*. Cambridge University Press, 1991.
9. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *11th International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.
10. J. Domingue. Tadzebao and WebOnto: Discussing, browsing, and editing ontologies on the web. In *11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, Apr. 1998.
11. F. Dretske. *Knowledge and the Flow of Information*. MIT Press, 1981.
12. A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727, 1997.

13. A. Gangemi. Ontology integration: Experiences with medical terminologies. In N. Guarino, editor, *Formal Ontology in Information Systems*, volume 46 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1998.
14. B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, 1999.
15. M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format (KIF). Draft proposed American National Standard, NCITS.T2/98-004, 1998.
16. W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu, and M. Musen. Knowledge modeling at the millennium (the design and evolution of Protégé-2000). In *12th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, Oct. 1999.
17. V. Gupta. *Chu Spaces: A Model of Concurrency*. PhD thesis, Stanford University, 1994.
18. A. Halevy. Answering queries using views. *The VLDB Journal*, 10:270–294, 2001.
19. J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. Encapsulation and composition of ontologies. In *AAAI'98 Workshop on Information Integration*, Madison, Wisconsin, USA, July 1998.
20. Y. Kalfoglou. *Deploying Ontologies in Software Design*. PhD thesis, Division of Informatics, The University of Edinburgh, June 2000.
21. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The state of the art. *Knowledge Engineering Review*, 18(2), 2003.
22. R. Kent. The information flow foundation for conceptual knowledge organization. In *6th International Conference of the International Society for Knowledge Organization*, Toronto, Canada, July 2000.
23. R. Kent. A KIF formalization of the IFF category theory ontology. In *IJCAI 2001 Workshop of the IEEE Standard Upper Ontology*, Seattle, Washington, USA, 2001.
24. R. Kent. The IFF foundation for ontological knowledge organization. In *Knowledge Organization and Classification in International Information Retrieval, Cataloging and Classification Quarterly*. The Haworth Press Inc., 2003.
25. A. Kiryakov, K. Simov, and M. Dimitrov. OntoMap: Portal for upper-level ontologies. In *Second International Conference on Formal Ontology in Information Systems*, Ogunquit, Maine, USA, Oct. 2001.
26. M. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *14th International FLAIRS Conference*, Key West, Florida, USA, May 2001.
27. O. Lassila and R. Swick. Resource description framework (RDF) model and syntax specification. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>, Feb. 1999. W3C Recommendation.
28. B. McBride. Jena: Implementing the RDF model and syntax specification. In *2nd International Workshop on the Semantic Web (SemWeb'2001)*, volume 40 of *CEUR Workshop Proceedings*, Hong Kong, China, May 2001.
29. D. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *7th International Conference on Principles of Knowledge Representation and Reasoning*, Breckenridge, Colorado, USA, Apr. 2000.
30. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Domain specific ontologies for semantic information brokering on the global information infrastructure. In N. Guarino, editor, *Formal Ontology in Information Systems*, volume 46 of *Frontiers in Artificial Intelligence and Applications*, pages 269–283. IOS Press, 1998.
31. E. Motta. *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*, volume 53 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1999.

32. N. Noy and C. Hafner. The state of the art in ontology design: A survey and comparative review. *AI Magazine*, 18(3):53–74, 1997.
33. N. Noy and M. Musen. SMART: Automated support for ontology merging and alignment. In *12th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, Oct. 1999.
34. N. Noy and M. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *17th National Conference on Artificial Intelligence (AAAI'00)*, Austin, Texas, USA, July 2000.
35. S. Pinto, A. Gómez-Pérez, and J. Martins. Some issues on ontology integration. In *IJCAI'99 Workshop on Ontologies and Problem-Solving Methods*, volume 18 of *CEUR Workshop Proceedings*, Stockholm, Sweden, Aug. 1999.
36. V. Pratt. The Stone gamut: A coordination of mathematics. In *10th Annual Symposium on Logic in Computer Science*, pages 444–454. IEEE Computer Society Press, 1995.
37. I. Schmitt and G. Saake. Merging inheritance hierarchies for database integration. In *3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98)*. IEEE Computer Society Press, 1998.
38. M. Schorlemmer. Duality in knowledge sharing. In *7th International Symposium on Artificial Intelligence and Mathematics*, Ft. Lauderdale, Florida, USA, 2002.
39. A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
40. J. Sowa. *Knowledge Representation and Reasoning: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.
41. G. Stumme and A. Maedche. FCA-Merge: Bottom-up merging of ontologies. In *17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, Seattle, Washington, USA, Aug. 2001.
42. P. Visser and V. Tamma. An experience with ontology-based agent clustering. In *IJCAI'99 Workshop on Ontologies and Problem-Solving Methods*, volume 18 of *CEUR Workshop Proceedings*, Stockholm, Sweden, Aug. 1999.



# OntoEdit: Multifaceted Inferencing for Ontology Engineering

York Sure<sup>1</sup>, Juergen Angele<sup>2</sup>, and Steffen Staab<sup>1,2</sup>

<sup>1</sup> Institute AIFB, University of Karlsruhe,  
76128 Karlsruhe, Germany  
{sure,staab}@aifb.uni-karlsruhe.de  
<http://www.aifb.uni-karlsruhe.de/WBS/>

<sup>2</sup> Ontoprise GmbH,  
Haid-und-Neu-Str. 7,  
76131 Karlsruhe, Germany,  
angele@ontoprise.de  
<http://www.ontoprise.de/>

**Abstract.** Ontologies now play an important role for many knowledge-intensive applications for which they provide a source of precisely defined terms. The terms are used for concise communication across people and applications. Tools such as ontology editors facilitate the creation and maintenance of ontologies. OntoEdit is an ontology editor that has been developed keeping five main objectives in mind: 1. Ease of use. 2. Methodology-guided development of ontologies. 3. Ontology development with help of inferencing. 4. Development of ontology axioms. 5. Extensibility through plug-in structure.<sup>1</sup>

## 1 Introduction

Ontologies now play an important role for many knowledge-intensive applications for which they provide a source of formally defined terms. They aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused, shared, and operationalized across applications and groups. However, because of their size, their complexity and their formal underpinnings ontologies are still far from being a commodity.

This urgent need motivated researchers in recent years to support the ontology engineering process. Mainly, we have seen three directions. Firstly, several seminal proposals for guiding and supporting the ontology development process have been proposed [UK95,GP96,WG01]. Secondly, a considerable number of tools that support the ontology engineering process [DSW<sup>+</sup>00,NFM00,ACFLGP01,Dom98] have been developed. Third, inferencing mechanisms for large ontologies have been developed and implemented (cf., e.g., [Hor98]). However, only few of these seminal works have worked towards integrating these aspects.

OntoEdit is an ontology editor that is rather unique in its kind as it is based on a recent methodology for ontology development (cf. [Sur03]) and as it makes comprehensive

---

<sup>1</sup> This paper is based on the previously published [SSA02].

use of inferencing. In particular, OntoEdit focuses on three main steps for ontology development (as also described in [SSSS01]), viz. requirements specification, refinement and evaluation<sup>2</sup>.

First, all requirements of the envisaged ontology are collected. Typically an ontology engineer captures domain and goal of the ontology, design guidelines, available knowledge sources (e.g. reusable ontologies and thesauri etc.), potential users and use cases and applications supported by the ontology. Output of this phase is a semi-formal description of the ontology. Second, during the refinement phase the semi-formal description is extended and completely formalized into an appropriate representation language. The language is chosen according to the requirements for the ontology, e.g. identified applications that will be supported by the ontology. Output of this phase is a mature ontology (aka. “target ontology”). Third, the target ontology needs to be evaluated according to the requirement specifications and formal evaluation criteria (e.g. as proposed in the OntoClean methodology, cf. [WG01]). Typically this phase serves as a proof for the usefulness of developed ontologies.

Support for these development steps is a crucial objective that must be merged with the conflicting needs for ease of use and the construction of complex ontology structures.

In the following, we will first present a brief, real-life case study on configuration management that motivates our examples given thereafter. Section 3 explains theoretical and practical issues of the inference engine that constitutes the internal knowledge model of OntoEdit. The Sections 4 to 6 correspond to the core steps of the ontology development methodology sketched above. In particular, they elucidate how inferencing is used for supporting methodology-based ontology construction and evaluation and how this support is clad into a user-friendly environment.

## 2 Case Study: Configuration Management

We have developed an ontology based configuration tool OnKo for the German Telecom. It represents a set of IT components together with their complex interrelationships. It supports an interactive configuration process of such components to IT systems and IT landscapes. It contains intelligent search and retrieval of already existing IT systems with similar functionality and thus integrates already existing experience of the company.

The ontology is exploited for navigation purposes. The user can switch between different views, i.e. *specialization view*, *part-of view* etc. to the available components and thus the user is able to choose the best presentation for the current configuration task. These views are represented as hierarchical trees and additional links between the nodes enabling a simple interactive search within the available IT components. By this way the user subsequently searches for components and adds them to the current configuration. In each step the interrelationships of the components of the current configuration are checked for consistency and feed back is delivered. Thus upcoming dead ends of configurations which would not work are recognized early. The system automatically derives new knowledge and makes further suggestions about possible extensions of the current configuration which therefore must not be specified by the user. This “mixed initiative

---

<sup>2</sup> In another companion paper [SEA<sup>+</sup>02], we have described how OntoEdit supports collaborative ontology engineering.

intelligence” strategy enables the development of a configuration in close cooperation with the user and thus supports the user without telling him what to do.

In Figure 1 a screenshot of OnKo is shown. In the left frame the user navigates within an is-a hierarchy of the components. This view may be switched to a part-of hierarchy. The attribute values of a selected component are editable in a form in the middle frame. The selected component together with its attribute values are used to derive attribute values of dependent components, i.e. the frequency of the processor is propagated upwards in the part-of hierarchy to constitute the frequency of the entire computer. The current configuration with all its components is shown in the right frame. If a configuration contains inconsistent components which is checked by applying consistency rules, appropriate error and warning messages are immediately given and alternatives for a selected component are presented to the user which make the configuration consistent. In each step the current configuration may be compared to similar existing configurations in the company. This makes existing experience transparent to the user compiling a new configuration.

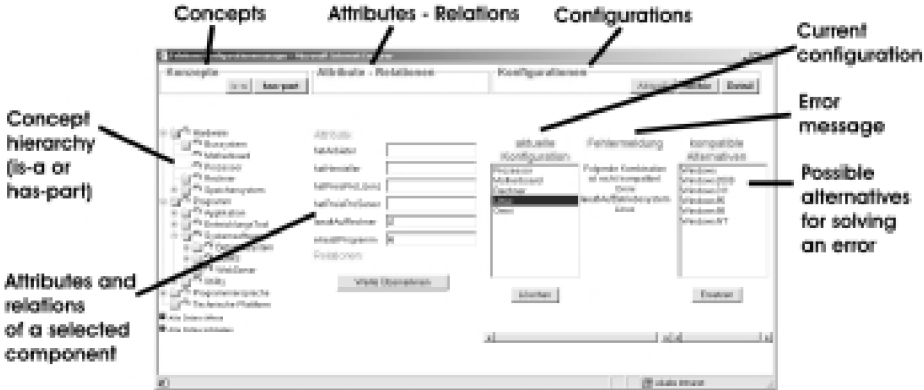


Fig. 1. OnKo an interactive configuration tool for the German Telecom

In our screenshot the current configuration contains two incompatible components: “Omni”, a web server software, does not work on “Linux”. To solve this incompatibility, the system suggests to use “Windows” or “Windows2000” or ... etc. instead of “Linux”. A user may now replace incompatible components to get a valid configuration for a computer system.

In the following, we will show some examples experienced in this case study.

### 3 Inferencing – Theoretical and Practical Issues

#### 3.1 Logical Framework

In order to provide a clearly defined semantics to the knowledge model of OntoEdit, the knowledge structures of OntoEdit correspond to a well-understood logical framework,

viz. F-Logic (cf. [KLW95], “F” stands for “Frames”). F-Logic combines deductive and object-oriented aspects: “*F-logic [...] is a deductive, object-oriented database language which combines the declarative semantics of deductive databases with the rich data modelling capabilities supported by the object oriented data model.*” (cf. [FHKS96])

F-Logic allows for concise definitions with object oriented-like primitives (classes, attributes, object-oriented-style relations, instances) that are reflected by the OntoEdit GUI. Furthermore, it also has Predicate Logic (PL-1) like primitives (predicates, function symbols), that are only partially reflected in the GUI but internally used within the data structures. F-Logic allows for axioms that further constrain the interpretation of the model. Axioms may either be used to describe constraints or they may define rules, e.g. in order to define a relation  $R$  by the composition of two other relations  $S$  and  $Q$ .

F-Logic rules have the expressive power of Horn-Logic with negation and may be transformed into Horn-Logic rules. The semantics for a set of F-Logic statements is defined by the well-founded semantics (cf. [vRS91]). This semantics is close to First-Order semantics. In contrast to First-Order semantics not all possible models are considered but one “most obvious” model is selected as the semantics of a set of rules and facts. It is a three valued logic, i.e. the model consists of a set of true facts and a set of unknown facts and a set of facts known to be false.

In comparison to other logic-based representation languages for ontologies, F-Logic is quite well suited for the usage within the World Wide Web (cf. [Dec02]). Unlike Description Logics (DL), F-Logic does not provide means for subsumption [Hor98], but (also unlike DL) it provides for efficient reasoning with instances and for the capability to express arbitrary powerful rules, e.g. ones that quantify over the set of classes.

The most widely published operational semantics for F-Logic is the alternating fixed point procedure. This is a forward chaining method which computes the entire model for the set of rules, i.e. the set of true and unknown facts. For answering a query the entire model must be computed (if possible) and the variable substitutions for the query are then derived. In contrast, the inference engine Ontobroker performs a mixture of forward and backward chaining based on the dynamic filtering algorithm (cf. [KL86]) to compute (the smallest possible) subset of the model for answering the query. In most cases this is much more efficient than the simple evaluation strategy. These techniques stem from the deductive data base community and are optimized to deliver all answers instead of one single answer as e.g. resolution does.

Within the F-Logic compiler F-Logic statements are translated to normal programs. Normal programs are Horn programs where rules may contain negated literals in their bodies. Horn Logic is Turing Complete, thus F-Logic programs are not decidable in principle. The semantics defined for these normal programs is the well-founded semantics [van93]. In contrast to the stratified semantics the well-founded semantics is also applicable for rules which depend on cycles containing negative rule bodies. Because F-Logic is very flexible, during the translation to normal programs such negative cycles often arise. In [vRS91] the alternating fixpoint has been described as a method to operationalize such logic programs. This method has been shown to be very inefficient. Therefore the inference engine realizes dynamic filtering [KL86] which combines top-down and bottom-up inferencing. Together with an appropriate extension to compute

the well-founded semantics this method has been proven to be very efficient compared to other horn based inference engines (cf., e.g., [SSA02]).

For detailed introductions to the syntax and the object model of F-Logic, in particular with respect to the implementation of F-Logic in Ontobroker, we refer to [Erd01], [Dec02] and [Ont02].

### 3.2 Inference Engine Ontobroker

The inference engine Ontobroker (cf. [DEFS99]) was developed in numerous years of scientific research (cf., e.g., [Ang93, Fen95, Erd01, Dec02]) and is now being commercialized by the company Ontoprise<sup>3</sup>. It comes with several features that makes it adequate not only for inferencing purposes but rather as a backbone for an ontology editor. In particular, it provides:

- A namespace mechanism: Thus, several ontologies (or ontology parts) may be syntactically split into modules and processed by different inference engines.
- Switch-off: It is possible to switch of (possibly singleton) sets of definitions. Thus, one may test interactions and easily distinguish between modules.
- Rule names: Each rule can be identified and accessed via a unique name.
- Database Connectors: Thus, one may easily map db tables into predicates via, e.g., JDBC.
- User-definable Built-ins: Besides of standard built-ins like “multiply”, the user may define his own ones for special purposes.
- An extensive API: Thus, one may remotely connect to the inference engine and one may also import and export several standards (e.g., RDF(S)).

## 4 Requirements Specification

Like in software engineering and as proposed by [GP96], we start ontology development with collecting requirements for the envisaged ontology. Typically this task is performed by a team of experts for the domain accompanied by experts for modeling. The outcome of this phase is (i) a document that contains all relevant requirement specifications (domain and goal of the ontology, design guidelines, available knowledge sources, potential users and use cases and applications supported by the ontology) (ii) a semi-formal ontology description, i.e. a graph of named nodes and (un-)named, (un-)directed edges, both of which may be linked with further descriptive text.

To operationalize a methodology it is desirable to have a tool that reflects and supports all steps of the methodology and guides users step by step through the ontology engineering process. Along with the development of the methodology we therefore extended the core functionalities of OntoEdit by two plug-ins to support first stages of the ontology development, viz. OntoKick and Mind2Onto<sup>4</sup>.

<sup>3</sup> Ontoprise GmbH, see <http://www.ontoprise.de/>

<sup>4</sup> Describing the plug-in framework is beyond the scope of this paper, it is described in [Han01]. The basic idea is to expand OntoEdit’s functionalities through plug-ins.

OntoKick targets at (i) creation of the requirement specification document and (ii) extraction of relevant structures for the building of the semi-formal ontology description. Mind2Onto targets at integration of brainstorming processes to build relevant structures of the semi-formal ontology description. As computer science researchers we were familiar with software development and preferred to start with a requirement specification of the ontology, i.e. OntoKick. People who are not so familiar with software design principles often prefer to start with “doing something”. Brain storming is a good method to quickly and intuitively start a project, therefore one also might begin the ontology development process with Mind2Onto.

## 4.1 OntoKick Plugin

OntoKick is an OntoEdit plug-in that extends the functionality of OntoEdit by support for requirements specification. OntoKick allows for describing important aspects of the ontology, viz.: the domain and the goal of the ontology, the design guidelines, the available knowledge sources (e.g. domain experts, reusable ontologies etc.), the potential users, the use cases, and the applications supported by the ontology. OntoKick stores these descriptions with the ontology definitions.

As proposed by [UK95], we use competency questions (CQ) to define requirements for an ontology. Each CQ defines a query that the ontology should be able to answer and therefore defines explicit requirements for the ontology. Typically, CQs are derived from interviews with domain experts and help to structure knowledge. We take further advantage of using them to create an initial version of the semi-formal description of the ontology. Based on the assumption that each CQ contains valuable information about the domain of the ontology we extract relevant concepts and relations (see example below). Furthermore, OntoKick establishes and maintains links between CQs and concepts derived from them. This allows for better traceability of the origins of concept definitions in later stages.

We illustrate the usage of CQs by an example from our case study. Figure 2 shows a screenshot of our ontology environment OntoEdit presenting the configuration management ontology. In the left column, one may recognize an excerpt of the *is-a* hierarchy of concepts from our case study ontology. One concept is selected, i.e. highlighted, (“Bus systems”, i.e. bus systems of microprocessors), in the right column all relations and attributes for the selected concept are shown. A context menu for the selected concept offers possibilities for typical modification (e.g. id, external representations and documentations in multiple languages etc.) including the feature to show corresponding competency questions.

The methodology is supported as follows. First, the ontology engineer has interviewed an expert in configuration management. Thereby they have identified CQs, e.g. “With which bus systems are AS400 equipped?”. Based on these CQs the ontology engineer has created the semi-formal description of the ontology. He has identified relevant concepts and relations from the above-mentioned CQ, e.g. “bus systems”. After capturing CQs and modeling the ontology with OntoEdit the ontology engineer has been able to retrieve the corresponding CQ for each concept and relation, helping him to identify the context in which they were modelled.

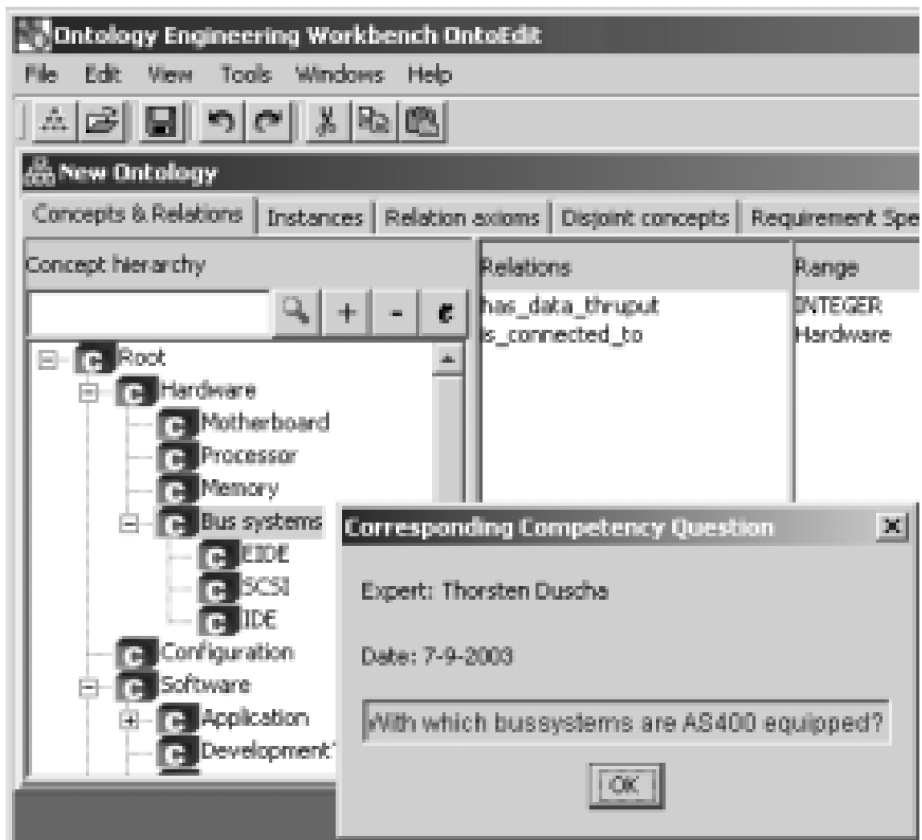


Fig. 2. A Competency Question (CQ) in the Ontology Engineering Environment OntoEdit

4.2 Mind2Onto Plugin

Mind2Onto is a plug-in for supporting brainstorming and discussion about ontology structures. Especially during early stages of projects, brainstorming methods are commonly used to quickly capture pieces of relevant knowledge. A widely used method are mind maps<sup>TM</sup> [Buz74], they were originally developed to support more efficient learning and evolved to a management technique used by numerous companies. A mind map<sup>TM</sup> provides information about a topic that is structured in a tree. Each branch of the tree is typically named and associatively refined by it's subbranches. Icons and pictures as well as different colors and fonts might be used for illustration purposes. The assumption is that our memory performance is improved by these visual aspects. There already exist numerous tools for the creation of digital mind maps<sup>TM</sup>. Many people from academia and industry are familiar with mind maps<sup>TM</sup> and related tools – including potential ontology engineers and domain experts. Therefore the integration of digital mind maps<sup>TM</sup> into the ontology development process is very attractive (cf., e.g., [LS02]).

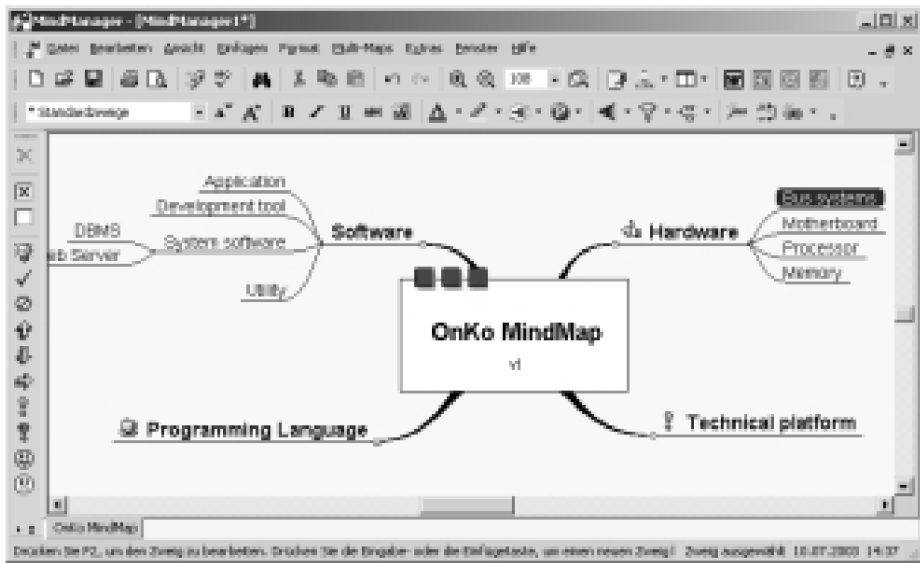


Fig. 3. Mind Map for OnKo

We relied on a widely used commercial tool<sup>5</sup> for the creation of mind maps<sup>TM</sup>. It has advanced facilities for graphical presentations of hierarchical structures, e.g. easy to use copy&paste functionalities and different highlighting mechanisms. It's strength but also it's weakness lies in the intuitive user interface and the simple but effective usability, which allows for quick creation of mind maps<sup>TM</sup> but lacks of expressiveness for advanced ontology modeling. By nature, mind maps<sup>TM</sup> have (almost) no assumptions for it's semantics, i.e. branches are somehow "associatively related" to each other. This assumption fits perfectly well during early stages of ontology development for quick and effective capturing of relevant knowledge pieces and makes the mind map<sup>TM</sup> tool a valuable add-on. Figure 3 shows a draft mind map<sup>TM</sup> for the OnKo system.

Mind2Onto integrates mind maps<sup>TM</sup> into the ontology engineering process. Currently OntoEdit and the mind map<sup>TM</sup> tool interoperate through import and export facilities based on XML.

## 5 Ontology Refinement

In the ontology refinement phase the semi-formal description of the ontology is extended and the ontology is completely formalized in order to make it machine understandable and processable. In this phase we take advantage of the inferencing capabilities of OntoEdit for several purposes.

<sup>5</sup> MindManager<sup>TM</sup> 2002 Business Edition, cf. <http://www.mindjet.com>



## 5.1 Reuse by Semantic Integration

Reuse in the refinement phase comes in two flavors. First, we may exploit existing thesauri, e.g. ones that are stored in databases by semantically integrating them into the inferencing model. As in general information integration [WG97], this involves two steps. The first step concerns the mapping onto a common data model. For this purpose, we take advantage of the inference engine's capabilities, viz. to read in RDF(S) [BG02], to connect to relational databases, to provide further built-ins (e.g. for connection to XML repositories). The outcome of this step are data in F-Logic structures, but with some rather arbitrary semantics. In the second step, we build rules to map the outcome of the first step into the desired categories. For instance, we may map a relational database table-like structure into a target structure of sub- and superconcepts. An example may be given with a database table that contains WordNet hyponyms and synonyms. Our example is based on a locally installed mySQL database system that contains a 'wordnet-db' database.

1. In the first step we map this table into an equivalent predicate `HYPONYM`:

$$\text{FORALL } C, D \text{ HYPONYM}(C, D) \leftarrow \\ \text{DBACCESS}(\text{hyponym}, F(\text{'sub'}, C, \text{'super'}, D), \text{'mySQL'}, \\ \text{'wordnet-db'}, \text{'localhost'}).$$

2. In the second step, we define the objects of the predicted hyponym to be subconcepts of *Computer* if it is known that one of their superconcepts is a subconcept of *Computer*:

$$\text{FORALL } C, D \text{ } C :: D \leftarrow \text{HYPONYM}(C, D) \text{ AND } D :: \text{Computer}.$$

Taken the two steps together, this means: "Specify every *C* to be a subclass of *D* if in the mySQL database called 'wordnet-db' on my local machine the table called hyponym there is a row with attribute 'sub' being *C* and 'super' being *D* and simultaneously *D* is already known to be a subclass of *Computer*."

Combining such a query for hyponyms with search for synonyms yields about 100 terms, not all of which are obvious and which may be considered for inclusion in the ontology. Thus, one may easily reuse existing thesauri or database schema in order to generate a large number of concepts fast and organize them in a taxonomy.

## 5.2 Reusing Axioms Integrating Them into the Ontology

Secondly, one may reuse axioms definitions from a library of ontology modules that are distinguished by namespace mechanism. A set of axiom definitions specified in one domain is reusable in another domain by the inference engine's capability to store and load axioms from a library to and into different namespaces in a way that is reusable for another domain.

For instance, *partonomic role propagation* may be given for a medical ontology (cf. [HSR99b] for a comprehensive description and appendix A for an illustrating example), but it has been reused for describing properties of computer systems. The underlying

idea of partonomic role propagation is that some properties of parts of a system are propagated to the whole. For instance, the clock frequency of the CPU is frequently used as being descriptive of the overall computer system — while others like frequency rates of the bus system are not used for that abstracting purpose.<sup>6</sup>

We specify an axiom library by a meta-predicate. In this current example, this predicate is named `PARTONOMICROLEPROPAGATION`. This meta-predicate takes four input parameters (cf. the formal specification in Table 1):

1. The relation that is propagated (`FREQUENCYOF`), because not every relation is propagated from a part to a whole.
2. The relation that is propagating (`PHYSICALPARTOF`), because not every relation may be propagated is propagated along all part-whole relations.
3. The whole up to which the relation is maximally propagated (*ComputerSystem*), because propagation may be stopped (e.g. `FREQUENCYOF` should not be propagated to a car that the computer system is a part of).
4. The concept for the instances of which the relation may be propagated (e.g., a (here fictitious) *WheelFrequency* might not be propagated), because not every class is treated the same.

**Table 1.** Partonomic Role Propagation

0	<code>PARTONOMICROLEPROPAGATION(FREQUENCYOF,</code> <code>PHYSICALPARTOF, ComputerSystem, ClockFrequency)</code>
1	$\forall x, y, z, R, S, C, D \ x[R \twoheadrightarrow z] \leftarrow$ <code>PARTONOMICROLEPROPAGATION(R, S, C, D) AND</code> $x : D \text{ AND } x[R \twoheadrightarrow y] \text{ AND } y[S \twoheadrightarrow z] \text{ AND}$ <code>PARTINSTANCEOF(z, C, S).</code>
2	$\forall z, C, S \text{ PARTINSTANCEOF}(z, C, S) \leftarrow$ $\exists E \ z : E \text{ AND PARTOFALONG}(E, C, S).$
3	$\forall C, S \text{ PARTOFALONG}(C, C, S).$
4	$\forall E, C, S \text{ PARTOFALONG}(E, C, S) \leftarrow$ $\exists F \text{ PARTOFALONG}(E, F, S) \text{ AND}$ $\exists Q \ F[Q \twoheadrightarrow C] \text{ AND } Q :: S.$
5	$\forall x, y, S, R \ x[S \twoheadrightarrow y] \leftarrow R :: S \text{ AND } x[R \twoheadrightarrow y].$

### 5.3 Reusing Axioms Applying Them to the Ontology

Besides of integrating axioms from a library into the ontology, one may apply axioms in order to enforce constraints on the ontology. We distinguish three major types:

1. **Axioms of F-Logic:** They are an integral part of the F-Logic definition. However, not all of them are needed for inferencing during the usage of the ontology. For instance, type coercion at the conceptual level:

<sup>6</sup> Similarly, the color of the car body is typically equated with the color of the car. This is not true for the color of the seats, though seats and car body are both parts of the car.

$$\text{FORALL } C, D, E, R, T \ E :: T \leftarrow C[R \Rightarrow T] \text{ AND } D :: C[R \Rightarrow E].$$

“Specify  $E$  as a subclass of  $T$  if some concept  $C$  has a relation  $R$  with range  $T$  and a subclass  $D$  of  $C$  has a relation  $R$  with range  $E$ .”

2. **Axioms for domain-specific consistency:** They enforce consistency constraints at building time. E.g., they may enforce that the relation `HASPHYSICALPART` is without cycles:

$$\text{NONCYCLIC}(\text{HASPHYSICALPART}).$$

$$\text{FORALL } X, R \text{ UNDEFINED } \leftarrow \text{NONCYCLIC}(R) \text{ AND } X[R \Rightarrow X].$$

“`HASPHYSICALPART` is of type `NONCYCLIC`. Indicate consistency violation if a relation  $R$  is of type `NONCYCLIC` and  $X$  is related via  $R$  to itself.”

3. **Axioms enforcing modeling policies:** Such axioms do not add to the semantic description, but they are applied in order to enforce semiotic constraints, e.g. that no subconcept should have more than  $n$  subconcepts, that no hierarchy should be deeper than  $m$ , or that every attribute symbol should begin with a lower case letter (see also Section 6.3):

$$\text{FORALL } R \text{ UNDEFINED } \leftarrow$$

$$\text{EXISTS } X, Y \ X[R \Rightarrow Y] \text{ AND NOT regexp}(\text{“}[a - z]\text{”, } R).$$

“Indicate consistency violation if there exists a relation symbol  $R$  between some classes  $X, Y$  and it does not match with a string beginning with lowercase alphabetical letters.”

The three types of axioms just described are not integrated into the ontology, because once the ontology is fixed and remains unchanged they are not violated anyway. Still, switching them off improves performance, because they need not be revisited and checked again.

## 6 Evaluation

The last step in ontology development is about evaluating the formal ontology.

### 6.1 Analysis of Typical Queries

For this purpose, the Ontology engineer may interactively construct and save instances and axioms into modules. `OntoEdit` contains a simple instance editor that the ontology engineer can use to create test sets. The test set can be automatically processed and checked for consistency. Once, the ontology goes into the evolution phase and needs changes to remain up-to-date, these test sets may be re-used for checking validity of the ontology.

For instance, one may create a test case for partonomic role propagation (test case cf. Table 2, the partonomic role propagation is explained in Appendix A or, more detailed, in [SEM01]), viz. an instance CPU1 of CPU with a clock frequency of 1600MHz (line 1), an instance PC1 of PC which has a MB456 (line 2) that is of type Motherboard and that has the CPU1 on board (line 3).

The test case is completed by the query that asks for all clock frequencies of all PCs. This query is reformulated as a rule (line 4), in order to allow for comparison with an intended set of result tuples (line 5), by a generally applicable rule (lines 6 – 8).

Table 2. Formalizing Test Cases

Test instances
1 CPU1:CPU[hasClockFrequency→‘1600MHz’].
2 PC1:PC[hasPhysicalPart→MB456].
3 MB456:Motherboard[hasPhysicalPart→CPU1].
Query formulated as test query
4 FORALL X,Y test1(X,Y) ← X :CPU[hasClockFrequency→ Y].
Intended set of result tuples
5 test2(CPU1,‘1600MHz’).
General rule for comparing query results with intended results
6 FORALL X,Y Undefined ←
7 (test1(X,Y) AND NOT test2(X,Y)) OR
8 (test2(X,Y) AND NOT test1(X,Y)).

In this small example, we have provided only one result tuple for testing, but the specification is modular and general enough to easily integrate sets of test cases.

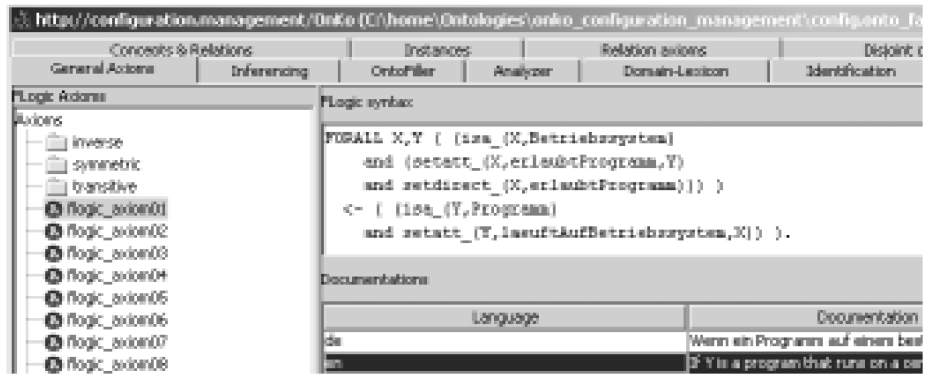


Fig. 4. Specifying F-Logic axioms in OntoEdit

## 6.2 Error Avoidance and Location

While the generation and validation of test cases allows for detection of errors, it does not really support the localization of errors. The set of all axioms, class and instance definitions express sometimes complex relationships and axioms often interact with other axioms when processed. Thus it is frequently very difficult to overview the correctness of a set of axioms and detect the faulty ones.

In principle there exist three types of problems with axioms:

- Axioms contain typing errors like variables not specified by a quantifier, typos in concept names or relationship names etc.
- Axioms contain semantic errors, i.e. the rules do not express the intended meaning.
- Performance issues, like axioms defined such that evaluation needs a lot of time, which is not always easily recognizable by the user.

In order to avoid problems, OntoEdit offers several means:

1. Some axiom definitions may be generated by asserting through clicks that relations or concepts belong to particular types. OntoEdit allows for defining several properties of relationships by clicking on the GUI, viz. symmetry, transitivity and composition of relations. Database connections as shown above in Section 5 need not be specified in F-Logic, but can be composed by drag-and-drop (cf. [HSV03]).
2. For other types of axioms a graphical rule editor is available which avoids syntactical errors, delivers axioms which are optimal in their performance (as seen in isolation from other axioms) and supports users not familiar in F-Logic.
3. Third, there are axioms that cannot be specified by either 1. or 2. For them, OntoEdit provides at least syntax highlighting in order to support the user avoiding syntactical errors.

In order to locate problems, OntoEdit takes advantage of the inference engine Ontobroker itself, which allows for introspection and also comes with a debugger. Axioms are operationalized by posing queries (e.g. on the test cases specified as seen above). Based on queries one may pursue several alternatives:

First, a very simple but effective method to test axioms with test cases is to switch off and switch on axioms or parts of the axiom premises. The different answers from Ontobroker then allow to draw conclusions about possible errors.

Second, for a given query the results and their dependencies on existing test instances and intermediate results may be examined by visualizing the proof tree. This proof tree shows graphically which instances or intermediate results are combined by which rules to the final answers. Thus the drawn inferences may be traced back to the test instances and semantic errors in rules may be discovered.

Third, the inference engine may be “observed” during evaluation. A graphical presentation of the set of axioms as a graph structure indicates which axiom is evaluated at the moment and also shows which intermediate results have already been created up to now and thus “have flown” in the axiom graph to other axioms. This also gives users a feeling how much time it is needed to evaluate special rules.

An example is given by the two Figures 4 and 5. The former illustrates how F-Logic axioms can be specified in OntoEdit in the “General Axioms” plugin (or, not shown here,

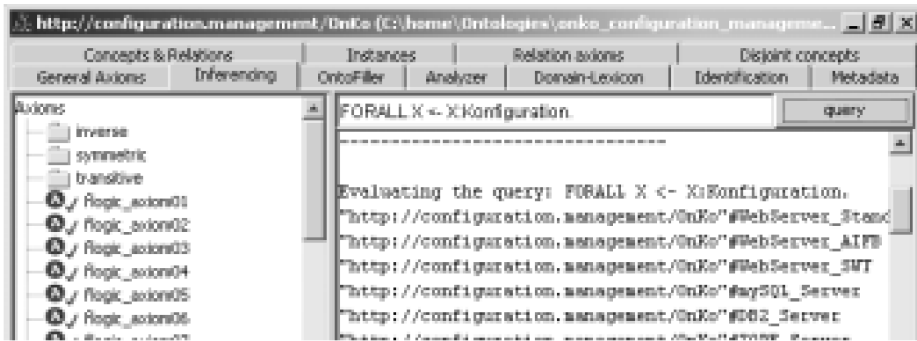


Fig. 5. Inferencing with Ontobroker in OntoEdit

a recently finished advanced graphical editor for specifying F-Logic axiom). On the left side are listed all specified axioms, on the right side one may see a selected F-Logic axiom as well as its documentation in different languages. The latter shows the GUI of the “Inferencing” plugin for OntoEdit which integrates Ontobroker into OntoEdit. On the left side each previously specified axiom can be switched on or off. On the right side one may enter an F-Logic query (here: give me all instances of the concept “Konfiguration”) which is subsequently answered by Ontobroker. The results are presented below the query, here one might see that for each result item the name of an instance includes the corresponding namespace.

In the future, it is planned to take more care about the efficient construction of efficiently handable ontologies. For this purpose, OntoEdit will provide a profiler that will deliver statistics about evaluation times.

### 6.3 Guidelines Evaluation

Guidelines for ontology modelling help to ensure a coherent modelling of ontologies and thus to ensure a consistent level of quality. This affects multiple ontology engineers working on the same ontology, but also single or multiple ontology engineers working on multiple ontologies. On the one hand, support for evaluation therefore inherently enhances collaboration (cf. [SEA<sup>+</sup>02]). On the other hand, we make extensive use of inferencing capabilities by applying axioms to ontologies for evaluating them.

An integration of guideline checking into ontology engineering environments helps to evaluate the guidelines during modelling time and guarantees immediate feedback for ontology engineers. However, different requirements for ontologies, e.g. coming from envisioned applications that should be supported, might require different guidelines. Therefore, a flexible way of using and adapting guidelines is needed instead of hard coding them. Guidelines might be used for technology-focussed evaluations, e.g. to ensure that naming conventions are fulfilled (for instance, some inference engines do not allow for white spaces in concept identifiers while others allow for them as long as brackets enclose the identifiers etc.), or for ontology-focussed evaluations, e.g. as shown in the next section on formal evaluations of ontologies.

The OntoAnalyzer plugin offers flexible and modularized checking of formalized guidelines and constraints.

From our own experiences of ontology development and deployment we learned that for different purposes ontologies must have different properties, e.g. for different target applications (cf., e.g., [LS02,SI02,DDS03]). The definition of evaluation methods for such properties must be very flexible and easily maintainable. So it is not convenient to program it into the OEE itself, but rather to have a modular and flexible way to ensure the quality of an ontology by checking such properties.

Logic is a very comfortable and powerful way on a conceptual level to express constraints for an ontology or to examine properties of an ontology. For that purpose the rule or constraint language must be able to access the ontology itself, i.e. to make statements about classes, relations, subclasses etc. In addition to the example presented in Section 5.3 we now present a more general way of formalising such guidelines.

**Formalization:** F-logic allows to define statements and rules about the ontology (concepts, subconcepts, relations). E.g. the examination whether in an ontology *a concept has at maximum one super concept* may be expressed by the following “check” axiom:

```
FORALL C CHECK("concept has more than one super concept",C) ←
  EXISTS S1, S2
    C :: S1
    AND C :: S2
    AND NOT EQUAL(S1, S2).
```

Further examples for modelling guidelines can be derived from [NM01]. E.g. consider the two guidelines (“slot” is synonymic used to our usage “relation”):

- If a list of classes defining a range of a slot includes a class and its subclass, remove the subclass.
- If a list of classes defining a domain of a slot includes a class and its subclass, remove the subclass.

They can be formalized as the following F-Logic axioms and automatically checked within OntoEdit<sup>7</sup>:

```
FORALL C CHECK("Remove concept from a range",C) ←
  EXISTS B, Domain, Rel
    Domain[Rel ⇒ B]
    AND Domain[Rel ⇒ C]
    AND C :: B.
```

```
FORALL C CHECK("Remove concept from a domain",C) ←
  EXISTS B, Range, Rel
```

<sup>7</sup> Please note that it is quite easy to present also the relation to which the domain and range concepts belong by extending the check predicate with an additional value.

$$\begin{aligned}
&B[Rel \Rightarrow Range] \\
&\text{AND } C[Rel \Rightarrow Range] \\
&\text{AND } C :: B.
\end{aligned}$$

**How does it work:** OntoAnalyzer is a tool which applies such axioms to an ontology opened in OntoEdit. The plugin transfers the ontology and check axioms to the Ontobroker inference engine and retrieves back and displays the results of this examination. OntoAnalyzer is able to load different axiom packages, each intended for a different target tool or target project.

To actually execute the checks, OntoAnalyzer asks for all values of the predicate  $CHECK(X,Y)$ , as shown in the following:

$$\text{FORALL } X, Y \leftarrow CHECK(X,Y).$$

The results are presented in the GUI of OntoAnalyzer (similarly to the results shown in Figure 9).

For the future we plan to develop standard rule packages for evaluation of various purposes to support efficient and effective engineering of ontologies and to improve the quality of ontologies at the same time. To enhance the usability of the plugin, the results derived from Ontobroker should immediately allow for the proposed actions, e.g. if a concept should be removed from the range of a relation, the dialog for doing so should be automatically opened.

## 6.4 Formal Evaluation with OntoClean

The previously introduced technology for checking ontologies can be extended to cover also formal ontology evaluations such as proposed by the OntoClean methodology (cf., e.g., [GW00a,GW00c,WG01,GW02]). The following building blocks constitute the basic infrastructure for implementing OntoClean: (i) a set of axioms that formalize definitions, constraints and guidelines given in OntoClean and (ii) a “meta-ontology”, viz. the so-called “taxonomy of properties”, that provides a frame of reference for evaluations. An ontology can be compared *vs.* a predefined ideal taxonomical structure to detect inconsistencies. Thus, the integration of the OntoClean methodology into ontology engineering environments like OntoEdit enables an integrated quality control for ontologies. The OntoClean plugin makes use of and extends the basic infrastructure of the OntoAnalyzer (cf. previous Section).

The OntoClean methodology is based on philosophical notions for a formal evaluation of taxonomical structures. It focuses on the cleaning of taxonomies and e.g. is currently being applied for cleaning the upper level of the WordNet taxonomy (cf. [GGOB02]). Core to the methodology are the four fundamental ontological notions of *rigidity*, *unity*, *identity*, and *dependence*. By attaching them as meta-relations to concepts in a taxonomy they are used to represent the behavior of the concepts. I.e. these meta-relations impose constraints on the way subsumption is used to model a domain (cf. [GW00b]). We can only briefly sketch the methodology in a simplified way and mention two of the introduced philosophical notions, viz. rigidity and unity:



- **Rigidity** is defined based on the idea of essence. A *property* is essential to an individual if and only if necessarily holds for that individual. Thus, a *property* is rigid (+R) if and only if it is necessarily essential to all its instances. A *property* is non-rigid (-R) if and only if it is not essential to some of its instances, and anti-rigid ( $\sim$ R) if and only if it is not essential to all its instances.

**Example:** Consider for example the property of *being hard*. We may say that it is an essential property of hammers, but not of sponges. Some sponges (dry ones) are hard, and some particular sponge may be hard for its entire existence, however this does not make being hard an essential property of that sponge. The fact is that it *could have* been soft at some time, it just happened that it never was.

Furthermore, *being a person* is usually conceptualized as rigid, while, as shown above, *being hard* is not. Rigidity is a subtle notion: every entity that *can* exhibit the property *must* exhibit it. So, every entity that is a person must be a person, and there are no entities that can be a person but aren't.

The property *being a student* is typically anti-rigid – every instance of student is not essentially a student (i.e. may also be a non-student).

- **Unity** is defined by saying that an individual is a whole if and only if it is made by a set of parts unified by a relation *R*. A *property P* is said to carry unity (+U) if there is a common unifying relation *R* such that all the instances of *P* are wholes under *R*. A *property* carries anti-unity ( $\sim$ U) if all its instances can possibly be non-wholes.

**Example:** E.g., the enterprize British Airways is a whole unified by the relation *has president*. To generalize, an *enterprize with president* carries unity since the relation *has president* is the relation that unifies every instance.

Based on these meta-relations OntoClean classifies concepts into categories as shown in Figure 6 (the figure is taken from [WG01]). E.g., a concept that is tagged with “+O +I +R” is called a “Type”.

The aim of the methodology is to produce a “clean” taxonomy as shown in the ideal structure in Figure 7 (figure is taken from [WG01]).

Beside these meta-relations OntoClean contains axioms that can be applied to evaluate the correctness of a given taxonomy. For instance, an axiom suggested in OntoClean is “a property carrying anti-unity has to be disjoint of a property carrying unity”. As a consequence, “a property carrying unity cannot be a subclass of a property carrying anti-unity” and “a rigid property and an anti-rigid property are ever disjoint”, to name but a few.

As an example we present the formalization of the disjointness in F-Logic:

```
FORALL C CHECK("A property cannot carry +R and -R",C) ←
  EXISTS B, Range, Rel
    B[Rel  $\Rightarrow$  Range]
    AND C[CARRYNOTR  $\rightarrow$  "true"].
```

Another example is, that a property that is defined as “anti-rigid” cannot subsume a property that is “rigid” (the check message is abbreviated for means of simplicity):

+O	+I	+R	+D	Type	Sortal
			-D		
-O	+I	+R	+D	Quasi-type	
			-D		
-O	+I	~R	+D	Material role	
-O	+I	~R	-D	Phased sortal	
-O	+I	¬R	+D	Mixin	
			-D		
-O	-I	+R	+D	Category	Non-Sortal
			-D		
-O	-I	~R	+D	Formal Role	
-O	-I	~R	-D	Attribution	
		¬R	+D		
			-D		
+O	-I			incoherent	
	+I	~R			
		-R			

Fig. 6. Combinations of OntoClean meta-relations

```

FORALL  $B$  CHECK("¬R can't subsume +R",  $B$ )  $\leftarrow$ 
  EXISTS  $C$ 
     $C :: B$ 
    AND  $B[\text{ANTIR} \rightarrow \text{"true"}]$ 
    AND  $C[\text{CARRYR} \rightarrow \text{"true"}]$ .

```

To implement the OntoClean methodology in OntoEdit, we

- formalized the constraints and definitions as axioms, and
- formalized the meta-relations and classifications as a “meta ontology” that can be used to classify concepts of an ontology.

We modelled the “meta ontology” and an example ontology (taken from [WG01]) that has to be evaluated, in OntoEdit. Each concept of the example ontology, i.e. all subconcepts of the root concept of the example ontology, viz. *Entity*, is then specified as being also an instance of the top-level concept *Property* of the meta ontology through an axiom:

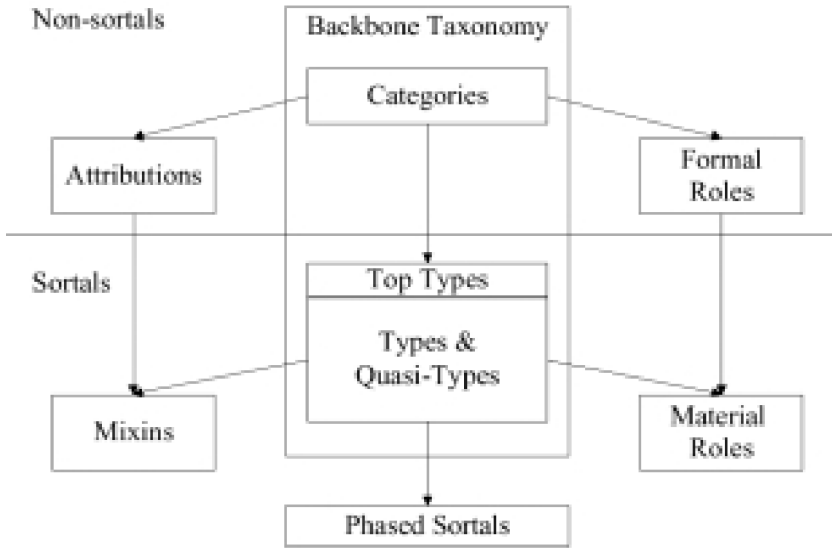


Fig. 7. Ideal taxonomy structure

FORALL  $C \ C : Property \leftarrow$   
 $C :: Entity$ .

Figure 8 shows the subsequent steps during implementation and employment of OntoClean as a plugin in OntoEdit (the numbers in the figure correspond to the following enumeration):

1. model both ontologies, the taxonomy of properties and the example ontology in OntoEdit,
2. fill the meta-relations with values (i.e. tag the concepts of the example ontology with `CARRYR (+R)` etc.), and
3. specify the definitions and constraints from OntoClean as axioms.

One can now ask queries to find inconsistencies in an ontology according to the OntoClean methodology. Figure 9 shows the inconsistencies derived from applying the OntoClean axioms the example ontology by making use of the inferencing capabilities of Ontobroker as a backend for OntoEdit. On the left side the list of implemented axioms is shown, for testing purposes they can be switched on and off. On the right side the result from an evaluation is shown. E.g. the concept *Agent* is defined as “anti-rigid” and subsumes the concept *Animal* which is defined as “rigid”. According to the OntoClean methodology this is a violation of a given constraint. To enhance the quality of the taxonomical structure an ontology engineer can now reconsider the modelled hierarchy.

The shown implementation is a first proof of concept. The next version of the plugin encapsules the meta ontology by using a dynamically built GUI to handle the tagging of concepts with meta-relations more intuitively. Similar to the OntoAnalyzer, the results

should automatically guide users through a set of possible actions that can be performed to fix the detected inconsistencies.

## 7 Related Work

There exist various ontology engineering environments, which we divide into two categories: *with* and *without* inferencing support.

A good overview, viz. a comparative study of existing tools up to 1999, is given in [DSW<sup>+</sup>00]. Typically the internal knowledge model of ontology engineering environments is capable of deriving is-a hierarchies of concepts and attached relations. On top of that we provide facilities for axiom modeling and debugging. Naturally, it could not fully consider the more recent developments, e.g. Protégé [NFM00], and WebODE [ACFLGP01].

About WebODE, [ACFLGP01] mentions that it offers inferencing services (developed in Prolog) and an axiom manager (providing functionalities such as an axiom library, axiom patterns and axiom parsing and verification), but the very brief mentioning of these functionalities is too short to assess precisely. The group from the Artificial Intelligence Laboratory of the Technical University of Madrid (UPM) is also working on

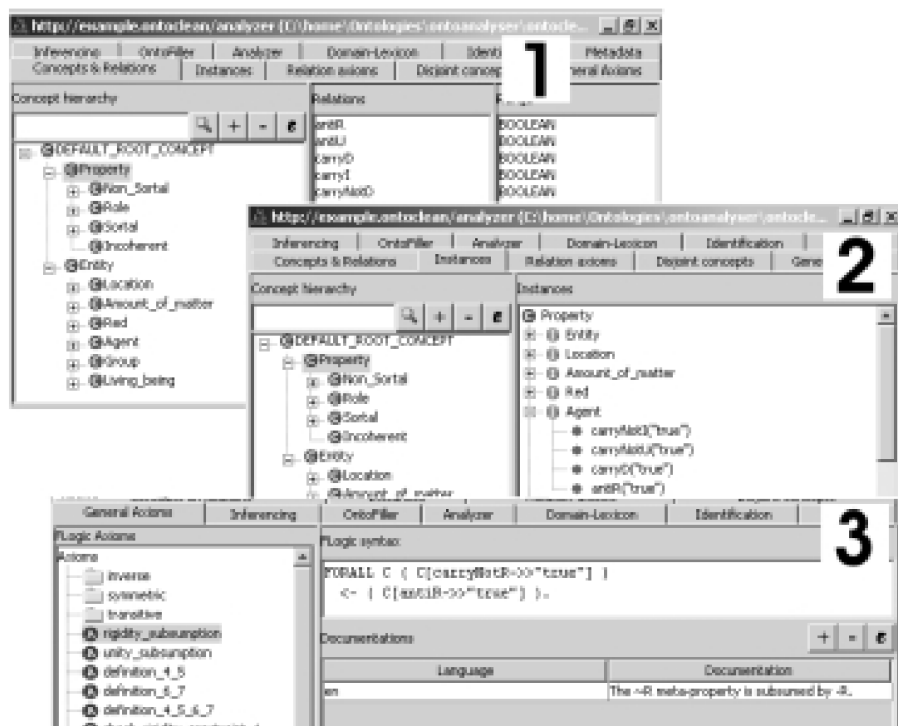


Fig. 8. Implementation of OntoClean in OntoEdit

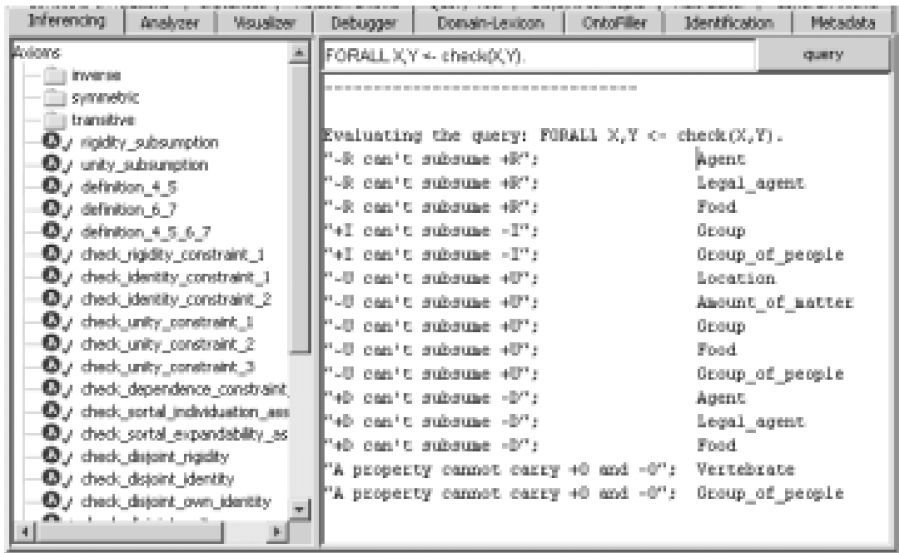


Fig. 9. Deriving inconsistencies with the OntoClean Plugin

the integration of the philosophically oriented OntoClean methodology with the process oriented METHONTOLOGY [FLGPSS99] by extending the WebODE [ACFLGP01] ontology development environment<sup>8</sup>.

A system well-known for its reasoning support is OilEd in combination with the description logics (DL) reasoner FaCT [BHGS01]. Their focus is to use reasoning to check class consistency and to infer subsumption relationships which are typical DL tasks. We may not provide subsumption, but we provide extensive reasoning on instances — in particular rules — or axioms that specify user-definable consistency constraints. Furthermore, we support the whole methodology cycle for developing ontologies.

Environments like Protégé [NFM00] or Chimaera [MFRW00] offer sophisticated support for ontology engineering and merging of ontologies. Protégé has also a flexible plugin-structure, that allows for modular extension of the functionalities. However, they do not provide comprehensive methodological support for ontology engineering and it is also difficult to assess the extent that they exploit reasoning capabilities.

## 8 Conclusion

In this paper we have presented OntoEdit, a sophisticated ontology editor that supports methodology-based ontology construction and that takes comprehensive advantage of its inferencing capabilities. OntoEdit also has some features that could not be presented here in full detail, e.g. an extremely capable plug-in structure (cf. [Han01]), a lexicon component and support for collaborative engineering of ontologies (cf. [SEA<sup>+</sup>02]).

<sup>8</sup> Cf. <http://www.ontoweb.org/workshop/ontoweb2/slides/ontocleansig3.pdf>

Obviously, there are a large number of ontology construction tools now available and many of them offer very intriguing features that are not in OntoEdit. However, according to our experiences the combination of a methodological basis with comprehensive reasoning on class and instance definitions with Ontobroker is a very powerful paradigm that has not been exploited before to the extent that OntoEdit does.

For the future, OntoEdit is planned to be developed in several directions: (1) new im- and exports will be developed and (2) the integration of ontology construction with requirements specification documents will be generalized by means of semantic document annotation, (3) stronger support for the integration of mind maps<sup>TM</sup> into the ontology development process, (4) finish the OntoClean implementation and apply it, to name but a few.

**Acknowledgements.** We thank our colleagues Alexander Maedche (now: Bosch AG, Germany) and Dirk Wenke (now: chief developer of OntoEdit at Ontoprise GmbH, Germany). Together they initiated the development of OntoEdit, which is now being continued by the constant efforts of Dirk. We also thank Siegfried Handschuh (Institute AIFB, University of Karlsruhe) for his plug-in framework OntoMat. Research for this paper was partially funded by EU in the project IST-1999-10132 “On-To-Knowledge” and in the thematic network IST-2000-29243 “OntoWeb”.

## References

- [ACFLGP01] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. WebODE: a scalable workbench for ontological engineering. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP) Oct. 21–23, 2001, Victoria, B.C., Canada*, 2001.
- [Ang93] J. Angele. *Operationalisierung des Modells der Expertise mit KARL*. Number 53 in DISKI. infix, St. Augustin, 1993. PhD Thesis.
- [BG02] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 12 November 2002, 2002. available at <http://www.w3.org/TR/PR-rdf-schema/>.
- [BHGS01] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A reason-able ontology editor for the semantic web. In *KI-2001: Advances in Artificial Intelligence*, LNAI 2174, pages 396–408. Springer, 2001.
- [Buz74] T. Buzan. *Use your head*. BBC Books, 1974.
- [DC02] R. Dieng and O. Corby, editors. *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management: Methods, Models, and Tools (EKAW 2000)*, volume 1937 of *Lecture Notes in Artificial Intelligence (LNAI)*, Juan-les-Pins, France, 2002. Springer.
- [DDS03] J. Davies, A. Duke, and Y. Sure. OntoShare – Evaluation of an ontology based knowledge sharing system. Submitted 2003, 2003.
- [Dec02] S. Decker. *Semantic Web Methods for Knowledge Management*. PhD thesis, Institute AIFB, University of Karlsruhe, 2002.
- [DEFS99] S. Decker, M. Erdmann, D. Fensel, and R. Studer. *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information*, pages 351–369. In Meersman et al. [MTS99], 1999.

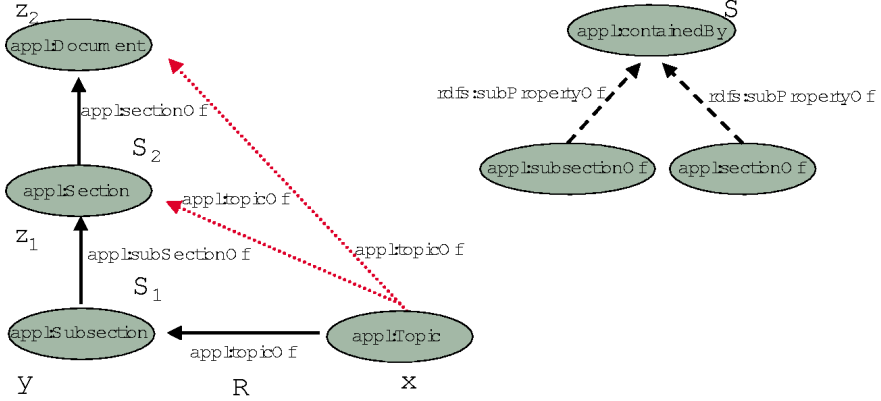
- [Dom98] J. Domingue. Tadzebao and WebOnto: Discussing, browsing, and editing ontologies on the web. In *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based Systems Workshop, April 18th–23rd. Banff, Canada, 1998*.
- [DSW<sup>+</sup>00] A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa, and V. R. Benjamins. Wondertools? a comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, 6(52):1111–1133, 2000.
- [Erd01] M. Erdmann. *Ontologien zur konzeptuellen Modellierung der Semantik von XML*. Books on Demand, 2001. PhD Thesis.
- [Fen95] D. Fensel. *The Knowledge Acquisition and Representation Language KARL*. Kluwer Academic Publisher, Boston, 1995.
- [FHKS96] J. Frohn, R. Himmeröder, P. Kandzia, and C. Schlepphorst. How to write F–Logic programs in FLORID. A tutorial for the database language F–Logic. Technical report, Institut für Informatik der Universität Freiburg, 1996. Version 1.0.
- [FLGPSS99] M. Fernández-López, A. Gómez-Pérez, J. P. Sierra, and A. P. Sierra. Building a chemical ontology using Methontology and the Ontology Design Environment. *Intelligent Systems*, 14(1), January/February 1999.
- [GGOB02] A. Gangemi, N. Guarino, A. Oltramari, and S. Borgo. Cleaning-up WordNet’s top-level. In *Proceedings of the 1st International WordNet Conference*, Mysore, India, January 2002.
- [GP96] A. Gómez-Pérez. A framework to verify knowledge sharing technology. *Expert Systems with Application*, 11(4):519–529, 1996.
- [GW00a] N. Guarino and C. Welty. A formal ontology of properties. Technical report, LADSEB/CNR Technical Report 01/2000, 2000. available at <http://www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html>.
- [GW00b] N. Guarino and C. Welty. A formal ontology of properties. In Dieng and Corby [DC02], pages 97–112.
- [GW00c] N. Guarino and C. Welty. Identity, unity, and individuality: Towards a formal toolkit for ontological analysis. *Proceedings of the European Conference on Artificial Intelligence (ECAI-2000), Berlin, 2000*, August 2000.
- [GW02] N. Guarino and C. Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, February 2002.
- [Han01] S. Handschuh. OntoPlugins – a flexible component framework. Technical report, University of Karlsruhe, May 2001.
- [HH02] I. Horrocks and J. A. Hendler, editors. *Proceedings of the First International Semantic Web Conference: The Semantic Web (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science (LNCS)*, Sardinia, Italy, 2002. Springer.
- [Hor98] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proceedings of the International Conference on Knowledge Representation (KR 1998)*, pages 636–649. Morgan Kaufmann, 1998.
- [HSR99a] Udo Hahn, Stefan Schulz, and Martin Romacker. Partonomic reasoning as taxonomic reasoning in medicine. In *Proc. of AAAI-99*, pages 271–276, 1999.
- [HSR99b] U. Hahn, S. Schulz, and M. Romacker. Part-whole reasoning: A case study in medical ontology engineering. *IEEE Intelligent Systems*, 14(5):59–67, 1999.
- [HSV03] S. Handschuh, S. Staab, and R. Volz. On deep annotation. In *Proceedings of the 12th International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20–24, 2003*. ACM Press, 2003.
- [KL86] M. Kifer and E. Lozinskii. A framework for an efficient implementation of deductive databases. In *Proceedings of the 6th Advanced Database Symposium*, pages 109–116, Tokyo, August 1986.
- [KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.

- [LS02] T. Lau and Y. Sure. Introducing ontology-based skills management at a large insurance company. In *Proceedings of the Modellierung 2002*, pages 123–134, Tutzing, Germany, March 2002.
- [MFRW00] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proceedings of KR 2000*, pages 483–493. Morgan Kaufmann, 2000.
- [MT<sup>+</sup>02] R. Meersman, Z. Tari, et al., editors. *Proceedings of the Confederated International Conferences: On the Move to Meaningful Internet Systems (CoopIS, DOA, and ODBASE 2002)*, volume 2519 of *Lecture Notes in Computer Science (LNCS)*, University of California, Irvine, USA, 2002. Springer.
- [MTS99] R. Meersman, Z. Tari, and S. Stevens, editors. *Database Semantics: Semantic Issues in Multimedia Systems*. Kluwer Academic Publisher, 1999.
- [NFM00] N. Noy, R. Fergerson, and M. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In Dieng and Corby [DC02], pages 17–32.
- [NM01] N. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05 and SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics, March 2001.
- [Ont02] Ontoprise. How to write F-Logic programs – a tutorial for the language F-Logic, 2002. Tutorial version 1.9 that covers Ontobroker version 3.5.
- [SEA<sup>+</sup>02] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the semantic web. In Horrocks and Hendler [HH02], pages 221–235.
- [SEM01] S. Staab, M. Erdmann, and A. Maedche. Ontologies in RDF(S). *ETAI Journal – Section on Semantic Web (Linkoepping Electronic Articles in Computer and Information Science)*, 9(6), 2001.
- [SI02] Y. Sure and V. Iosif. First results of a semantic web technologies evaluation. In *Proceedings of the Common Industry Program held in conjunction with Confederated International Conferences: On the Move to Meaningful Internet Systems (CoopIS, DOA, and ODBASE 2002)*, cf. [MT<sup>+</sup>02], 2002.
- [SSA02] Y. Sure, S. Staab, and J. Angele. OntoEdit: Guiding ontology development by methodology and inferencing. In Meersman et al. [MT<sup>+</sup>02], pages 1205–1222.
- [SSSS01] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems, Special Issue on Knowledge Management*, 16(1):26–34, January/February 2001.
- [Sur03] Y. Sure. *Methodology, Tools & Case Studies for Ontology based Knowledge Management*. PhD thesis, Institute AIFB, University of Karlsruhe, 2003.
- [UK95] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, Montreal, Canada, 1995.
- [van93] A. van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185–221, 1993.
- [vRS91] A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, July 1991.
- [WG97] G. Wiederhold and M. Genesereth. The conceptual basis for mediation services. *IEEE Expert / Intelligent Systems*, 12(5):38–47, September/October 1997.
- [WG01] C. A. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering*, 39(1):51–74, 2001.



## A Example of Partonomic Role Propagation

*Partonomic role propagation* is about propagating particular property values from parts to wholes. For instance, if the engine of my car is defunct, the whole car is defunct (“defunct” being propagated from the part “engine” to the whole “car”). However, if the rear window is broken, it might be less safe to drive the car, but it would be strange to consider it defunct.



**Fig. 10.** Partonomic Role Propagation

Figure 10 depicts an example that propagates topics from subparts of documents to superparts (using the graphical notion of RDF(S), cf. [BG02]). It takes four input parameters:

1. The relation that is propagated (TOPICOF in Figure 10), because not every relation is propagated from a part to a whole.
2. The relation that is propagating (CONTAINEDBY in Figure 10), because not every relation that may be propagated is propagated along all part-whole relations.
3. The whole up to which the relation is maximally propagated (Document in Figure 10), because propagation may be stopped (e.g. TOPICOF may be considered to be not propagated to an additional Library).
4. The concept for the instances of which the relation may be propagated (e.g., Topic in Figure 10), because not every class is treated the same (cf. [HSR99a] for comprehensive examples).

# Distributed Description Logics: Assimilating Information from Peer Sources

Alex Borgida<sup>1</sup> and Luciano Serafini<sup>2</sup>

<sup>1</sup> Dept. of Computer Science  
Rutgers University  
New Brunswick, USA  
`borgida@cs.rutgers.edu`

<sup>2</sup> ITC-IRST  
Trento, Italy  
`serafinit@itc.it`

**Abstract.** Due to the availability on the Internet of a wide variety of sources of information on related topics, the problem of providing seamless, integrated access to such sources has become (again) a major research challenge. Although this problem has been studied for several decades, there is a need for a more refined approach in those cases where the original sources maintain their own independent view of the world. In particular, we investigate those situations where there may not be a simple one-to-one mapping between the individuals in the domains of the various Information Sources.

Since Description Logics have already served successfully in information integration and as ontology languages, we extend this formalism with the ability to handle complex mappings between domains, through the use of so-called “bridge rules”. We investigate, among others, the exploitation of bridge rules to deduce new information, especially subsumption relationships between concepts in local information sources.

## 1 Introduction

A significant problem of modern information management is the integration of information from multiple sources. The standard version presumes a framework where users are accessing through a single interface data from several information sources (local ISs), which can include databases, web data, files, etc. The important goal here is to provide seamless access to the data, making the users unaware of the original source of the information. This is achieved by providing a single global schema, which traditionally was the result of merging the local schemas. In such situations, query processing consists of identifying the relevant ISs, translating the user’s query into collections of queries over local ISs, and collating the answers, expressed in terms of the global schema. In more traditional approaches, the global schema is defined by integrating the local schemas, and afterwards defining its contents through views over the local ISs, making query processing relatively easy. More recently, Levy et al [24] have investigated

an approach where the global, conceptual schema is developed independently, and then local IS are defined in terms of it.

A somewhat different, but related, approach is one which preserves the identity of each local IS and its user interface. However, the local system wishes to import information available in other sources, which are related to it directly through some kinds of assertion, or indirectly through chains of such relations. This approach, pioneered in the work on federated databases [17], is more appropriate for loosely related information sources, such as information appearing on the web, or distributed agents, where each source of information is independent. This paper is concerned with investigating this second, “peer to peer” kind of combination.

According to database integration research, the semantic world-views of local IS may exhibit mismatches (often called “conflicts” or “semantic heterogeneities”) which need to be resolved in order to allow information from one source to be properly visible in the other source. Saltor and Rodriguez [29] identify three high-level categories of such semantic heterogeneities: heterogeneities between object classes, between class structures, and between object instances. The first two categories relate to the schema of the IS, and have been thoroughly studied. Some of the conflicts in the third category deal with “the facts”: e.g., two IS may record the capital of China as Beijing and Peking respectively. In other situations, however, there may be more interesting, systematic relationships between individuals. For example, consider the case when one IS contains personal information (e.g., from credit card purchases), while another one contains census information, which only records information about *households*. The correspondence between households and the people in it is not the identity relation, neither is it a simple functional bijection. Yet it will be important to establish and record this relationship if the two IS are to be integrated. In Section 2, we provide further examples of complex correspondences between the domains of multiple IS in a federated system.

We must then decide in what context to explore the properties of such federated ISs. Description Logics are formalisms for knowledge representation and reasoning [3]. They have been used for a variety of roles in databases [7,8], including the description of data semantics and meta-data in the form of conceptual schemas. In turn, such DL conceptual schemas play a central role in many recent proposals for database integration, and more general information integration (e.g., [13,20,2,24,11,26,5]). Moreover, DMAL+OIL/OWL [18] – the current leading contenders as ontology language for the semantic web, are clear examples of Description Logics. And the term taxonomies used in web sites such as DMOZ and Yahoo!, or e-commerce stores, can be viewed as lightweight Description Logic ontologies. Finally, Tim Berners-Lee’s vision of the semantic web [4], explicitly abandons the notion of a universal ontology, and embraces the kind of “distributed” ontologies we have in mind.

For these reasons, we will carry out our investigations concerning complex correspondences between IS domains, in the context of DLs.

We start by providing several examples which motivate the new kinds of relationships between peer IS, and review related work in the database and ontology literature. After introducing informally Description Logics and Distributed Description Logics (DDL), we show how some examples are handled by them (Section 4). Formal definitions of DL and DDL are given in Section 5, and some desirable properties of DDL are presented in Section 6. Section 7 provides a theorem which shows that under some rather general conditions, DDL reasoning can be translated into reasoning in a single, global but ordinary DL. Finally, we investigate the properties of DDL in the case when component ontologies are simple taxonomies of atomic terms. We prove that, surprisingly, adding the feature of distributed reasoning does not increase the complexity of computation in this case.

## 2 Motivating Examples

Much, though by no means all, work in information integration, has assumed that the same individual appears in identical form in the different sources of information being integrated. We will look at a variety of examples where the correspondences between individuals in the domains of discourse of two IS,  $IS_1$  and  $IS_2$  say, is not so simple.<sup>1</sup>

First, in some cases one needs to identify/match different representations of the same real-world individual. In the case of scaling conflicts (e.g., use of different units of measure), this relationship is quite systematic, and can be described through mathematical equations. In other cases, this needs to be achieved through the use of key attributes or heuristics (e.g., persons having the same name). Either way, the result can be expressed as a binary relation between objects in the two IS, which is assumed to be a bijection.

Several complications may arise even in this situation. Consider the case when the unit of measure is currency: the conversion function `Euro_to_Dollars` is not the inverse of the function `Dollars_to_Euro`, because banks add a surcharge to all transactions. Therefore a single conversion relation is not sufficient, and we must acknowledge the need for **directional mappings** between the domains, e.g., one from Dollars to Euro, and another from Euro to Dollars. A different complication can arise in the case when the mapping between the domains cannot be described extensionally. For example, suppose that one school assigns simple letter grades, e.g., A,B,C, while another school allows them to be qualified by plus and minus, e.g., A+. An A at the first school corresponds to one of {A+,A,A-} at the other school, but there is no way to tell which. Note however that this partial information is still important: having a grade of “A+, A or A-” is known to be better than having a grade of B!

The following examples explore further intricacies of the relationship between domain elements in different IS.

---

<sup>1</sup> Some of these issues were noted already in [21].

**Example 1.** Suppose BasicC, IntermC and AdvancedC are 3 increasingly difficult courses on some topic. University Univ<sub>1</sub> offers BasicC and AdvancedC, while university Univ<sub>2</sub> offers IntermC. The universities are concerned about what courses a student has completed, in order to check the pre-requisites of other courses they are enrolling in, or to meet degree requirements. In particular, both universities allow a course  $x$  to be substituted for another course  $y$  if  $x$  is harder than  $y$ , and covers most of the material of  $y$  (say 80%). The universities also allow credits earned at one to be transferred to the other. Univ<sub>1</sub> may decide to accept IntermC as a substitute for BasicC; on the other hand, Univ<sub>2</sub> may only accept AdvancedC as a substitute for IntermC. Suppose that in the university information systems, courses are modeled as classes whose instances are the sets of students who have completed them. Then we have a situation where we want the instances of IntermC to be included among the instances of BasicC according to Univ<sub>1</sub>, while IntermC should include the instances of AdvancedC according to Univ<sub>2</sub>. Despite this, university Univ<sub>1</sub> may not necessarily want to see the elements of AdvancedC as instances of BasicC, since the courses might disagree on more than 80% of the material. This means that the relationship between the instances of courses at the two universities should not be expressed as simple subset, but must be mediated by some more subtle, directional, relations.

**Example 2.** Suppose IS<sub>1</sub> has information about married couples, while IS<sub>2</sub> has information about persons. We therefore need to express correspondences between individuals in the two domains, e.g., between couple23 in IS<sub>1</sub>, and each of Gianni and Mary, in IS<sub>2</sub>. But there are more general relationships between the information in the two IS. For example, we know that each couple involves exactly two objects, which are persons.

**Example 3.** Suppose IS<sub>lit</sub> maintains information about literary works, including plays, poems, novels, etc. On the other hand IS<sub>perf</sub> tracks performances in the past month, including stadium half-time shows, events at Stratford, and theatrical performances. We are interested in the correspondence between individuals in IS<sub>lit</sub> and IS<sub>perf</sub> which matches a literary work to its performances. Of course, some literary works were not performed in the past month, while others had multiple performances. So, in this case we need again a general binary relationship. Moreover, suppose we are told that all events in Stratford were performances of Shakespeare's tragedies, and all play performances were theatrical events; then, intuitively, we should be able to deduce, in IS<sub>perf</sub>, that all Stratford events were theatrical performances.

This shows that information about the nature of the correspondences can be used to deduce new facts in one IS (Stratford events are theatrical performances), based on facts in another IS (Shakespeare's tragedies are plays).

In this and other cases, IS<sub>1</sub> contains information about individuals that are in some sense *abstractions* over individuals in IS<sub>2</sub>. Similar examples will arise in most other situations where the so-called “materialization abstraction” [27]

occurs: one individual standing for a class of individuals in some more refined view of the world.

**Example 4.** *As a final example, consider a situation where there are three IS. Two of them,  $IS_{Harvard}$  and  $IS_{MIT}$ , serve the needs of college libraries in some town. The libraries have information about copies of books, which can be taken by borrowers or are available on the shelf. On the other hand, the third IS,  $IS_{Student}$ , is a database accessed by students who want to know which library they should go to if they need some book. Notice that the student does not care about which copies of a book are available, so we have once again an abstraction: the student's Tractatus corresponds to TractatusCopy1,... in  $IS_{Harvard}$ , as well as TractatusCopy2 in  $IS_{MIT}$ . Moreover, the student only wants to know about some material being located at MIT if there is a copy of it currently on the shelf at the MIT library.*

The above examples demonstrate the need to consider in greater detail the role of the mapping between the domain of objects in the IS's being integrated into a federation. First, one needs relations, not just functions, to describe the correspondences between the domains, as illustrated by Examples 2, 3 and 4. Second, Example 1 and the case of monetary conversion show that between any two IS, one needs a *pair* of correspondences, one in each direction. Moreover, there are two aspects of these mappings that will need to be considered:

- How are specific individual objects related to each other? (e.g., couple23 in  $IS_1$  and Mary in  $IS_2$ ).
- What general statements can one make about the mappings of individuals? (e.g., Couple instances in  $IS_1$  correspond to exactly two Person instances in  $IS_2$ ).

Finally, we will be interested in what new deductions one can make in a local IS, using the correspondences linking it to other IS.

### 3 Related Work on Information Integration

There has been a great deal of work on the problem of integrating databases. In the beginning, this work was motivated by the process of integrating user views in order to arrive at a “corporate” schema that satisfies the needs of everyone. Later, additional motivation was provided by the advent of heterogeneous/federated databases, and especially data warehouses, which by their nature integrate sources that may be radically different. Key questions addressed by this research include how to match up both schema and data-level information between multiple databases, and most effort has been devoted to heuristics and tools for *finding* such relationships. Rahm and Bernstein [28] provide a survey of recent solutions, while [16] is an early paper on the integration problem in the context of federated databases.

Information integration has also been studied in the field of Artificial Intelligence, where *ontologies* are essential components of the knowledge-rich environment required by problems such as natural language understanding. Klein [22] provides a recent review of the literature of ontology integration/alignment, which involves relating some of the terms in the two ontologies.

We have cited earlier three aspects that mark our work: (i) a co-ordinated (rather than unified) view of the information sources, (ii) the need to deal with complex relationships between individuals of the IS's being co-ordinated, and (iii) the use of Description Logics as the medium in which to explore the potential for *defining and reasoning* about information integration having the first two characteristics. Although there isn't much research that is specific to the first topic, we look more closely at the other two.

### 3.1 Expressing Relationships between Individuals of Different ISs

Kent [21] provides an extensive list of problems that arise due to data-level mismatches between databases, recognizing the need for both domain relationships (e.g., currencies) and context-dependent use of them (e.g., salaries vs. stock prices in different currencies). He examines complex solutions that use domain mapping functions orchestrated by integrator functions, and expresses them in the Iris database programming language.

SchemaLog [23] is just one representative of the class of declarative languages for relating multi-databases that use powerful data restructuring facilities. (The paper has a fine section reviewing other similar approaches.) Its higher-order syntax allows querying the schema, as well as inter-relating schema and value identifiers. Similar comments apply to work on integrating heterogeneous semi-structured data sources, as surveyed in [14].

A different declarative approach to the specific problem of data mediation is illustrated in [30], where meta-attributes and rules are used to deal with complex value conversions. A desirable feature of the approach is that it introduces the notion of "contexts", which allows for the *automatic invocation* of conversion functions.

### 3.2 Description Logics and Information Integration

Many approaches to information integration are founded on the belief that this is best accomplished at a higher, semantic/conceptual level, and Description Logics have been used for the purpose of describing the data semantics in a many efforts, including [13,20,2,24,11,26,5]. Most of these papers therefore describe methods of ontology integration, for the case when ontologies are described in DLs, and often rely on the ability to *reason* with Description Logic concepts in order to perform the integration, detect inconsistencies, or to translate queries.

Calvanese et al [12] present a detailed mechanism for specifying data integration in the context of data warehouses. The conceptual data model is specified in a DL, and acts in some sense as a global terminology, while the local ISs have

relational schemas, which are connected via Datalog-like rules augmented by important information concerning keys and domains. The aspect that distinguishes their work, and corresponds more closely to intended focus of our paper, is the specification of “*reconciliation correspondences*”, including ones for conversion, matching and merging of specific data. These can be thought of as describing different aspects of instance-level connections. Their work is characterized by the use of rules, with a declarative part and a procedural part, performing operations such as conversions. Unfortunately, reasoning with rules is not amalgamated with reasoning about schema concepts expressed in DL, and complex procedural aspects of rules make this improbable in principle.

The goal of the present paper is to study the declarative specification of constraints on correspondences between the domains of multiple IS in a manner which makes it possible to reason about the final, co-ordinated system. We investigate, among others, the exploitation of correspondence rules to deduce new relationships between concepts in local information sources.

## 4 Informal Introduction to Distributed Description Logics

We present next an informal introduction to Description Logics and to our extension of them, Distributed Description Logics. Readers comfortable with formal presentations are welcome to skip this section.

### 4.1 Description Logics

Description Logics are a family of object-centered knowledge representation formalisms. They view the world as being populated by individuals that can be grouped into classes, called *concepts*, and that can be related to each other by binary relationships, called *roles*. A specific DL provides a particular set of “constructors” for building more complex concepts and roles, much like a programming language type system provides type constructors for building complex types from simpler ones. For example, concept constructors such as conjunction (written as  $A \sqcap B$ ) and value restriction ( $\forall r.C$ ) can be used to describe object-oriented style classes with multiple superclasses and type-like constraints on members/attributes: the Java class specification **class** STUDENT **extends** PERSON{INTEGER stdId; UNIVERSITY attends;} asserts, among others, that student objects satisfy a type that might be described in DL notation as  $\text{PERSON} \sqcap \forall \text{stdId}.\text{INTEGER} \sqcap \forall \text{attends}.\text{UNIVERSITY}$ . The following examples of descriptions are meant to provide additional intuitions about the ability to nest descriptions and to use them to specify both necessary and sufficient conditions:

- primitive concepts PERSON, UNIVERSITY, INTEGER
- primitive roles attends, hasAge, hasLocation
- $\forall \text{attends}.\text{UNIVERSITY}$  (\* *Objects whose attends role values are instances of UNIVERSITY* \*)



- $\forall \text{hasLocation}.\text{NEW\_ENGLAND} \ (* \text{ Objects located in places that are considered to be New England } *)$
- $\forall \text{attends} . (\text{UNIVERSITY} \sqcap \forall \text{hasLocation}.\text{NEW\_ENGLAND}) \ (* \text{ Objects attending universities in New England } *)$
- $\text{PERSON} \sqcap \forall \text{attends} . (\text{UNIVERSITY} \sqcap \forall \text{hasLocation}.\text{NEW\_ENGLAND}) \ (* \text{ Persons attending universities in New England } *)$

As seen in the above examples, a description usually corresponds to a noun-phrase (a unary formula in logic). In some DLs, it is possible to construct more complex roles as well. For example, one can use  $\text{attends}^-$  to refer to the inverse of the  $\text{attends}$  role, which might otherwise be called  $\text{hasEnrolees}$ .

One can then use description terms for several different tasks.

First, one can define a new concept in terms of an intentional description of the properties of its members. For example,  $\text{NEW\_ENGLAND}$  may itself be defined as  $\forall \text{hasAddress} . \forall \text{inState} . \{\text{Maine}, \text{Vermont}, \dots\}$ .

Second, one can claim that one description,  $D$ , *subsumes* or *is more general than* another one,  $C$ , written as  $C \sqsubseteq D$ , meaning that anything that satisfies the conditions of  $C$  must, by necessity, satisfy the conditions of  $D$ . This is useful in asserting necessary conditions, such as  $\text{STUDENT} \sqsubseteq \text{PERSON}$ , rather than the necessary and sufficient conditions provided by definitions. One can also use  $\sqsubseteq$  to make more general statements, sometimes called axioms, relating various terms in a complex ontology. For example, if  $\exists p.C$  is a concept constructor representing objects that have at least one  $p$  role filler in the concept  $C$ , then  $(\text{STUDENT} \sqcap \forall \text{attends} . \text{IVY-LEAGUE}) \sqsubseteq (\text{PERSON} \sqcap \exists \text{hasParents} . \text{RICH-PERSON})$  says that students attending Ivy League universities must have at least one rich parent. Collections of such assertions specify the terminology used to describe some application domain. Such a collection is called a *T-box*, and resembles the schema of a database, or an ontology. Subsumption has additional uses. According to its formal definition, it is possible to deduce new subsumptions, just like it is possible to deduce new implications from a logical theory. For example,  $\forall \text{attends} . \text{UNIVERSITY}$  subsumes  $\text{PERSON} \sqcap \forall \text{attends} . \text{IVY-LEAGUE}$ , assuming that  $\text{UNIVERSITY}$  can be deduced to subsume  $\text{IVY-LEAGUE}$ . The subsumption relation provides the specification of an algorithm which can be used to organize the terminology of the domain into the familiar IS-A hierarchy, and to detect whether some description is incoherent, in the sense that it cannot hold of any individual. (Incoherence is detected by checking for subsumption by the empty concept.)

Third, one can assert the membership of an *individual* in a concept, as a way of giving it a partial description. For example,  $\text{STUDENT} \sqcap \neg \text{MALE}(\text{Anna})$  asserts that Anna is a student who is not male. In addition, one can assert the inter-relatedness of two individuals, e.g.,  $\text{attends}(\text{Anna}, \text{Harvard})$ . Collections of assertions about individuals partially describe some state of world, and form an *A-box*, which resembles a database state (the collection of tuples in relations, for example). As with subsumption, not only can one assert membership in concepts, but one can deduce/compute whether some arbitrary individual is an instance of an arbitrary description, possibly given some A-box and T-box.

For example,  $\exists \text{ attends.IVY-LEAGUE}(\text{Anna})$  can be deduced from the above two assertions, assuming that **Harvard** in turn is, or can be deduced to be, an instance of **IVY-LEAGUE**.

In a DL, the complexity of reasoning about questions such as subsumption and concept membership depends on the particular concept and role constructors (e.g.,  $\forall, \sqcap, \exists$ ) available. Much of the effort in DL research has been in identifying different sets of such constructors which have at least decidable inferences, and characterizing their computational complexity [3].

## 4.2 Distributed Description Logics

Suppose we have a two information sources  $IS_1$  and  $IS_2$ , each using some (potentially different) description logic to describe its contents. Let us now try to express connections between them, as a way of aligning them. The pioneering work of Catarci and Lenzerini [13] proposed the continued use of description logics. In particular, subsumption assertions could relate descriptions in different knowledge bases:  $\text{GradStudent}_2 \sqsubseteq_{\text{int}} \text{Student}_1$  would indicate that every graduate student in the part of the world described by  $IS_2$  was also a student in the overlapping part of the world described by  $IS_1$ . However, the semantics in [13] implies that inter-schema assertions only have an effect for those *individuals that are shared* between the respective IS domains – i.e., the correspondence between the domain elements is identity. The reason for this is, in part, that in current DL the definition of a new concept can only retrieve a subset of the existing set of individuals, rather than create new individuals.

In order to deal with our more complex examples, we turn for inspiration to the work of Ghidini and Serafini [15] on Distributed First Order Logic. The idea is to introduce, at least conceptually, some binary relation  $\mathbf{r}_{12}$  describing the correspondences between the domains of  $IS_1$  and  $IS_2$ , and to use so-called **bridge rules** to constrain these relationships *in an implicit manner*.

In order to best support directionality, we will need two sets of bridge rules,  $\mathfrak{B}_{12}$  and  $\mathfrak{B}_{21}$ , with rules in  $\mathfrak{B}_{12}$  viewed as describing “flow of information” from  $IS_1$  to  $IS_2$  *from the point of view of  $IS_2$* , i.e.,  $IS_2$  is “importing” information from  $IS_1$ . The rules in  $\mathfrak{B}_{12}$  and  $\mathfrak{B}_{21}$ , when given, will constrain the possible correspondences  $\mathbf{r}_{12}$  and  $\mathbf{r}_{21}$  respectively.

Based on studies in [15], here are some patterns of constraints on the correspondence relationships that one might like to express using bridge rules:

1. Every instance of concept  $A$  ( $A$ -instances) in  $IS_1$  corresponds only to a  $G$ -instances in  $IS_2$ ; i.e.,  $\mathbf{r}_{12}(A) \subseteq G$ .
2. All  $G$ -instances in  $IS_2$  have a corresponding  $A$ -instance in  $IS_1$ ; i.e.,  $\mathbf{r}_{12}^{-1}(G) \subseteq A$ , or  $G \subseteq \mathbf{r}_{12}(A)$ .
3. Each  $A$ -instance has at least/at most  $n$  corresponding  $G$ -instances in  $IS_2$ .
4. The domain relation from  $IS_1$  to  $IS_2$  is the identity relationship.
5. The domain relations between  $IS_1$  and  $IS_2$  are symmetric; i.e.,  $\mathbf{r}_{12} = \mathbf{r}_{21}^{-1}$ .

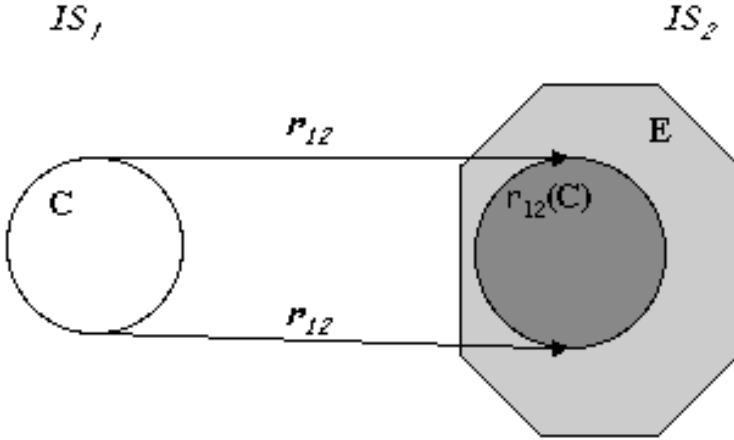
In this paper we will study the first two kinds of bridge rules.

In order to distinguish the descriptions from various  $IS_i$ , ( $i = 1, 2$ ) we start by labeling them with the index  $i$ , written as  $i:E$ . (However, when talking about subsumption within a single  $IS_i$ , we will use the more readable  $i:A \sqsubseteq B$ , instead of the formally correct  $i:A \sqsubseteq i:B$ .) We now introduce the two kinds of bridge rules:

Given concepts  $C$  and  $E$  of  $IS_1$  and  $IS_2$  respectively, a *bridge rule from  $IS_1$  to  $IS_2$*  has one of two forms:

- an *into-bridge rule*  $1:C \xrightarrow{\sqsubseteq} 2:E$ , specifying that  $C$ -objects in  $IS_1$  correspond only to  $E$ -objects in  $IS_2$  (according to the  $r_{12}$  relation);
- an *onto-bridge rule*  $1:C \xrightarrow{\supseteq} 2:E$ , stating that every object in  $E$  has a corresponding pre-image in concept  $C$  of  $IS_1$ .

The intuition behind into- and onto-bridge rules is shown in Figures 1 and 2.



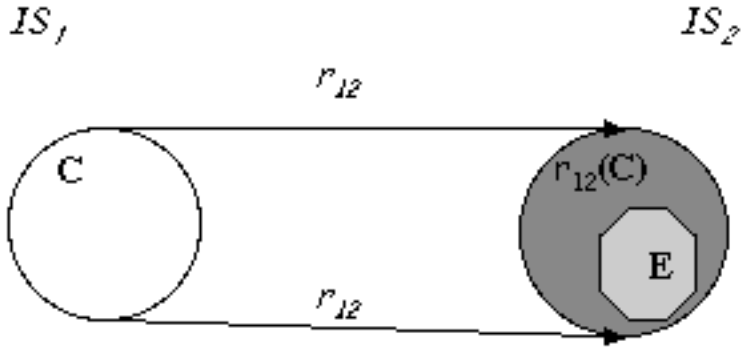
**Fig. 1.** Intuitive diagram of  $C \xrightarrow{\sqsubseteq} E: r_{12}(C) \subseteq E$

As an illustration of using the above terms, in Example 2 one would have the into-bridge rule  $1:\text{COUPLE} \xrightarrow{\sqsubseteq} 2:\text{PERSON}$  to indicate that couples correspond only to persons, but would *not* normally include the onto-bridge rule  $1:\text{COUPLE} \xrightarrow{\supseteq} 2:\text{PERSON}$ , because there may be unmarried persons, who would not be partners in any couple.

### 4.3 Some Examples Revisited

We will now recast some of the earlier examples into the notation of DDLs, and briefly consider some of the subtle aspects of the representation.

To begin with, the course correspondences in Example 1 yield two bridge rules:



**Fig. 2.** Intuitive diagram of  $C \xrightarrow{\exists} E: E \subseteq r_{12}(C)$

$$\begin{aligned} 1:\text{ADVANCED\_C} &\xrightarrow{\sqsubseteq} 2:\text{INTERM\_C} \\ 2:\text{INTERM\_C} &\xrightarrow{\sqsubseteq} 1:\text{BASIC\_C} \end{aligned}$$

These allow each IS to import appropriate information from the other, but do not entail  $1:\text{ADVANCED\_C} \sqsubseteq \text{BASIC\_C}$ , because the two bridge rules concern different mappings:  $r_{12}$  and  $r_{21}$  respectively. This is a good example of the peer-to-peer approach, where each IS imports information independently.

In Example 3, the correspondences we had specified would be represented as two bridge rules:

$$\begin{aligned} \text{lit}:\text{PLAY} &\xrightarrow{\sqsubseteq} \text{perf}:\text{THEATRICAL\_PERFORMANCES} \\ \text{lit}:\text{SHAKESPEAREAN\_TRAGEDY} &\xrightarrow{\exists} \text{perf}:\text{STRATFORD\_EVENTS} \end{aligned}$$

Finally, in Example 4, involving libraries, we have three T-boxes  $\mathbf{T}_h$ ,  $\mathbf{T}_m$  and  $\mathbf{T}_s$ .  $\mathbf{T}_h$  and  $\mathbf{T}_m$  describe the information systems of the libraries. They both have concepts **BOOK**, corresponding to copies of books that can be loaned, and concepts **PERSON**, to model the borrowers. The role **taken\_by** is meant to record who has borrowed a book, so that the concept **BOOK\_ON\_SHELF**, in  $\mathbf{T}_h$ , can be defined as follows

$$h:\text{BOOK\_ON\_SHELF} \equiv \text{BOOK} \sqcap \neg \exists \text{taken\_by}.\text{PERSON}$$

The T-box  $\mathbf{T}_s$ , modeling the students' information system, also includes a concept called **BOOK**, but its meaning is different, since, as we mentioned before, these are abstractions over the libraries' copies of the book. The following bridge rules are intended to capture the fact that students see books based on copies from the respective libraries

$$h:\text{BOOK} \xrightarrow{\sqsubseteq} s:\text{BOOK} \tag{1}$$

$$m:\text{BOOK} \xrightarrow{\sqsubseteq} s:\text{BOOK} \tag{2}$$

Note that this does not imply that *all* books at the Harvard library can be seen in the students database (the mapping  $r_{hs}$  may be partial). Nor does it imply that the same book copy cannot be mapped to several books in  $IS_{Student}$  – this would require a different kind of bridge rule, one expressing that the correspondence has a certain cardinality.

In addition,  $T_s$  uses the role `located_at` to capture the name of the library where the student should go to in order to get the material in question, assuming it is available there. For convenience,  $T_s$  contains a concept, `AVAILABLE_BOOK`, that lets students tell quickly if some book is available:

$$s:\text{AVAILABLE\_BOOK} \equiv \text{BOOK} \sqcap \exists \text{located\_at}$$

Since students only want to hear about material located at a library if there are some copies of it on the shelf there, we use onto-bridge rules as follows:

$$h:\text{BOOK\_ON\_SHELF} \xrightarrow{\exists} s:\exists \text{located\_at}.\{\text{"Harvard"}\} \quad (3)$$

$$m:\text{BOOK\_ON\_SHELF} \xrightarrow{\exists} s:\exists \text{located\_at}.\{\text{"Mit"}\} \quad (4)$$

As we shall explain formally later, one of consequences of DDL reasoning with bridge rules (1) and (3) is the following subsumption

$$s:\exists \text{located\_at}.\{\text{"Harvard"}\} \sqsubseteq \text{AVAILABLE\_BOOK}$$

Intuitively, this follows because the bridge rules allow us to infer in  $IS_{Student}$  that anything that is located at the Harvard library must be a book, and hence an instance of the concept `AVAILABLE_BOOK` defined earlier.

#### 4.4 Distributed A-Boxes

In order to deal with correspondences between specific individuals, we can follow two approaches. First, if the description logic is sufficiently expressive, we can state such correspondences by using bridge rules. For example, the correspondence of `couple23` to Gianni and Mary in Example 2, can be expressed by bridge rules  $1:\{\text{couple23}\} \xrightarrow{\sqsubseteq} 2:\{\text{Gianni, Mary}\}$  and  $1:\{\text{couple23}\} \xrightarrow{\exists} 2:\{\text{Gianni, Mary}\}$ , if the description logics support concepts formed by enumeration.

Otherwise, we need to introduce the individual-level equivalent of bridge-rules: If  $x$  is an individual in  $IS_1$ , while  $y, y_1, \dots$  are individuals of  $IS_2$ , then

- a *(partial) individual correspondence* is an expression  $1:x \mapsto 2:y$ , indicating that  $r_{12}(x, y)$  holds;
- a *complete individual correspondence* is an expression  $1:x \mapsto 2:\{y_1, y_2, \dots\}$ , indicating that  $\{y_1, y_2, \dots\}$  is the set of all  $z$  such that  $r_{12}(x, z)$ .

The second kind of correspondence is needed because  $1:\text{couple23} \mapsto 2:\text{Gianni}$  and  $1:\text{couple23} \mapsto 2:\text{Mary}$  allows in principle additional objects to be related to `couple23`. This possibility is eliminated by the complete correspondence  $1:\text{couple23} \mapsto 2:\{\text{Gianni, Mary}\}$ , which lists the entire set of objects corresponding to `couple23`.

## 5 Formal Definitions

We present next the formal specification of the above notions.

### 5.1 Formal Syntax and Semantics of Description Logics

A typical DL starts with several primitive notions: atomic concepts  $A$ , as well as constant concepts **ANYTHING** and **NOTHING**, denoting the universe and the empty set respectively<sup>2</sup>; atomic roles  $P$ ; and possibly individual names  $In$ . Complex concepts  $C$  and roles  $R$  can then be built according to the recursive syntax in the second column of Figure 3.

Construct name	Syntax	Semantics
primitive concept	$A$	$A^{\mathcal{I}}$
top concept	<b>ANYTHING</b>	$\Delta^{\mathcal{I}}$
bottom concept	<b>NOTHING</b>	$\emptyset$
conjunction	$C_1 \sqcap \dots \sqcap C_n$	$C_1^{\mathcal{I}} \cap \dots \cap C_n^{\mathcal{I}}$
disjunction	$C_1 \sqcup \dots \sqcup C_n$	$C_1^{\mathcal{I}} \cup \dots \cup C_n^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
value restriction	$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
exists restriction	$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
number restrictions	$\geq n R$	$\{d \in \Delta^{\mathcal{I}} \mid  R^{\mathcal{I}}(d)  \geq n\}$
	$\leq n R$	$\{d \in \Delta^{\mathcal{I}} \mid  R^{\mathcal{I}}(d)  \leq n\}$
qualified number restriction	$\geq n R.C$	$\{d \in \Delta^{\mathcal{I}} \mid  R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}  \geq n\}$
	$\leq n R.C$	$\{d \in \Delta^{\mathcal{I}} \mid  R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}  \leq n\}$
same-as	$R_1 = R_2$	$\{d \in \Delta^{\mathcal{I}} \mid R_1^{\mathcal{I}}(d) = R_2^{\mathcal{I}}(d)\}$
enumeration	$\{In_1, \dots, In_n\}$	$\{In_1^{\mathcal{I}}, \dots, In_n^{\mathcal{I}}\}$
Role		Semantics
primitive role	$P$	$P^{\mathcal{I}}$
role inverse	$R^-$	$\{(y, x) \mid (x, y) \in R^{\mathcal{I}}\}$
role composition	$R_1 \circ R_2$	$\{(x, z) \mid y \in R_1^{\mathcal{I}}(R_2^{\mathcal{I}}(x))\}$
role conjunction	$R_1 \sqcap R_2$	$\{(x, y) \mid y \in R_1^{\mathcal{I}}(x) \text{ and } y \in R_2^{\mathcal{I}}(x)\}$
role disjunction	$R_1 \sqcup R_2$	$\{(x, y) \mid y \in R_1^{\mathcal{I}}(x) \text{ and } y \in R_2^{\mathcal{I}}(x)\}$
role negation	$\neg R$	$\{(x, y) \mid (x, y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \text{ and } y \notin R^{\mathcal{I}}\}$

**Fig. 3.** Syntax and semantics of some typical Description Logic constructs

In order to provide a denotational semantics for DLs, we begin with the notion of an interpretation/structure  $\mathcal{I}$ , which is a pair  $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ , where  $\Delta^{\mathcal{I}}$  is a non-empty domain of objects, and  $\cdot^{\mathcal{I}}$  is a *valuation*, which maps every concept to a subset of  $\Delta^{\mathcal{I}}$ , every role to a subset of  $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ , and every individual to a

<sup>2</sup> In order to simplify matters, we will use the term “atomic concept” to also refer to the special concepts **ANYTHING** and **NOTHING**.

distinct element of  $\Delta^{\mathcal{I}}$ . The denotation of complex descriptions is obtained from that of its components using the rules in column 3 of Figure 3<sup>3</sup>.

In any interpretation  $\mathcal{I}$ , the subsumption relation  $\alpha \sqsubseteq \beta$  will be said to hold between concepts or roles  $\alpha$  and  $\beta$ , if  $\alpha^{\mathcal{I}} \subseteq \beta^{\mathcal{I}}$ . This will be written formally as  $\mathcal{I} \models \alpha \sqsubseteq \beta$ . The subsumption  $\alpha \sqsubseteq \beta$  will be said to hold in general if  $\mathcal{I} \models \alpha \sqsubseteq \beta$  for all possible interpretations  $\mathcal{I}$ .

A *T-box*, or terminology, is then a collection of axioms of the form  $\alpha \sqsubseteq \beta$  or  $\alpha \equiv \beta$ . For the purposes of this paper, we will replace  $\alpha \equiv \beta$  by two subsumption axioms,  $\alpha \sqsubseteq \beta$  and  $\beta \sqsubseteq \alpha$ . We can now introduce convenient notation to talk about interpretations “satisfying” T-boxes.

- $\mathcal{I} \models \mathbf{T}$  (\* read as “the subsumptions in  $\mathbf{T}$  are satisfied in  $\mathcal{I}$ ” \*) if  $\mathcal{I} \models \alpha \sqsubseteq \beta$ , for all  $\alpha \sqsubseteq \beta$  in T-box  $\mathbf{T}$ .
- $\mathbf{T} \models C \sqsubseteq D$  (\* “ $D$  subsumes  $C$  with background theory/axioms  $\mathbf{T}$ ” \*) if  $\mathcal{I} \models C \sqsubseteq D$  for all interpretations  $\mathcal{I}$  such that  $\mathcal{I} \models \mathbf{T}$ .

An *A-box* is a collection of assertions of the form  $C(a)$ , where  $C$  is a concept and  $a$  is an individual identifier, and  $R(a, b)$ , where  $R$  is a role, and  $a$  and  $b$  are individual identifier. The definitions of satisfaction in an interpretation is extended to A-boxes according to the rules

- $\mathcal{I} \models C(a)$  if  $a^{\mathcal{I}} \in C^{\mathcal{I}}$ .
- $\mathcal{I} \models R(a, b)$  if  $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$ .
- $\mathcal{I} \models \mathbf{A}$  if  $\mathcal{I} \models \pi$  for every assertion  $\pi = C(a), R(a, b) \in \mathbf{A}$ .
- $\mathbf{K} \models C(a)$  iff  $\mathcal{I} \models C(a)$  for all interpretations  $\mathcal{I}$  such that  $\mathcal{I} \models \mathbf{K}$ . Similarly for  $p(a, b)$ .

A (DL) *knowledge base*  $\mathbf{K}$  will then be a pair  $\langle \mathbf{T}, \mathbf{A} \rangle$ , where  $\mathbf{T}$  is a terminology and  $\mathbf{A}$  is an A-box, which uses only descriptions valid according to  $\mathbf{T}$ .

The complexity of reasoning in DLs with a wide variety of concept and role constructors has been studied. *ALCN* [9], *DLR* [10], and *SHIQ* [19] are among the best known. This complexity usually increases when background theories are used, unless these can be eliminated by “unfolding” or “incorporating” them. We mention that in *SHIQ* subsumption is decidable in worst-case exponential deterministic time, even in the presence of T-boxes, and in fact current algorithms behave much better on knowledge bases encountered in practice.

## 5.2 Formal Syntax and Semantics of Distributed Description Logics

Supposing  $I$  is a set of indices, we start with a collection of description logics  $\{\mathcal{DL}_i\}_{i \in I}$ . Each of these is used to describe either T-boxes or knowledge bases for information sources  $IS_i$ . The notion of Distributed Description Logic (DDL) is centered around relations  $\mathbf{r}_{ij}$ , relating the individuals in the interpretations of  $IS_i$  and  $IS_j$ . However, these relations will only appear in the *semantics* of DDLs – in the syntax one will only see constraints on their properties, expressed as bridge rules.

<sup>3</sup> Given a binary relation  $r$ , we use  $r(x)$  to refer to the set  $\{y \mid r(x, y)\}$ .

**Definition 1.** Given concepts  $C$  and  $E$  of  $\mathcal{DL}_i$  and  $\mathcal{DL}_j$  respectively, a bridge rule from  $i$  to  $j$  ( $i \neq j$ ), is an expression of one of the following two forms:

$i:C \xrightarrow{\sqsubseteq} j:E$ , an into-bridge rule

$i:C \xrightarrow{\sqsupseteq} j:E$ , an onto-bridge rule

If  $x$  is an individual in  $IS_i$ , while  $y, y_1, \dots$  are individuals of  $IS_j$ , then an individual correspondence from  $i$  to  $j$  is an expression of one of the following two forms:

$i:x \mapsto j:y$  a (partial) individual correspondence

$1:x \mapsto 2:\{y_1, y_2, \dots\}$  a complete individual correspondence

**Definition 2.** A distributed T-box (DTB)  $\mathfrak{T} = \langle \{\mathbf{T}_i\}_{i \in I}, \mathfrak{B} \rangle$  consists of a collection of ordinary DL T-boxes  $\{\mathbf{T}_i\}_{i \in I}$ , and a collection  $\mathfrak{B} = \{\mathfrak{B}_{ij}\}$ , where  $\mathfrak{B}_{ij}$  is a set of bridge rules from  $i$  to  $j$ , for  $i \neq j \in I$ . For every  $k \in I$ , all descriptions in  $\mathbf{T}_k$  must be in the corresponding language  $\mathcal{DL}_k$ , and for every bridge rule  $i:A \xrightarrow{\sqsubseteq} j:B$  or  $i:A \xrightarrow{\sqsupseteq} j:B$  in  $\mathfrak{B}_{ij}$ , the concepts  $A$  and  $B$  must be in the languages  $\mathcal{DL}_i$  and  $\mathcal{DL}_j$  respectively.

A distributed A-box (DAB)  $\mathfrak{A} = \langle \{\mathbf{A}_i\}_{i \in I}, \mathfrak{C} \rangle$  consists of a set of A-boxes  $\{\mathbf{A}_i\}_{i \in I}$ , and a set  $\mathfrak{C} = \{\mathfrak{C}_{ij}\}_{i \neq j \in I}$  of partial and complete individual correspondences from  $i$  to  $j$ . For every  $k \in I$ , all descriptions in  $\mathbf{A}_k$  must be in the corresponding language  $\mathcal{DL}_k$ , and for every correspondence rule  $i:x \mapsto j:y$  or  $i:x \mapsto j:\{y_1, y_2, \dots\}$  in  $\mathfrak{C}_{ij}$ , the individual name  $x$  must be in  $\mathcal{DL}_i$ , and  $y_1, y_2, \dots$  in  $\mathcal{DL}_j$ .

A DDL knowledge base is then a pair  $\langle \mathfrak{T}, \mathfrak{A} \rangle$ , consisting of a distributed T-box and a distributed A-box

We provide semantics for distributed description logics by using local interpretations for the individual information systems, and connecting their domains using relations  $\mathbf{r}_{ij}$ .

**Definition 3.** A distributed interpretation  $\mathfrak{J} = \langle \{\mathcal{I}_i\}_{i \in I}, \{\mathbf{r}_{ij}\}_{i,j \in I} \rangle$  of  $\mathfrak{T}$  consists of interpretations  $\mathcal{I}_i$  for  $\mathcal{DL}_i$  over domain  $\Delta^{\mathcal{I}_i}$ , and binary relations  $\mathbf{r}_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ , for  $i \neq j \in I$ .

**Definition 4.** A distributed interpretation  $\mathfrak{J}$  d-satisfies (written  $\mathfrak{J} \models_d$ ) the elements of a DTB  $\mathfrak{T} = \langle \{\mathbf{T}_i\}_{i \in I}, \mathfrak{B} \rangle$  according to the following clauses. For  $i, j \in I$

- $\mathfrak{J} \models_d i:A \xrightarrow{\sqsubseteq} j:G$ , if  $\mathbf{r}_{ij}(A^{\mathcal{I}_i}) \subseteq B^{\mathcal{I}_j}$  (\* Satisfaction of into-bridge rules \*)
- $\mathfrak{J} \models_d i:B \xrightarrow{\sqsupseteq} j:H$ , if  $\mathbf{r}_{ij}(B^{\mathcal{I}_i}) \supseteq H^{\mathcal{I}_j}$  (\* Satisfaction of onto-bridge rules \*)
- $\mathfrak{J} \models_d i:A \sqsubseteq B$ , if  $\mathcal{I}_i \models A \sqsubseteq B$  (\* Satisfaction of local subsumptions \*)
- $\mathfrak{J} \models_d \mathbf{T}_i$  if  $\mathcal{I}_i \models \mathbf{T}_i$ . (\* Satisfaction of local T-boxes \*)
- $\mathfrak{J} \models_d \mathfrak{T}$  if, for every  $i \in I$ ,  $\mathfrak{J} \models_d \mathbf{T}_i$ , and  $\mathfrak{J}$  d-satisfies every bridge rule in the union of the sets in  $\mathfrak{B}$ .



Finally,  $\mathfrak{I} \models_d i:C \sqsubseteq D$  if, for every distributed interpretation  $\mathfrak{J}$ ,  $\mathfrak{J} \models_d \mathfrak{I}$  implies  $\mathfrak{J} \models_d i:C \sqsubseteq D$ .

Concerning individuals, we have the following

**Definition 5.** A distributed interpretation  $\mathfrak{J}$  d-satisfies the elements of a DAB  $\mathfrak{A} = \{\langle \mathbf{A}_i \rangle_{i \in I}, \mathfrak{C}\rangle$  according to the following clauses. For  $i, j \in I$

- $\mathfrak{J} \models_d i:x \mapsto j:y$ , if  $y^{\mathcal{I}_j} \in \mathbf{r}_{ij}(x^{\mathcal{I}_i})$  (\* Satisfaction of individual correspondences \*)
- $\mathfrak{J} \models_d i:x \mapsto \{y_1, y_2, \dots\}$  if  $\mathbf{r}_{ij}(x^{\mathcal{I}_i}) = \{y_1^{\mathcal{I}_j}, y_2^{\mathcal{I}_j}, \dots\}$  (\* Satisfaction of complete correspondences \*)
- $\mathfrak{J} \models_d i:C(a)$ , if  $\mathcal{I}_i \models C(a)$  (\* Satisfaction of local assertions \*)
- $\mathfrak{J} \models_d i:p(a, b)$ , if  $\mathcal{I}_i \models p(a, b)$  (\* Satisfaction of local assertions \*)
- $\mathfrak{J} \models_d \mathbf{A}_i$  iff  $\mathfrak{J} \models_d \text{BasicC}(a)$  and  $\mathfrak{J} \models_d p(a, b)$  for every assertion  $C(a)$  and  $p(a, b)$  in  $\mathbf{A}_i$ . (\* Satisfaction of local A-boxes \*)
- $\mathfrak{J} \models_d \mathfrak{A}$  if, for every  $i \in I$ ,  $\mathfrak{J} \models_d \mathbf{A}_i$ , and  $\mathfrak{J}$  d-satisfies every individual correspondence in the union of the sets in  $\mathfrak{C}$ .

Finally,  $\mathfrak{A} \models_d i:C(a)$  if, for every distributed interpretation  $\mathfrak{J}$ ,  $\mathfrak{J} \models_d \mathfrak{A}$  implies  $\mathfrak{J} \models_d i:C(a)$ . Similarly for  $p(a, b)$ .

### 5.3 Revisiting Some Examples Formally

Consider first the library example, and ignore, for simplicity, one of the libraries,  $\mathbf{T}_m$  say. This problem is encoded by defining the distributed T-box  $\mathfrak{T}_{lib} = \langle \mathbf{T}_h, \mathbf{T}_s, \{\mathfrak{B}_{hs}\} \rangle$ , where

$\mathbf{T}_h = \{\text{BOOK\_ON\_SHELF} \equiv \text{BOOK} \sqcap \neg \text{taken\_by.PERSON}\}$ ,  $\mathbf{T}_s = \{\text{AVAILABLE\_BOOK} \equiv \text{BOOK} \sqcap \exists \text{located\_at}\}$ , and  $\mathfrak{B}_{hs} = \{ \vdash : h\text{BOOK} \xrightarrow{\sqsubseteq} s:\text{BOOK}, h:\text{BOOK\_ON\_SHELF} \xrightarrow{\sqsupseteq} s:\exists \text{located\_at}.\{\text{"Harvard"}\} \}$ .

Figure 4 provides an example distributed interpretation  $\mathfrak{J}_{lib}$  for  $\mathfrak{T}_{lib}$ .

Note that bridge rule  $h:\text{BOOK} \xrightarrow{\sqsubseteq} s:\text{BOOK}$  is satisfied by  $\mathfrak{J}_{lib}$  even though  $\text{BOOK}^{\mathcal{I}_h}$  is not contained in  $\text{BOOK}^{\mathcal{I}_s}$ ; indeed,  $\mathbf{r}_{hs}(\text{BOOK}^{\mathcal{I}_h}) = \{\text{Tractatus}\}$ , is a subset of  $\text{BOOK}^{\mathcal{I}_s}$ , which is  $\{\text{Tractatus}, \text{Philosophical\_Investigations}\}$ .  $\mathfrak{J}_{lib}$  also satisfies the second bridge rule above, since  $\mathbf{r}_{hs}(\text{BOOK\_ON\_SHELF}^{\mathcal{I}_h}) = \mathbf{r}_{hs}(\{\text{TractCpy3}, \text{DB\_Pples}\}) = \{\text{Tractatus}\}$ , which is a superset of  $(\exists \text{located\_at}.\{\text{"Harvard"}\})^{\mathcal{I}_s} = \{\text{Tractatus}\}$ .

Next, consider the bridge rules for Example 3:

$$\begin{aligned} \text{lit}:\text{SHAKESPEAREAN\_TRAGEDY} &\xrightarrow{\sqsupseteq} \text{perf}:\text{STRATFORD\_EVENTS} \\ \text{lit}:\text{PLAY} &\xrightarrow{\sqsubseteq} \text{perf}:\text{THEATRICAL\_PERFORMANCES} \end{aligned}$$

Using words in *italics* for the extensions of the corresponding concepts, the above translate to semantic constraints  $\text{StratfordEvents} \subseteq \mathbf{r}_{12}(\text{ShakespeareanTragedy})$  and  $\mathbf{r}_{12}(\text{Play}) \subseteq \text{TheatricalPerformances}$ . Together with the subsumption  $\text{SHAKESPEAREAN\_TRAGEDY} \sqsubseteq \text{PLAY}$ ,<sup>4</sup>

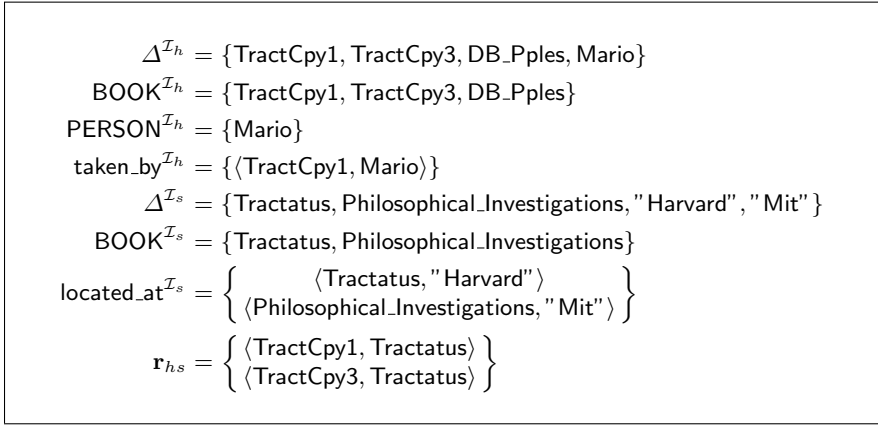
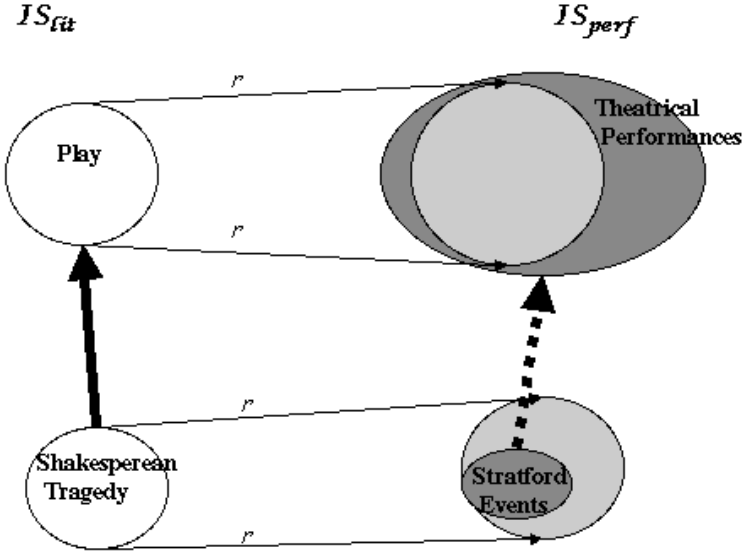
Fig. 4. Example of distributed interpretation for  $\mathcal{I}_{lib}$ 

Fig. 5. Example of Inferred Subsumption Relationship (dashed line) in DDL

entailing the set relationship  $\text{ShakespeareanPlay} \subseteq \text{Play}$ , we get the chain  $\text{StratfordEvents} \subseteq \mathbf{r}_{12}(\text{ShakespeareanPlay}) \subseteq \mathbf{r}_{12}(\text{Play}) \subseteq \text{TheatricalPerformances}$  which justifies the conclusion

$$\text{perf} : \text{STRATFORD\_EVENT} \sqsubseteq \text{THEATRICAL\_PERFORMANCE}.$$

<sup>4</sup> In a more complete example, this could be *deduced* from a definition of SHAKESPEAREAN-TRAGEDY, such as  $\text{PLAY} \sqcap \forall \text{author} . \{\text{Shakespeare}\} \sqcap \forall \text{genre} . \{\text{tragedy}\}$ .

Figure 4 illustrates this reasoning.

The above example exhibits a common pattern of inference in DDL:

starting from  
      $A$  is mapped *into*  $G$  by a bridge rule  
      $A$  subsumes  $B$  in  $IS_1$   
      $B$  is mapped *onto*  $H$  by a bridge rule  
 conclude that  
      $G$  subsumes  $H$  in  $IS_2$

By applying the same inference pattern to the library example, we can deduce that  $s:\exists \text{located\_at.}\{\text{"Harvard"}\} \sqsubseteq \text{BOOK}$ , and since  $s:\exists \text{located\_at.}\{\text{"Harvard"}\} \sqsubseteq \exists \text{located\_at.}$ , we can conclude  $s:\exists \text{located\_at.}\{\text{"Harvard"}\} \sqsubseteq \text{AVAILABLE\_BOOK}$ .

## 6 On the Basic Properties of DDL

Based on our initial motivations, the following is a list of intuitively desirable properties for a DDL  $\mathfrak{T} = \langle \{\mathbf{T}_i\}_{i \in I}, \mathfrak{B} \rangle$

1. When deducing things at  $IS_i$  in the distributed system, all local information should be available. Formally, if  $\mathbf{T}_i \models X \sqsubseteq Y$ , then  $\mathfrak{T} \models_d i: X \sqsubseteq Y$ .
2. In the absence of bridge rules, no information should pass between the component systems. Formally, if  $\langle \{\mathbf{T}_i\}_{i \in I}, \emptyset \rangle \models_d i: X \sqsubseteq Y$  then  $\mathbf{T}_i \models X \sqsubseteq Y$ .
3. A DDL should exhibit “directionality”/“no backflow”: we have said that  $\mathfrak{B}_{ij}$  contains bridge rules that are designed to provide information flow from  $IS_i$  to  $IS_j$ . Therefore, such a setup should not affect, by itself, reasoning in  $IS_i$ . Formally, we would like to have that if  $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$ , and  $\mathfrak{T}_{12} \models_d 1: A \sqsubseteq B$ , then  $\mathbf{T}_1 \models A \sqsubseteq B$ . This would also allow for more effective reasoning because there would be no need for a feedback loop between new inferences in  $IS_2$  and those in  $IS_1$ .
4. A collection of independent ISs should not allow local inconsistencies to “pollute” the entire system, in the sense that if the information at  $IS_k$  is not satisfiable (e.g., because  $k:\text{ANYTHING} \sqsubseteq k:\text{NOTHING}$ ), then deductions at other sites should not be affected. Formally, this would mean that if  $\mathbf{T}_1$  is inconsistent then  $\mathfrak{T}_{12} \models_d 2: X \sqsubseteq Y$  iff  $\mathbf{T}_2 \models X \sqsubseteq Y$ .

It can be easily seen that our definition of DDL does have the first two properties, if the component T-boxes are consistent. Unfortunately, the other two properties do not always hold.

Concerning “backflow”, it is possible to use onto bridge rules to enforce certain properties of descriptions in  $IS_1$ . For example, a rule such as  $1:A \xrightarrow{\exists} 2:\text{ANYTHING}$  requires every distributed interpretation to have the property  $\mathbf{r}_{12}(A^{\mathcal{I}_1}) \neq \emptyset$ , because the extension of the **ANYTHING** concept is never empty, and hence there must always be individuals in  $A^{\mathcal{I}_1}$  that correspond to it. This kind of reasoning can sometimes be translated into new subsumptions for  $IS_1$ , which depend on  $IS_2$ . To show this, let us introduce a role constant,  $\text{Universal}_{\text{ROLE}}$ .

It relates every possible pair of objects:  $\text{Universal}_{\text{ROLE}}^{\mathcal{I}_1} = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1}$ . Now consider the DTB  $\langle \emptyset, \{\text{ANYTHING} \sqsubseteq G\}, \{1:A \xrightarrow{\exists} 2:G\} \rangle$ . It d-entails the subsumption  $1:\text{ANYTHING} \sqsubseteq \exists \text{Universal}_{\text{ROLE}}.A$  because if  $A$ 's interpretation is never empty then there is always some  $A$ -object to which every object can be related by  $\text{Universal}_{\text{ROLE}}$ .

By using Theorem 1 in the next section, and information about the specific reasoning algorithms used, it is possible to prove that in some DDLs, such as ones using T-boxes expressed in the *SHIQ* description logic, no backflow will occur.

As far as inconsistency propagation is concerned, if one of the component T-boxes is unsatisfiable then there is no distributed interpretation satisfying the entire DTB, and therefore every possible assertion is automatically d-entailed by an unsatisfiable DTB. (This kind of inconsistency propagation is not peculiar to DDL – it is a feature of most logic-based formal models of IS with multiple components.)

There are several options in dealing with this problem. One alternative is to modify the definition of d-entailment so that unsatisfiable T-boxes are excluded in the definition of the integrated system. A different approach is to search for a “semantic” solution which has the desired properties without explicitly checking for unsatisfiability. A number of such “non-standard” semantic approaches are possible: The original paper on Distributed First Order Logics [15] uses *sets* of ordinary interpretations to achieve this effect. We have considered simpler versions of this, such as allowing special interpretations that assign the empty set to every concept, including **ANYTHING**, or the entire domain to every concept, including **NOTHING**. Such interpretations satisfy every subsumption, including  $\text{ANYTHING} \sqsubseteq \text{NOTHING}$ , and therefore all T-boxes, including inconsistent ones, have at least one satisfying interpretation. A full investigation of the properties of various proposals is beyond the scope of this paper.

## 7 Relating DDL and Ordinary DL

We are interested in finding a connection between DLs and DDLs, which would allow us to transfer theoretical results (such as complexity analysis) and reasoning techniques from the extensive current DL literature.

We will do so by building a “global” DL, which encodes the information available in the local T-boxes and the bridge rules of the DDL. To do so, we start by creating a language,  $\mathcal{GDL}$ , for descriptions in this global DL. Suppose one is given a family of description logics  $\{\mathcal{DL}_i\}_{i \in I}$ . For any primitive concept  $A$  (resp. role  $R$ ) of  $\mathcal{DL}_i$ , let  $i:A$  (resp.  $i:R$ ) be a primitive concept (resp. role) of  $\mathcal{GDL}$ . Moreover,  $\mathcal{GDL}$  will use all the concept and role constructors appearing in any  $\mathcal{DL}_i$ . Therefore  $\mathcal{GDL}$  permits the equivalents of at least all composite descriptions of  $\mathcal{DL}_i$ .  $\mathcal{GDL}$  has the usual top and bottom concepts, **ANYTHING** and **NOTHING**, which in this case are distinguished from the tops and bottoms of the hierarchies in  $\mathcal{DL}_i$ , which are now ordinary concepts, with names  $i:\text{ANYTHING}$

and  $i:\text{NOTHING}$ . (To emphasize this, we will use instead the symbols  $\text{Top}_i$  and  $\text{Bot}_i$ .)  $\mathcal{GDL}$  has a special set of role symbols  $\text{R}_{ij}$ , which will be used to simulate the domain relations, as well as an additional role symbol  $\hat{P}$ , used strictly for technical reasons.

We start by providing a translation  $\#()$  from  $\mathcal{DL}_i$  concepts/roles to concepts/roles in the global DL, and then extending it to map entire DDL T-boxes. Not unexpectedly, such a translation will be based on the recursive structure of concepts, which is based on the use of DL *constructors*. To emphasize this, we will view concept and role constructors as functors,  $\rho$ , that take as arguments simpler descriptions. For example,  $\forall p.D$  can be viewed as  $\mathbf{all}(p, D)$ , where  $\mathbf{all}(\langle \text{role expression} \rangle, \langle \text{concept expression} \rangle)$  is the constructor functor. The mapping  $\#$  is defined recursively as follows:

1.  $\#(i, M) = i:M$  for atomic concepts and roles  $M$  (including  $M = \text{ANYTHING}$  and  $M = \text{NOTHING}$ ). (\* Atomic base cases. \*)
2. if  $\rho$  is a concept constructor taking  $k$  arguments, then  $\#(i, \rho(M_1, \dots, M_k)) = \text{Top}_i \sqcap \rho(\#(i, M_1), \dots, \#(i, M_k))$ . (\* Complex concepts are translated by first translating the constructor's arguments, and then intersecting with the domain  $\text{Top}_i$ . \*)
3. if  $\rho$  is a role constructor taking  $k$  arguments, then  $\#(i, \rho(M_1, \dots, M_k)) = \rho(\#(i, M_1), \dots, \#(i, M_k))$ . (\* Complex roles are translated by first translating the arguments.<sup>5</sup> \*)

For example,  $\#(i, \text{PERSON} \sqcap \forall \text{likes}^- . \text{TEACHER})$  produces

$$\text{Top}_i \sqcap i:\text{PERSON} \sqcap \forall (i:\text{likes})^- . i:\text{TEACHER}$$

We are now ready to construct the global DL T-box:

**Definition 6.** Applying  $\#$  to a DTB  $\mathfrak{T} = \langle \{\mathbf{T}_i\}_{i \in I}, \{\mathfrak{B}_{ij}\}_{i,j \in I} \rangle$ , yields a T-box  $\#(\mathfrak{T})$  in the language  $\mathcal{GDL}$ , consisting of the following axioms:

1. (\* Copies of axioms from local T-boxes. \*)  
 $\#(i, A) \sqsubseteq \#(i, B)$  for all  $i:A \sqsubseteq B \in \mathbf{T}_i$ ;
2. (\* Translations of into bridge rules as value restrictions on  $\text{R}_{ij}$ . \*)  
 $\#(i, A) \sqsubseteq \forall \text{R}_{ij} . \#(j, G)$ <sup>6</sup> for every into bridge rule  $i:A \xrightarrow{\sqsubseteq} j:G \in \mathfrak{B}_{ij}$ ;
3. (\* Translations of onto bridge rules as existential restrictions on the inverse of  $\text{R}_{ij}$ . \*)  
 $\#(j, H) \sqsubseteq \exists \text{R}_{ij}^- . \#(i, A)$  for every onto bridge rule  $i:A \xrightarrow{\sqsupseteq} j:H \in \mathfrak{B}_{ij}$ ;
4. (\* Restricting  $\text{Bot}_i$  to always be the incoherent concept. \*)  
 $\text{Bot}_i \sqsubseteq \text{NOTHING}$ .
5. (\* Ensuring that  $\text{Top}_i$  is the proper local top of its IS-A hierarchy \*)  
 $i:A \sqsubseteq \text{Top}_i$ , for every atomic concept  $A$  of  $\mathcal{DL}_i$

<sup>5</sup> We do not use top role and role conjunction instead of **ANYTHING** and  $\sqcap$  in the previous item, because these constructors are rarely present in DLs, unlike **ANYTHING** and  $\sqcap$  are ubiquitous.

<sup>6</sup> This is logically equivalent to  $\exists \text{R}_{ij}^- . \#(i, A) \sqsubseteq \#(j, G)$ .

6. (\* Ensuring that  $Top_i$  is not empty \*)  
 $ANYTHING \sqsubseteq \exists \hat{P}.Top_i$
7. (\* Ensuring that every i-role  $p$  has as domain and range  $Top_i$  \*)  
 $Top_i \sqsubseteq \forall(i:p).(Top_i)$  for every role  $p$  of  $\mathcal{DL}_i$  (\* the range of  $i:p$  is in  $\Delta^{\mathcal{I}_i}$  \*);  
 $\exists(i:p).ANYTHING \sqsubseteq Top_i$  (\*  $i:p$  is only defined over  $\Delta^{\mathcal{I}_i}$  \*)

We propose to show next that under certain quite general circumstances d-entailment can be reduced to ordinary DL-reasoning through the use of the above translation; namely that  $\#(\mathfrak{T}) \models \#(i, X) \sqsubseteq \#(i, Y)$  if and only if  $\mathfrak{T} \models_d i:X \sqsubseteq Y$ .

The proof would be based, as usual, on constructing an appropriate interpretation for  $\#(\mathfrak{T})$ , given one for  $\mathfrak{T}$ , and conversely. In turn, the proof of correctness of the constructions relies on induction on the structure of concepts. So the proof depends on the concept and role constructors of the particular  $\mathcal{DL}$ 's involved. In fact, the proof is quite similar for a great variety of constructors,  $\rho$ , so we will state it in more general terms in order to abstract out this commonality.

We start by noting that the semantics of most descriptions is *compositional*, so that the denotation of  $\rho(arg_1, \dots, arg_n)$  is a function,  $f_\rho$ , of the denotation of its arguments. For example,  $\mathbf{all}(\mathbf{R}, \mathbf{C})^{\mathcal{I}}$  is equal to  $f_{all}(R^{\mathcal{I}}, C^{\mathcal{I}})$  for an associated “semantic function”  $f_{all}(XR, XC) = \{d \in \Delta^{\mathcal{I}} \mid XR(d) \subseteq XC\}$ . However, sometimes such a function depends on more than the interpretation of its arguments — in this case the set  $\Delta^{\mathcal{I}}$ . Although we can imagine more general scenarios, we will consider here only constructors that depend in some explicit way on  $\Delta^{\mathcal{I}}$ . We will require that the function  $f_\rho$  make this dependency explicit by taking an additional argument,  $DY$ , which is to be instantiated by  $\Delta^{\mathcal{I}}$  when finding the denotation of a particular concept constructed with  $\rho$ . For example,  $f_{all}$  should take three arguments,  $XR$ ,  $XC$  and  $DY$ , with  $f_{all}(XR, XC, DY)$  defined by  $\{d \in DY \mid XR(d) \subseteq XC\}$ ; and  $(\forall p.C)^{\mathcal{I}}$  would be expressed as  $f_{all}(p^{\mathcal{I}}, C^{\mathcal{I}}, \Delta^{\mathcal{I}})$ .

Semantic functions with the following property will be of interest to us:

**Definition 7.** Let  $\rho$  be a concept constructor whose semantics can be expressed as  $\rho(arg_1, \dots, arg_n)^{\mathcal{I}} = f_\rho(arg_1^{\mathcal{I}}, \dots, arg_n^{\mathcal{I}}, \Delta^{\mathcal{I}})$ , for a function  $f_\rho(X_1, \dots, X_n, DY)$  whose definition contains no references to  $\mathcal{I}$ .<sup>7</sup> Let  $B_1, \dots, B_n, W, \Delta$  be sets such that  $W \subseteq \Delta$ , and each  $B_j$  is either a subset of  $W$  or of  $W \times W$ ,  $1 \leq j \leq n$ . Then  $\rho$  is called a local constructor if  $f_\rho$  satisfies the condition

$$W \cap f_\rho(B_1, \dots, B_n, \Delta) = f_\rho(B_1, \dots, B_n, W)$$

when  $\rho$  is a concept constructors, or

$$f_\rho(B_1, \dots, B_n, \Delta) = f_\rho(B_1, \dots, B_n, W)$$

when  $\rho$  is a role constructor.

The main result is then

<sup>7</sup> In the case when one of the arguments  $arg_k$  of the constructor  $\rho$  is not a concept or a role description (e.g., for number restrictions  $\leq n p$ , one argument is an integer),  $arg_k^{\mathcal{I}}$  is extended to evaluate to  $arg_k$ .

**Theorem 1.** *Suppose  $\mathfrak{T} = \langle \{\mathbf{T}_i\}_{i \in I}, \mathfrak{B} \rangle$  is a DTB in  $\{\mathcal{DL}_i\}$ , where every concept and role constructor is local. Then  $\#(\mathfrak{T}) \models \#(i, X) \sqsubseteq \#(i, Y)$  if and only if  $\mathfrak{T} \models_d i : X \sqsubseteq Y$ .*

*Proof*

“If”. Let  $\mathfrak{J} = \langle \{\mathcal{I}^i\}, \{\mathbf{r}_{ij}\} \rangle$  be a d-interpretation. Define  $\mathfrak{J}_G$  to be the interpretation with domain  $\Delta^{\mathfrak{J}_G} = \uplus_i \Delta^{\mathcal{I}^i}$  (the disjoint union of the domains of  $\mathcal{DL}_i$ ), which interprets  $i : A$  in  $\mathcal{GDL}$  according to the rule  $\#(i, A)^{\mathfrak{J}_G} = A^{\mathcal{I}^i}$ , whenever  $A$  is an atomic concept or atomic role. Moreover, let  $\mathbf{r}_{ij}^{\mathfrak{J}_G}$  equal  $\mathbf{r}_{ij}$ , and  $\hat{P}^{\mathfrak{J}_G} = \Delta^{\mathfrak{J}_G} \times \Delta^{\mathfrak{J}_G}$ . We then start by verifying that

$$\#(i, M)^{\mathfrak{J}_G} = M^{\mathcal{I}^i}$$

for arbitrary concepts and role  $M$  in  $\mathcal{DL}_i$ . The proof is by structural induction on  $M$ . The base cases for atomic concepts and roles hold by definition. So consider some composite concept description  $\rho(arg_1, \dots, arg_n)$ , where the arguments are descriptions or values invariant with respect to the interpretation (e.g., numbers). Then

$$\begin{aligned} \#(i, \rho(arg_1, \dots, arg_n))^{\mathfrak{J}_G} &= (Top_i \sqcap \rho(\#(i, arg_1), \dots, \#(i, arg_n)))^{\mathfrak{J}_G} \\ &= \Delta^{\mathcal{I}^i} \cap f_\rho(\#(i, arg_1)^{\mathfrak{J}_G}, \dots, \#(i, arg_n)^{\mathfrak{J}_G}, \Delta^{\mathfrak{J}_G}) \\ &= \Delta^{\mathcal{I}^i} \cap f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathfrak{J}_G}) \end{aligned}$$

On the other hand,  $\rho(arg_1, \dots, arg_n)^{\mathcal{I}^i} = f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathcal{I}^i})$ . By letting  $W$  be  $\Delta^{\mathcal{I}^i}$ ,  $B_i$  be  $arg_i^{\mathcal{I}^i}$ , and  $\Delta$  be  $\Delta^{\mathfrak{J}_G}$  in the definition of local constructor, we have that  $\Delta^{\mathcal{I}^i} \cap f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathfrak{J}_G}) = f_\rho(arg_1^{\mathcal{I}^i}, \dots, arg_n^{\mathcal{I}^i}, \Delta^{\mathcal{I}^i})$ , which is equal to  $\rho(arg_1, \dots, arg_n)^{\mathcal{I}^i}$ . The same argument, but simplified by the absence of set intersection, holds for role constructors  $\rho$ .

Now suppose we are given that  $\#(\mathfrak{T}) \models \#(i, X) \sqsubseteq \#(i, Y)$ , and that  $\mathfrak{J} = \langle \{\mathcal{I}^i\}, \{\mathbf{r}_{ij}\} \rangle$  is a d-interpretation satisfying  $\mathfrak{T}$ ; we must show  $\mathfrak{J} \models_d i : X \sqsubseteq i : Y$ . We start by showing that  $\mathfrak{J}_G \models \#(\mathfrak{T})$ . So consider the various items in the definition of  $\#(\mathfrak{T})$ . Since  $\#(i, M)^{\mathfrak{J}_G} = M^{\mathcal{I}^i}$ , then  $\#(i, V)^{\mathfrak{J}_G} \subseteq \#(i, W)^{\mathfrak{J}_G}$  if and only if  $V^{\mathcal{I}^i} \subseteq W^{\mathcal{I}^i}$ , and hence  $\mathfrak{J}_G \models \#(i, V) \sqsubseteq \#(i, W)$  iff  $\mathcal{I}^i \models V \sqsubseteq W$ . As a result,  $\mathfrak{J}_G$  satisfies the subsumptions in item 1.  $\mathfrak{J}_G$  satisfies items 2 and 3 because  $\mathfrak{J}$  satisfies the bridge rules of  $\mathfrak{T}$ . Axioms in item 6 are satisfied because the  $\mathcal{I}^i$  are interpretations, and therefore their domains  $\Delta^{\mathcal{I}^i}$  are nonempty, and in turn these are the interpretations of  $Top_i$ . The remaining items follow from the fact that each  $\mathcal{I}^i$  is a model of theory  $\mathbf{T}_i$ , with domain  $\Delta^{\mathcal{I}^i}$ . Therefore  $\mathfrak{J}_G$  satisfies  $\#(\mathfrak{T})$ , and hence  $\mathfrak{J}_G \models \#(i, X) \sqsubseteq \#(i, Y)$ . By using again the fact that  $V^{\mathcal{I}^i} = \#(i, V)^{\mathfrak{J}_G}$ , we therefore get  $\mathcal{I}^i \models X \sqsubseteq Y$ , and hence that  $\mathfrak{J} \models_d i : X \sqsubseteq i : Y$ .

The “only if” part of the proof assumes  $\mathfrak{T} \models_d i : X \sqsubseteq Y$ , and starts from an arbitrary interpretation  $\mathfrak{J}_G$  of  $\#(\mathfrak{T})$ , trying to show that  $\mathfrak{J}_G \models \#(i, X) \sqsubseteq \#(i, Y)$ . We must construct from  $\mathfrak{J}_G$  a d-interpretation  $\mathfrak{J} = \langle \{\mathcal{I}^i\}, \{\mathbf{r}_{ij}\} \rangle$  that d-satisfies

$\mathfrak{T}$ . Let  $\Delta^{\mathcal{I}^i}$  be  $\#(i, \text{ANYTHING})^{\mathcal{J}_G}$ , and define the mappings  $\cdot^{\mathcal{I}^i} : \mathcal{DL}_i \rightarrow \Delta^{\mathcal{I}^i}$  to be  $M^{\mathcal{I}^i} = \#(i, M)^{\mathcal{J}_G}$  for atomic concepts and roles  $M$ .

First, we need to show that  $(\Delta_i, \cdot^{\mathcal{I}^i})$  is indeed an interpretation of  $\mathcal{DL}_i$ . This requires verifying that

- $\Delta^{\mathcal{I}^i}$  is a non-empty set; (\* true, because otherwise axiom 6 of  $\#(\mathfrak{T})$  cannot be satisfied by  $\mathcal{J}_G$  \*)
- $\text{ANYTHING}^{\mathcal{I}^i} = \Delta^{\mathcal{I}^i}$  (\* By the definition of  $\Delta^{\mathcal{I}^i}$  \*)
- $\text{NOTHING}^{\mathcal{I}^i} = \emptyset$  (\* By axiom 4 in the definition of  $\#(\mathfrak{T})$  \*)
- $A^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{I}^i}$  for atomic concepts  $A$  (\* By axioms 5 \*)
- $p^{\mathcal{I}^i} \subseteq \Delta^{\mathcal{I}^i} \times \Delta^{\mathcal{I}^i}$  for atomic roles  $p$ . (\* By axioms 7 \*)

Then one proves that  $\#(i, A)^{\mathcal{J}_G} = A^{\mathcal{I}^i}$  using the same argument as in the “if” case, and hence that  $\mathcal{I}^i \models \mathbf{T}_i$ , because of the presence of items 1 in the translation  $\#(\mathfrak{T})$ .

The relation  $\mathbf{r}_{ij}$  is defined by restricting the interpretation of  $R_{ij}$  to the parts of interest:  $R_{ij}^{\mathcal{J}_G} \cap (\#(i, \text{ANYTHING})^{\mathcal{J}_G} \times \#(j, \text{ANYTHING})^{\mathcal{J}_G})$ . Then the axioms in items 2 and 3 of the translation  $\#(\mathfrak{T})$  ensure that the bridge rules of  $\mathfrak{T}$  are also satisfied. Hence  $\mathcal{J} \models_d \mathfrak{T}$  as desired, and therefore  $\mathcal{J} \models_d i : X \sqsubseteq Y$  by assumption. Making use again of the equivalence  $\#(i, A)^{\mathcal{J}_G} = A^{\mathcal{I}^i}$ , we get the desired  $\mathcal{J}_G \models \#(i, X) \sqsubseteq \#(i, Y)$ .  $\square$

The following corollary uses the properties of the abstract constructor  $\rho$  to describe a collection of familiar concept and role constructors which allow the above translation of DDL reasoning to DL reasoning to go through.

**Corollary 1.** *The equivalence  $\#(\mathfrak{T}) \models \#(i, X) \sqsubseteq \#(i, Y)$  if and only if  $\mathfrak{T} \models_d i : X \sqsubseteq Y$  holds when  $\mathcal{DL}_i$  offers a subset of concept constructors  $\{C \sqcap D, C \sqcup D, \neg C, \forall p.C, \geq n pD, \leq n pD, \langle \text{role1} \rangle = \langle \text{role2} \rangle\}$  and role constructors  $\{\langle \text{role} \rangle^-, \sqcap, \sqcup, \circ\}$*

*Proof* First, we provide for each constructor the definition of the corresponding semantic function.

Concept constructor	Semantic function
$C_1 \sqcap C_2$	$f_{and}(XC1, XC2, DY) = X1 \cap X2$
$C_1 \sqcup C_2$	$f_{or}(XC1, XC2, DY) = X1 \cup X2$
$\neg C$	$f_{not}(XC, DY) = \{d \in DY \mid d \notin XC\}$
$\forall R.C$	$f_{forall}(XR, XC, DY) = \{d \in DY \mid XR(d) \subseteq XC\}$
$\exists R.C$	$f_{exists}(XR, XC, DY) = \{d \in DY \mid XR(d) \cap XC \neq \emptyset\}$
$\geq n R.C$	$f_{qatleast}(n, XR, XC, DY) = \{d \in DY \mid  XR(d) \cap C  \geq n\}$
$\leq n R.C$	$f_{qatmost}(n, XR, XC, DY) = \{d \in DY \mid  XR(d) \cap C  \leq n\}$
$R_1 = R_2$	$f_{sameas}(XR_1, XR_2, DY) = \{d \in DY \mid XR_1(d) = XR_2(d)\}$
Role constructor	Semantic function
$R^-$	$f_{inverse}(XR, DY) = \{(y, x) \mid (x, y) \in XR\}$
$R_1 \circ R_2$	$f_{compose}(XR_1, XR_2, DY) = \{(x, z) \mid y \in XR_1(XR_2(x))\}$
$R_1 \sqcap R_2$	$f_{role\_and}(XR_1, XR_2, DY) = XR_1 \cap XR_2$
$R_1 \sqcup R_2$	$f_{role\_or}(XR_1, XR_2, DY) = XR_1 \cup XR_2$



It is straightforward to demonstrate that each constructor is local, once one notes that DY is not used in the semantics of role constructors, and in the case of concept constructors, it appears only in the form  $\{d \in DY \mid \dots\}$ , which has the property that  $W \cap \{d \in \delta \mid \dots\} = \{d \in W \mid \dots\}$  if  $W \subseteq \delta$ .  $\square$

Since in the construction of  $\#(\mathfrak{T})$ , roles  $R_{ij}$  occur only in descriptions of the form  $\exists R_{ij}^-. \alpha$ , we can replace each  $R_{ij}^-$  by a new role identifier  $S_{ij}$ , thus making the role inverse constructor unnecessary in the theory  $\#(\mathfrak{T})$ . Therefore the above corollary has a consequence that DDLs with  $\mathcal{DL}_i$  that are in  $\mathcal{ALCN}\mathcal{R}$  [9],  $\mathcal{DLR}$  [10], or  $\mathcal{SHIQ}$  [19] can use their reasoners for determining  $\mathfrak{T} \models_d i : \alpha \sqsubseteq \beta$  by operating with the theory  $\#(\mathfrak{T})$ .

The following are some examples of concept and role constructors that are not local:

- Role complement, the identity role, or any role constructor whose semantic function cannot be written without using DY.
- A hypothetical concept constructor **each**(R), which denotes objects that are related by role R to every concept in the domain of interpretation. Such a constructor would have  $f_{each}(XR, DY) = \{d \in DY \mid XR(d) = DY\}$ , and  $W \cap f_{each}(V, \Delta) \neq f_{each}(V, W)$  because of the second DY in the body.

The above construction and theorem can be extended to deal with individuals (and hence A-boxes) by (a) adding to the language  $\mathcal{GDL}$  individual names  $i:ind$ , corresponding to individuals appearing in  $\mathcal{DL}_i$ ; (b) extending  $\#$  so that  $\#(i, ind) = i:ind$ ; and (c) adding the axioms  $Top_i(i:ind)$ , indicating that individuals are restricted to be interpreted in the sub-domain engendered by the representation of each local top concept. Moreover, the corollary can be shown to apply to constructors involving individuals such as **fills** and **one\_of**:

Syntax	Semantic function
$R(ind_1, \dots, ind_n)$	$f_{fills}(XR, \{v_1, \dots, v_n\}, DY) = \{d \in DY \mid \{v_1, \dots, v_n\} \subseteq XR(d)\}$
$\{ind_1, \dots, ind_n\}$	$f_{oneof}(v_1, \dots, v_n) = \{v_1, \dots, v_n\}$

## 8 Atomic DDL

In many practical applications on the web, an ontology is simply a list of words organized in a generalization hierarchy. Such a scheme is visible in the topic areas of search sites such DMOZ, Google, and Yahoo!. Taxonomies of terms are also widely used in e-commerce sites, such as Eclass ([www.eclass-standard.net/](http://www.eclass-standard.net/)) and UNSPSC ([www.unspsc.com](http://www.unspsc.com)). Such “lightweight” ontologies correspond to DLs where there are no concept and role constructors, only atomic identifiers. Hence there is no opportunity to define complex descriptions, and the T-box describes essentially a hierarchy of identifiers.

Although the computation of subsumption reduces to computing transitive closure for such T-boxes, there are several additional services that can be provided by actual systems:

- They can present visually a reduced partial order for IS-A hierarchies, where there are no redundant IS-A links: If  $A \sqsubseteq B$  and  $B \sqsubseteq C$  then there is no *explicit* IS-A link between  $A$  and  $C$ .
- Subsumption questions  $X \sqsubseteq Y$  can be answered in constant time for tree hierarchies, and still quite efficiently for hierarchies that are “almost” like trees, by special algorithms that preprocess the final hierarchy [1,6].

There is an obvious need to integrate even such simple ontologies, if one wants to provide access to a variety of related resources. For example, [25] show how the integration of the ontologies of two software repositories, Tucows and Downloads.com, can provide seamless retrieval of software from multiple sources. However, such ontologies, especially in e-commerce, would be expected to change daily, and would definitely not want to be merged into a single monolithic terminology. Therefore our approach of peer-to-peer coupling of IS seems reasonable in this case. As in other work, for example [20], the process starts by expressing statements that link terms in the two ontologies, usually adding subsumption relationships holding between terms in the two ontologies. One can then deduce the relationships of two terms by reasoning in the T-box which is the union of the two original ontologies, and these “cross-relating” subsumptions. The framework of DDLs provides a more refined tool, through the use of into- and onto-bridge rules. The question in this case is how one can reason effectively with the resulting DDL. The natural course is to use Theorem 1 to transform the problem into ordinary DL reasoning in the global DL. The most unsettling aspect of this approach is that even if the original  $\mathcal{DL}_i$  only had atomic concepts, the theory  $\#(\mathfrak{T})$  uses qualified existential quantifiers  $(\exists p.C)$ , for which it is known that subsumption is computationally difficult in the presence of general axioms. This is in contrast with the situation for atomic hierarchies, mentioned above.

There is however some hope for our reasoning in  $\#(\mathfrak{T})$ : if one looks carefully, the translation does not contain any *nested* existential restrictions. So let us focus on a simple DTB  $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$ , and define the language  $\mathcal{GDL}_{atomic}$  to allow only descriptions of the form  $1:B, 2:E$  or  $2:\exists R_{12}.B$ , with  $B$  an atomic concept of  $\mathbf{T}_1$ , and  $E$  an atomic concept of  $\mathbf{T}_2$ . For simplicity, we will drop explicit labels, and henceforth concepts  $A$  and  $B$  will be assumed to have label 1, while all other concepts will be assumed to have label 2. Moreover, we shall abbreviate descriptions of the form  $\exists R_{12}.B$  as  $\exists B$ .

**Definition 8.** Let  $\Gamma$  be a set of subsumption axioms involving descriptions in  $\mathcal{GDL}_{atomic}$ . A derivation/proof from  $\Gamma$  is a sequence of descriptions  $\alpha_1, \dots, \alpha_n$  ( $1 \leq n$ ) such that for each  $i < n$ , the relation  $\alpha_i \preceq \alpha_{i+1}$ , defined by the three inference rules below, holds

$$\begin{array}{ll} \alpha \preceq \beta & \text{if } \alpha \sqsubseteq \beta \text{ is an axiom in } \Gamma \\ \exists A \preceq \exists B & \text{if } A \sqsubseteq B \text{ is an axiom in } \Gamma \\ \exists Bot_1 \preceq Bot_2 & \end{array}$$

If  $\alpha_1, \dots, \alpha_n$  is a derivation from  $\Gamma$ , we shall write  $\Gamma \vdash \alpha_1 \preceq \alpha_n$ ,<sup>8</sup> or sometime  $\Gamma \vdash \alpha_1 \preceq \dots \preceq \alpha_n$ .

<sup>8</sup> The case  $n = 1$  yields the proof  $\Gamma \vdash \alpha \preceq \alpha$  for any description  $\alpha$ .

Next, define “proof theoretic” consistency

**Definition 9.** A DTB  $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$  is 1-consistent if there is no derivation  $\#(\mathfrak{T}_{12}) \vdash Top_1 \preceq Bot_1$ , and is 2-consistent if there is no derivation  $\#(\mathfrak{T}_{12}) \vdash Top_2 \preceq Bot_2$ .  $\mathfrak{T}_{12}$  will be said to be consistent if it is both 1-consistent and 2-consistent.

By extension,  $\mathbf{T}_1$  will be said to be consistent if  $\langle \mathbf{T}_1, \emptyset, \emptyset \rangle$  is (1-)consistent.

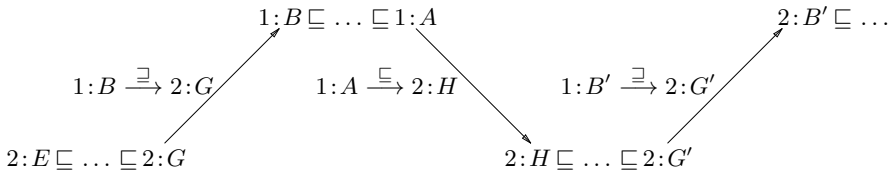
In order to simplify the proof of the next theorem, we will modify  $\#(\mathfrak{T}_{12})$  by adding to it some axioms that would normally be inferred by additional inference rules. First, replace axioms of type 4 with “bottom” axioms  $Bot_1 \sqsubseteq A$ ,  $Bot_2 \sqsubseteq E$  and  $Bot_2 \sqsubseteq \exists A$ . Second, replace axioms of type 5 with “top” axioms  $A \sqsubseteq Top_1$ ,  $E \sqsubseteq Top_2$ , and  $\exists A \sqsubseteq Top_2$ . Since our original theories are atomic, there are no axioms of type 7. Suppose we also eliminate axioms of type 6, so that  $\#(\mathfrak{T}_{12})$  now only has axioms of four types:  $A \sqsubseteq B$ ,  $E \sqsubseteq F$ ,  $E \sqsubseteq \exists A$ , and  $\exists B \sqsubseteq F$ . Then it can be shown that the interpretations of the original  $\#(\mathfrak{T}_{12})$  correspond exactly to what we shall call *acceptable interpretations* of the modified  $\#(\mathfrak{T}_{12})$ : ones which assign the empty set to  $Bot_1$  and  $Bot_2$ , and non-empty sets to  $Top_1$  and  $Top_2$ .

We are now ready to state the main result

**Theorem 2.** Given a DTB  $\mathfrak{T}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$ , where  $\mathbf{T}_1$ ,  $\mathbf{T}_2$ , and  $\mathfrak{B}_{12}$  involve only atomic concepts, and such that  $\mathbf{T}_1$  is consistent. Then  $\mathfrak{T}_{12} \models_d 2:G \sqsubseteq H$  if and only if  $\#(\mathfrak{T}_{12}) \vdash G \preceq H$ .

The proof of this theorem has been relegated to the Appendix, since it does not provide any particular insight into DDL.

According to the above theorem, modulo inconsistencies, derivations  $g(\mathfrak{T}_{12}) \vdash 2:E \preceq 2:F$  in atomic theories are essentially chains of subsumptions in  $\mathbf{T}_2$ , interspersed with “switches up” to  $IS_1$  via onto-rules, and then switches back down via into-rules as shown in Figure 6. In fact, by adding into-bridge rule



**Fig. 6.** Intuitive shape of subsumption proofs in atomic DDL

$1:\text{NOTHING} \xrightarrow{\sqsubseteq} 2:\text{NOTHING}$  to  $\mathfrak{B}_{12}$ , and by adding to  $\#(\mathfrak{T}_{12})$  all axioms  $\exists A \sqsubseteq \exists B$  whenever  $A \sqsubseteq B$  is in  $\mathbf{T}_1$ , we can eliminate the need for any additional inference rules, and we can treat expressions of the form  $\exists C$  as atomic. Therefore the cost of reasoning about subsumptions  $2:E \sqsubseteq F$  in the DDL  $\mathfrak{T}_{12}$  is just the cost of reasoning with atomic concepts in the modified atomic theory  $\#(\mathfrak{T}_{12})$ , whose size is linear in the size of  $\mathfrak{T}_{12}$ .

In the case when the number of bridge rules is much smaller than the size of  $\mathbf{T}_1$ , it may be worthwhile pre-computing the “up/down switches”. In particular, collect all onto bridge rules  $B_i \xrightarrow{\sqsubseteq} G_i$  and into bridge rules  $A_j \xrightarrow{\sqsubseteq} H_j$  such that  $\mathbf{T}_1 \models B_i \sqsubseteq A_j$ , and incorporate the corresponding subsumption  $G_i \sqsubseteq H_j$  in  $\mathbf{T}_2$  to obtain  $\mathbf{T}'_2$ . Assuming that subsumption computation is fast in  $\mathbf{T}_1$ , finding such additions is a quick process. Moreover, checking whether  $\mathfrak{T}_{12} \models_d 2:E \sqsubseteq F$  now reduces to checking  $\mathbf{T}'_2 \models E \sqsubseteq F$ .

As an aside, note that if in one of the above cases  $\vdash H_j \sqsubseteq G_i$  also holds, then indeed  $H_j$  and  $G_i$  are equivalent *as far as  $IS_2$  sees it in the DDL*; but this “bow-tie” situation [31] does **not** imply in DDL that the corresponding  $A_j$  and  $B_i$  must also be equivalent, or that there is an inconsistency. This is the advantage of allowing for directed mappings to connect the domains of the two ISs.

## 9 Conclusions

The seminal work of Catarci and Lenzerini [13] on integrating ISs described by DLs, and most of the work following it, made an implicit assumption that the local ISs have the same notion of what individual objects are, and that there was only one set of (subsumption) assertions relating  $IS_1$  and  $IS_2$ . We have argued that in cases such as loosely coordinated IS, when there is no single global view, these conditions need to be relaxed, by allowing general correspondence relationships between objects in the local domains, and by having “directed” import assertions. Moreover, in contrast to earlier approaches, our work on DDL extends the *reasoning* available on ordinary schemas to the case of multiple schemas aligned by binary correspondences between individuals. Such reasoning can be used for a wide variety of tasks, including inconsistency detection, query (re)formulation, and dealing with partially-specified individuals, as surveyed in [7]. Stuckenschmidt [32] provides additional evidence that this approach has some potential benefits by formalizing the co-ordination of local ontologies for agents using DDL.

This paper has also identified other desirable properties of aligned DL, such as “no backflow”, and localizing the effect of inconsistencies so that one IS does not “infect” the reasoning of the entire system.

Among the interesting results obtained are a translation scheme of DDL reasoning to DL reasoning, which is applicable to a large class of description logics. The representation of the target DL knowledge base requires only the presence of the qualified existential restriction DL-constructor  $\exists p.C$  and the ability to reason with (acyclic) background axioms, over and above the needs of the local ISs. This allows both complexity results and reasoning algorithms to be transferred for DDL whose local theories come from decidable logics such as *SHIQ* and *DLR*.

Finally, we have taken a closer look at the case of DDL without concept constructors, i.e., ones in which there are only atomic concepts organized in a subsumption hierarchy. These are of interest since they correspond to a large class of ontologies currently used on the web. In this case, we showed that the

reasoning needed to compute d-entailment is no different from that needed to compute regular entailment, and can be accomplished by the addition of a relatively small number of pre-computed links to the hierarchy of the ontology that “imports” information.

Among the many open problems remaining, we mention the investigation of the behaviour of DDL with more than two local T-boxes, both in the presence and absence of cycles in the domain relations; the introduction of a greater variety of bridge rules, especially ones that are of both practical interest and can preserve the very useful DDL-to-DL translation theorem; and studying approaches to localizing the effect of inconsistencies.

**Acknowledgements.** We wish to thank the anonymous referees for their comments and constructive suggestions, which we have hopefully been able to follow to improve the paper.

## References

1. R. Agrawal, A. Borgida, H. V. Jagadish. Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. *Proc. SIGMOD 1989*: 253–262
2. Y. Arens, C.A. Knoblock, W.M. Shen. Query Reformulation for Dynamic Information Integration. *J. Intelligent Information Systems* 6(2/3): 99–130 (1996)
3. F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider editors, *The Description Logic Handbook*, Cambridge University Press, January 2003.
4. T. Berners-Lee. What the Semantic Web can represent. September 1998.  
<http://www.w3.org/DesignIssues/RDFnot.html>
5. S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering* 36(3): pp. 215–249 (2001).
6. M. F. van Bommel, and T. J. Beck: Incremental Encoding of Multiple Inheritance Hierarchies. *Proc. CIKM 1999*: pp. 507–513.
7. A. Borgida. Description Logics In Data Management. *IEEE Trans. on Knowledge and Data Engineering* 7(5), pp.671–682 (1995).
8. A. Borgida, M. Lenzerini, and R. Rosatti. Description Logics for Databases. Chapter 16 in [3].
9. M. Buchheit, F. M. Donini and A. Schaerf. Decidable Reasoning in Terminological Knowledge Representation Systems. *J. Artificial Intelligence Research* 1, pp. 109–138 (1993).
10. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. KR’98*, pp. 2–13, 1998.
11. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. *Proc. CoopIS’98*, pp.280–291, 1998.
12. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems* 10(3), pp. 237–271 (2001).

13. T. Catarci and M. Lenzerini: Representing and using interschema knowledge in cooperative information systems. *International Journal of Intelligent and Cooperative Information Systems* 2(4), pp.375–398 (1993).
14. D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 27(3), pp.59–74, September 1998.
15. C. Ghidini and L. Serafini. Distributed First Order Logics. In D. Gabbay and M. de Rijke, editors, *Frontiers Of Combining Systems 2*, Studies in Logic and Computation, pp. 121–140. Research Studies Press, 1998.
16. J. Hammer and D. McLeod. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. *International Journal of Intelligent & Cooperative Information Systems* 2(1), pp.51–83 (1993).
17. D. Heimbigner and D. McLeod: A Federated Architecture for Information Management. *ACM TOIS* 3(3), pp. 253–278 (1985)
18. I. Horrocks, P. Patel Schneider, and F. van Harmelen. Reviewing the design of DAML+OIL: language for the semantic web. In *Proc. AAAI’02*, 2002.
19. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. Logic and Computation*, 9(3), pp.385–410 (1999).
20. V. Kashyap and A. Sheth. Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach. *VLDB Journal* 5(4), pp. 276–304 (1996).
21. W. Kent. Solving Domain Mismatch and Schema Mismatch Problems with an Object-Oriented Database Programming Language. *Proc. VLDB’91*, pp. 147–160, 1991.
22. M. Klein. Combining and relating ontologies: an analysis of problems and solutions. *Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing* Seattle, USA, 2001.
23. L. Lakshmanan, F. Sadri, and I.N. Subramanian. Logic and Algebraic Languages for Interoperability in Multi-database Systems. *Journal of Logic Programming* 33(2), pp. 101–149 (1997).
24. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Query answering algorithms for information agents *Proc. AAAI’96*
25. E. Mena, A. Illarramendi, and A. Goni: Automatic Ontology Construction for a Multiagent-Based Software Gathering Service, in *Proc. Workshop on Cooperative Information Agents IV*, Springer LNCS 1860, pp. 232–243, 2000.
26. E. Mena, V. Kashyap, A. Illarramendi and A. Sheth. Imprecise Answers on Highly Open and Distributed Environments: An Approach based on Information Loss for Multi-Ontology Based Query Processing. *Int. Journal of Cooperative Information Systems* 9(4), pp.403–425 (2000).
27. A. Pirotte, E. Zimanyi, D. Massart, and T. Yakusheva. Materialization: A Powerful and Ubiquitous Abstraction Pattern. In *Proc. VLDB’94*, pp. 630–641, 1994.
28. E. Rahm, and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal* 10(4), pp.334–350 (2001).
29. F. Saltor and E. Rodríguez. On Intelligent Access to Heterogeneous Information. *Proc. KRDB’97*, Athens, Greece, August 30, 1997.
30. E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM TODS* 19(2):254–290, June 1994.
31. J. Sowa. *Building, Sharing, and Merging Ontologies*. <http://www.jfsowa.com/ontology/ontoschar.htm>
32. H. Stuckenschmidt. Exploiting Partially Shared Ontologies for Multi Agent Communication. *Proc. Cooperative Information Agents VI (CIA 2002)*, Springer LNCS 2446.

## Appendix: Proof of Theorem 2

First, given a DTB  $\mathfrak{I}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$ , recall the convention that symbols  $A, B$  and  $C$  stand for atomic concepts in  $\mathbf{T}_1$  including  $Top_1$  and  $Bot_1$ , while  $E, F, G$  and  $H$  are atomic concepts in  $\mathbf{T}_2$ , including  $Top_2$  and  $Bot_2$ ; and  $\exists A$  is an abbreviation of  $\exists R_{12}^- . A$ . Let us repeat here the inference rules used in the definition of “derivation”:

Rule	Rule Name
$\alpha \preceq \beta$ , if $\alpha \sqsubseteq \beta$ is an axiom in $\Gamma$	(“axiom”)
$\exists A \preceq \exists B$ , if $A \sqsubseteq B$ is an axiom in $\Gamma$	(“some”)
$\exists Bot_1 \preceq Bot_2$	(“some-bottom”)

And we restate the desired theorem:

**Theorem 3.** *Given a DTB  $\mathfrak{I}_{12} = \langle \mathbf{T}_1, \mathbf{T}_2, \mathfrak{B}_{12} \rangle$ , where  $\mathbf{T}_1$ ,  $\mathbf{T}_2$ , and  $\mathfrak{B}_{12}$  involve only atomic concepts, and which is 1-consistent. Then  $\mathfrak{I}_{12} \models_d 2:G \sqsubseteq H$  if and only if  $\#(\mathfrak{I}_{12}) \vdash G \preceq H$ .*

By Theorem 1, we know that  $\mathfrak{I}_{12} \models_d 2:G \sqsubseteq H$  iff  $\#(\mathfrak{I}) \models \#(2, G) \sqsubseteq \#(2, H)$ . The rules of inference are evidently sound: if  $\mathcal{I} \models \#(\mathfrak{I})$ , and  $\#(\mathfrak{I}) \vdash \alpha \preceq \beta$  then  $\mathcal{I} \models \alpha \sqsubseteq \beta$ . As usual, the more complex part is to prove the completeness of the rules: if  $\#(\mathfrak{I}_{12}) \models G \sqsubseteq H$  then  $\#(\mathfrak{I}) \vdash G \preceq H$ . This is shown by proving the contrapositive: assume that there is no proof  $\#(\mathfrak{I}) \vdash G \preceq H$  (written as  $\#(\mathfrak{I}) \not\vdash G \preceq H$ ), and then construct an interpretation  $\mathfrak{J}_0$  such that  $\mathfrak{J}_0 \models \#(\mathfrak{I})$  yet  $G^{\mathfrak{J}_0}$  is not a subset of  $H^{\mathfrak{J}_0}$ .

The construction of  $\mathfrak{J}_0$  is based on the idea of interpreting an atomic concept  $M$  as the empty set if  $M \preceq Bot_i$  can be proven. Concepts  $E$  for which there is no proof  $E \preceq Bot_2$  will be interpreted as  $Top_2$ . Only concepts  $A$  for which there is no proof  $A \preceq Bot_1$  will require more careful handling, in order to interpret descriptions  $\exists A$  appropriately.

To be precise, let us assume that there is no proof  $\#(\mathfrak{I}) \vdash G \preceq H$ , and let  $\Sigma = \#(\mathfrak{I}) \cup \{Top_2 \sqsubseteq G, H \sqsubseteq Bot_2\}$ . Note that  $G$  and  $H$  cannot be identical because otherwise we have, by definition of  $\vdash$ ,  $\#(\mathfrak{I}) \vdash G \preceq G$ , which contradicts our assumption.

Let  $\Delta_0 = \{a, b, g\}$  be a set of three objects. Define the mapping  $\cdot^{\mathfrak{J}_0}$  as follows:

- $E^{\mathfrak{J}_0} = \emptyset$  if  $\Sigma \vdash E \preceq Bot_2$ ;
- $E^{\mathfrak{J}_0} = \{g\}$  if  $\Sigma \not\vdash E \preceq Bot_2$ ;
- $A^{\mathfrak{J}_0} = \emptyset$  if  $\Sigma \vdash A \preceq Bot_1$ ;
- $A^{\mathfrak{J}_0} = \{a\}$  if  $\Sigma \not\vdash A \preceq Bot_1$ , but  $\Sigma \vdash \exists A \preceq Bot_1$ ;
- $A^{\mathfrak{J}_0} = \{a, b\}$  if both  $\Sigma \not\vdash A \preceq Bot_1$ , and  $\Sigma \not\vdash \exists A \preceq Bot_1$ ;
- $R_{12}^{\mathfrak{J}_0} = \{(b, g)\}$ , so that  $R_{12}^{-\mathfrak{J}_0} = \{(g, b)\}$ .

We claim that  $\mathfrak{J}_0$  is an “acceptable interpretation”. First,  $Bot_i^{\mathfrak{J}_0} = \emptyset$  for  $i = 1, 2$  because we have  $\Sigma \vdash Bot_i \preceq Bot_i$  by “bottom” axioms in  $\#(\mathfrak{I})$ . Second,  $Top_1^{\mathfrak{J}_0}$  and  $Top_2^{\mathfrak{J}_0}$  are not empty because we will prove below Lemma 1, showing that  $\Sigma$  is consistent. Moreover, once we establish that  $\mathfrak{J}_0 \models \Sigma$

(Lemma 2 below), then it only remains to note that because  $\Sigma$  is defined as  $\#(\mathfrak{T}) \cup \{Top_2 \sqsubseteq G, H \sqsubseteq Bot_2\}$ , then according to the definition of  $\mathfrak{J}_0$  and inference rule “axiom”,  $G^{\mathfrak{J}_0} = \{g\}$  and  $H^{\mathfrak{J}_0} = \emptyset$ , so that  $G^{\mathfrak{J}_0}$  is not a subset of  $H^{\mathfrak{J}_0}$ .

*Lemma 1.  $\Sigma$  is consistent.*

*Proof* First note that the rules of inference and the form of axioms in  $\#(\mathfrak{T})$  guarantee that if there is a derivation  $\Sigma \vdash A \preceq \alpha_1 \preceq \dots \preceq \alpha_n$ , then  $\alpha_i$  cannot be concepts of the form  $E$  or  $\exists B$ , and if  $n > 1$  then each step must be based on an axiom in  $\mathbf{T}_1$ . One consequence of the above observation is that if there is a derivation  $\Sigma \vdash A \preceq \dots \preceq \alpha_n$ , then since each step is based on subsumptions in  $\#(\mathbf{T}_1)$ , then one can repeatedly use inference rule “some” to obtain  $\Sigma \vdash \exists A \preceq \dots \preceq \exists \alpha_n$ . A second, more immediately relevant consequence is that if  $\Sigma \vdash Top_1 \preceq Bot_1$ , then we must have had  $\#(\mathbf{T}_1) \vdash Top_1 \preceq Bot_1$ , which contradicts the consistency assumption about  $\mathbf{T}_1$ . Therefore  $\Sigma$  is 1-consistent.

So suppose, by way of contradiction, that  $\Sigma$  is not 2-consistent:  $\Sigma \vdash Top_2 \preceq Bot_2$ . If  $\#(\mathfrak{T})$  itself was inconsistent, i.e.,  $\#(\mathfrak{T}) \vdash Top_2 \preceq Bot_2$ , then due to the presence in  $\#(\mathfrak{T})$  of “top” axiom  $G \sqsubseteq Top_2$ , and “bottom” axiom  $Bot_2 \sqsubseteq H$ , there would be a derivation  $\#(\mathfrak{T}) \vdash G \preceq H$ , contradicting the original assumption. Therefore  $\#(\mathfrak{T})$  must have been consistent, and only the use of one or both the axioms in  $\{Top_2 \sqsubseteq G, H \sqsubseteq Bot_2\}$  permits the deduction of the 2-inconsistency.

If the derivation  $\Sigma \vdash Top_2 \preceq Bot_2$  is of length 1, then either  $H = Top_2$  or  $G = Bot_2$ . In either case, “top” or “bottom” axioms in  $\#(\mathfrak{T})$  would allow the derivation of  $H \preceq G$  – a contradiction. Otherwise, the derivation must have had at least two steps, one or two of which would have been justified by  $Top_2 \sqsubseteq G$  and  $H \sqsubseteq Bot_2$ . (Note that by the nature of the rules of inference and the form of the axioms, repeated use of an axiom can be eliminated, arriving at a shorter proof.)

Case (i): If the proof is of the form  $\Sigma \vdash Top_2 \preceq \dots Top_2 \preceq G \dots \preceq Bot_2$ , then there must be a subproof  $\#(\mathfrak{T}) \vdash G \preceq Bot_2$ . But by the presence in  $\#(\mathfrak{T})$  of “bottom” axiom  $Bot_2 \sqsubseteq H$ , we then get  $\#(\mathfrak{T}) \vdash G \preceq H$  – a contradiction.

Case (ii): If the proof is of the form  $\Sigma \vdash Top_2 \preceq \dots H \preceq Bot_2 \dots \preceq Bot_2$ , then there must be subproof of  $\#(\mathfrak{T}) \vdash Top_2 \preceq H$ . But by the presence in  $\#(\mathfrak{T})$  of “top” axiom  $G \sqsubseteq Top_2$ , we then get  $\#(\mathfrak{T}) \vdash G \preceq H$  – a contradiction.

Case (iii): If the proof is of the form  $\Sigma \vdash Top_2 \preceq \dots Top_2 \preceq G \preceq \dots \preceq H \preceq Bot_2 \dots \preceq Bot_2$ , then either  $G = H$ , which we have said was impossible, or there must have been a subproof  $\#(\mathfrak{T}) \vdash G \preceq H$  – a contradiction.

Case (iv): If the proof is of the form  $Top_2 \preceq \dots H \preceq Bot_2 \dots \preceq Top_2 \preceq G \preceq \dots Bot_2$ , then we are in the same situation as case (ii) above.  $\square$

*Lemma 2.  $\mathfrak{J}_0 \models \Sigma$*

*Proof* Consider the 4 different kinds of axioms  $\alpha \sqsubseteq \beta$  in  $\Sigma$ ; in each case we will show that if  $\alpha^{\mathfrak{J}_0} \not\subseteq \beta^{\mathfrak{J}_0}$  then there is a contradiction.

(1)  $E \sqsubseteq F$  — If  $E^{\mathfrak{J}_0} \not\subseteq F^{\mathfrak{J}_0}$ , then we must have  $E^{\mathfrak{J}_0} \neq \emptyset$  and  $F^{\mathfrak{J}_0} = \emptyset$ ; the latter holds when  $\Sigma \vdash F \preceq Bot_2$ , and this, together with the presence of  $E \sqsubseteq F$  in



$\Sigma$ , provides a derivation  $\Sigma \vdash E \preceq Bot_2$ . On the other hand,  $E^{\mathcal{I}_0} \neq \emptyset$  only when  $\Sigma \not\vdash E \preceq Bot_2$  – a contradiction.

(2)  $A \sqsubseteq B$  — If  $A^{\mathcal{I}_0} \not\subseteq B^{\mathcal{I}_0}$ , then  $A^{\mathcal{I}_0} \not\subseteq \emptyset$ , which means that  $\Sigma \not\vdash A \preceq Bot_1$ . If  $B^{\mathcal{I}_0} = \emptyset$ , then we get a contradiction as in the previous case. Otherwise, we must have (i)  $\Sigma \vdash \exists B \preceq Bot_2$  but (ii)  $\Sigma \not\vdash \exists A \preceq Bot_2$ . In this case, from the assumption that  $A \sqsubseteq B$  is in  $\Sigma$ , we get, using inference rule “some”, that  $\Sigma \vdash \exists A \preceq \exists B$ , which together with (i) yields  $\Sigma \vdash \exists A \preceq Bot_2$ , contradicting (ii).

(3)  $\exists A \sqsubseteq E$  — Then  $E^{\mathcal{I}_0}$  must be empty, requiring  $\Sigma \vdash E \preceq Bot_2$ , which together with the presence of  $\exists A \sqsubseteq E$  in  $\Sigma$ , provides  $\Sigma \vdash \exists A \preceq Bot_2$ . However,  $(\exists A)^{\mathcal{I}_0} = (\exists R_{12}^- . A)^{\mathcal{I}_0}$  must not be empty, which means that  $A^{\mathcal{I}_0}$  must contain  $b$ , by the definition of the interpretation of the role  $R_{12}^-$ . But this requires that  $\Sigma \not\vdash \exists A \preceq Bot_2$  – a contradiction.

(4)  $F \sqsubseteq \exists B$  — Then  $F^{\mathcal{I}_0} \neq \emptyset$  while  $\exists B^{\mathcal{I}_0} = \emptyset$ , which requires either that  $B^{\mathcal{I}_0} = \emptyset$ , occurring when  $\Sigma \vdash B \preceq Bot_1$ , or  $B^{\mathcal{I}_0} = \{a\}$ , occurring when  $\Sigma \vdash B \preceq Bot_2$ . In the former case,  $\Sigma \vdash \exists B \preceq \exists Bot_1$ , using inference rule “some”, which also leads to  $\Sigma \vdash \exists B \preceq Bot_2$  using inference rule “some-bottom”. Thus, in either case we get, using the fact that  $F \sqsubseteq \exists B$  is in  $\Sigma$ , that  $\Sigma \vdash F \preceq Bot_2$ , contradicting the requirement that  $F^{\mathcal{I}_0} \neq \emptyset$ .  $\square$

# On Using Conceptual Data Modeling for Ontology Engineering

Mustafa Jarrar, Jan Demey, and Robert Meersman

STARLab Vrije Universiteit Brussel, Pleinlaan 2,  
Brussels, 1050, Belgium  
{mjarrar, jdemey, meersman}@vub.ac.be

**Abstract.** This paper tackles two main disparities between conceptual data schemes and ontologies, which should be taken into account when (re)using conceptual data modeling techniques for building ontologies. Firstly, conceptual schemes are intended to be used during design phases and not at the run-time of applications, while ontologies are typically used and accessed at run-time. To handle this first difference, we define a conceptual markup language (ORM-ML) that allows to represent ORM conceptual diagrams in an open, textual syntax, so that ORM schemes can be shared, exchanged, and processed at the run-time of autonomous applications. Secondly, unlike ontologies that are supposed to hold application-independent domain knowledge, conceptual schemes were developed only for the use of an enterprise application(s), i.e. “in-house” usage. Hence, we present an ontology engineering-framework that enables reusing conceptual modeling approaches in modeling and representing ontologies. In this approach we prevent application-specific knowledge to enter or to be mixed with domain knowledge. To end, we present DogmaModeler: an ontology-engineering tool that implements the ideas presented in the paper.

**Keywords:** Ontology, Conceptual data modeling, Context, Ontology tools, Reusability, DOGMA, DogmaModeler, ORM, ORM-ML.

## 1 Introduction and Motivation

Successful conceptual data modeling approaches, such as ORM or EER, became well known because of their methodological guidance in building conceptual models of information systems. They are semantically rich disciplines and support quality checks at a high level of abstraction [V82]; they provide conceptual constructs like integrity, taxonomy, and derivation rules [H01] [F02]. Merely, conceptual data schemes -also called semantic data models- were developed to capture the meaning of an application domain as perceived by its developers [WSW99] [M99]. Nevertheless, this meaning is being represented in diagram formats, and typically used in an *off-time mode*, i.e. used during the design phases and not at the run-time of applications.

Nowadays, the Internet and the open connectivity environments create a strong demand for sharing and exchanging not only data but also data semantics. Therefore, emerging ontologies are intended to represent agreed and shared domain semantics. So that, based on such ontologies, computer systems can meaningfully communicate to exchange data and make transactions interoperate independently of their internal technologies. For example, by sharing an ontology, heterogeneous information resources can be integrated and searched through mediators [TSC01] [SOVZJSSM02], e-commerce applications can meaningfully communicate, etc.

Conceptual data schemes and ontologies are quite similar, as both consist of conceptual relations<sup>1</sup> and rules<sup>2</sup>. Thus, several researchers have proposed to (re) use those conceptual methodologies and tools for ontology modeling, e.g. [F02] [CHP01] [BKKHSHLA01] [M01]. Reusing conceptual modeling techniques for ontology engineering is ultimately beneficial: the large set of existing conceptual modeling methods, graphical notations, and tools can make ontologies better understandable, and easier to adopt, construct, visualize, etc. Furthermore, legacy conceptual schemes can be mined and/or “ontologized”.

For such purposes, some extensions have been made to the foundation of the conceptual modeling constructs. For example, to effectively capture knowledge about application domains, [WSW99] redefined several conceptual modeling constructs of the ER approach, by defining some rules –based on ontological analyses– to resolve ambiguities that exist in current practice of modeling relationships. [GHW02] suggested *ontological semantics* for UML class diagrams, by rooting the UML constructs to the GOL upper level ontology. [BCD01][BB03][MC02] have shown how to reason about conceptual schemes.

However, complementary to these efforts, there are two main disparities between ontologies and conceptual data schemes that we aim to tackle in this paper:

1) Conceptual data schemes are being preserved in off-time mode diagrams; while, ontologies typically are *shareable and exchangeable at run-time*, i.e. machine-processable semantics.

2) Unlike conceptual data schemes that capture semantics for a *given application domain*, ontologies are supposed to capture semantics about real-world domains, independent from specific application needs, i.e. “*relatively*” *generic knowledge*. Therefore, *the genericity (/application-independency) of knowledge is a fundamental asset in ontology modeling, and that mostly distinguishes ontologies from conceptual data schemes*.

The first goal of this paper is to provide a way for conceptual diagrams to be exchanged and shared at run-time. Therefore, we *define a conceptual markup language in which one can textually represent ORM conceptual diagrams in an open*

---

<sup>1</sup> Conceptual relations can be unary relations (usually called “concepts” or object types as called in ORM), or n-ary relations, which also are called fact types in ORM.

<sup>2</sup> Rules, formally, are well-formed formulae; defined in an ontology (or a conceptual schema) in order to specify and constrain the intended models that can hold. In conceptual data modeling they are commonly called “constraints”. Notice that rules can be used for e.g. enforce integrity, derivation and inference, taxonomy, etc.

(*nonproprietary*) way. The second goal is to present an ontology engineering framework (and tool) that enables ORM conceptual schemes to be used for modeling and representing *ontological commitments*<sup>3</sup>. Please notice that our approach is not restricted to ORM. The same techniques can be applied to other conceptual modeling methods.

*Remark:* Let us now clarify some terminology that we frequently use in the paper. The intended use of the term 'task', is related and limited to the inferential knowledge that is required to describe a task to be performed, e.g. it does not describe temporal aspects. An application may convey one or more *kinds* of tasks. However the term task is often interchanged with the 'application' that conveys one kind of task. Notice that the '*reusability*' of knowledge implies the maximization of using this knowledge among *different kinds* of applications; while increasing the *usability* implies maximizing the number of applications among the *same kind* of task [JM02b]. Moreover, we use the term '*generic task*' to refer to a reusable kind of task.

**Structure of this paper:** In section 2, we give a bird's eye introduction to ORM. The ORM markup language (ORM-ML) will be presented in Section 3. In section 4, we discuss the knowledge independency and genericity issues, then we present our framework that uses ORM in ontology modeling. In section 5, we briefly present our DogmaModeler Tool for ontology engineering. Finally, in section 6, we conclude and present some future work.

## 2 ORM Background

Conceptual modeling methodologies are well developed and have proven to be quite successful for building information systems in a graphical way at the conceptual level. By representing the data on a higher level of abstraction conceptual models quality checks can be performed easily. ORM (Object-Role Modeling) [H01] is such a conceptual modeling approach that was developed in the early 70's. Originally it is a successor of NIAM (Natural-language Information Analysis Method) [VB82].

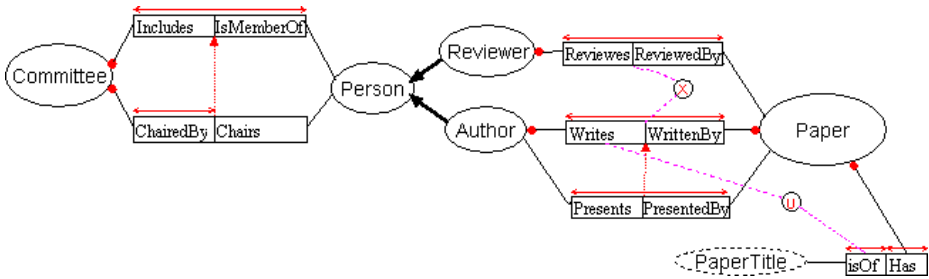
In ORM, the world is represented under the form of objects playing roles. There are two kinds of object types: lexical object types (LOTs) and non-lexical object types (NOLOTs). The distinction between LOTs and NOLOTs is a linguistic distinction. Lexical object types correspond to "utterable" entities (for instance: 'ISBN', 'Title', 'FistName'), while Non-lexical object types (for instance: Book, Person) refer to "non-utterable" entities [VB82]. The design process of information systems is simplified by using ORM, because ORM has an easy to understand graphical notation which includes most frequently used constraints.

Based on ORM, several conceptual modeling tools exist, such as Microsoft's VisioModeler™ and the older InfoModeler, which have the functionality of modeling a certain Universe of Discourse (UoD) in ORM, and support the automatic generation of a consistent and normalized relational database schema for a modeled UoD.

---

<sup>3</sup> See section 4 for the definition.

Also, ORM schema's can be translated into pseudo natural language statements. The graphical representation and the translation into pseudo natural language make it a lot easier, also for non-computer scientists, to create, check and adapt the knowledge about the UoD needed in an information system.



**Fig. 1.** An example of ORM schema

In Fig. 1, object types are shown as named ellipses. Logical predicates (fact types) appear as named sequences of roles, where each role appears as a box. Roles are connected by line segments to the object types that 'play' them.

In the Figure, the object types are Committee, Person, Reviewer, Author, Paper and PaperTitle. The dotted ellipse indicates that PaperTitle is a lexical object type while the others are non-lexical. The predicates in the figure can be verbalized as follows: 'A Paper Has a PaperTitle and a PaperTitle IsOf a Paper', 'An Author Reviews a Paper and a Paper is Reviewed by a Reviewer', 'An Author Writes a Paper and A Paper is Written By an Author', etc. The arrows connecting the object types Author and Reviewer to Person denote an is-a (predefined subtype) relationship.

In what follows, we briefly name and explain the constraints appearing in the diagram, in fact by giving an (approximate) verbalization of the example in Fig. 1. For other types of ORM constraints we refer to [VB82] or [W90]; the notation and definitions used here are taken from [H01].

Black dots indicate a mandatory role constraint. Example verbalization of Fig. 1: 'Each Paper must be WrittenBy at least one Author'. The arrow-tipped bars above the roles are uniqueness constraints. For example: 'Each Committee is ChairedBy at most one Person. Uniqueness constraints can span more than one role, indicating that 'any combination that instantiates these roles should be unique'. E.g. for the predicate (Author writes paper, Paper is WrittenBy author), there holds that every combination of author and paper is unique, so an author can only once be the author of the same paper. In ORM, one can also define constraints between different predicates or fact types. For instance, the circled 'X' (which stands for eXclusion constraint) between 'Reviewer Reviews/ReviewedBy Paper' and 'Author Writes/WrittenBy Paper' means that an Author (Person) who has written a certain Paper is not allowed to be a reviewer of the same paper. An arrow between two predicates indicates a subset constraint between the roles involved: 'Each Author who presents a Paper must have written that Paper'.

### 3 ORM Markup Language

As we noted in the introduction, conceptual modeling approaches were developed to assist system developers during the design phases. They were not meant to be accessed and processed at the run-time of applications. Besides, conceptual diagrams are typically saved in graphical formats, which are proprietary and therefore are limited to use inside specific CASE tools.

In this section we show how conceptual diagrams can be marked up and thus accessed and processed at run-time of applications. We illustrated this by defining a new open syntax markup language to represent ORM conceptual diagrams textually. ORM Markup Language is based on the XML syntax, and is defined in an XML-Schema [ORMML-XMLS] that acts as its complete and formal grammar, thus any ORM-ML file should be valid according to this XML-Schema.

ORM-ML is not meant to be written by hand or interpreted by humans. It is meant to be implemented as a “save as” or “export to” functionality in ORM tools. In section 6 we will present an ontology engineering tool that makes use of ORM-ML.

In what follows, we describe the main elements of the ORM-ML grammar and demonstrate it using a few elementary examples. A more complete example is provided in Appendix A. We chose to respect the ORM structure as much as possible by not “collapsing” it through the usual relational transformer that comes with most ORM-based tools. ORM-ML allows the representation of any ORM schema without loss of information or change in semantics, except for the geometry and topology (graphical layout) of the schema (e.g. location, shapes of the symbols), which we provide as a separate *graphical style sheet* to the ORM Schema.

We represent the ORM document as a one node element called ORMSchema, which consists itself of two nodes: ORMMeta and ORMBody. As a header to an ORM document, and to illustrate the “ORM Schema Document” (instance) nature of the described schema, ORMMeta node includes *meta data* about the ORM *document* using the 16 Dublin Core<sup>4</sup> Meta Tags [RFC2413]; an example of their use appears in Table 1 below.

**Table 1.** Example of an ORMMeta Node in an ORM-ML File

```

...<ORMMeta>
  <dc:Title>Bookstore</dc:title>
  <dc:Creator>Mustafa Jarrar</dc:creator>
  <dc:contributor>Jan Demey</dc:contributor>
  <dc:contributor>Robert Meersman</dc:contributor>
  <dc:Description>An example of ORM-ML </dc:description>
  <dc:Language>English</dc:language>.
.....
</ORMMeta>....

```

The ORMBody node consists of at most these five different kinds of (meta-ORM) elements: Object, Subtype, Predicate, Predicate\_Object and Constraint.

We adopt in the sequel the ORM modeling technique as defined in [H01]. Object elements are abstract XML elements and are used to represent Object Types. They are

<sup>4</sup> Dublin Core is a standard of 16 metadata tags that can be used for “annotating” any information object.

identified by an attribute ‘Name’ which is the name of the Object Type in the ORM Schema (see fig. 2 in Example 2). Objects might have some Value or ValueRange elements, which are used for value constraints on the Object Type (not present in Fig. 2). A ValueRange element has 2 attributes: begin and end, with obvious meanings. Object Types are implemented by two XML elements: LOT (Lexical Object Type, called Value Types in [H01]) and NOLOT (Non-Lexical Object Type, called Entity Types in [H01]). LOT elements may have a numeric attribute, which is a boolean and indicates whether we deal with a numeric Lexical Object Type. NOLOT elements have a boolean attribute called independent, which indicates whether the Non Lexical Object Type is independent. NOLOT elements may also have a reference element. A reference element would indicate how this NOLOT is identified by LOTs and other NOLOTs in a given application environment. A reference element has 2 attributes: ref\_name, the name of the reference and numeric, a boolean to indicate whether it is a numeric reference.

Example 2

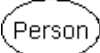
	... <Object xsi:type="NOLOT" Name="Person"/> ...
---	--

Fig. 2

Table 2. ORM-ML representation of Fig. 2

Subtype elements are used to represent subtype relationships between non-lexical object types. A subtype element is required to have two attributes: parent and child, which are references to object elements (see Example 3).

Example 3

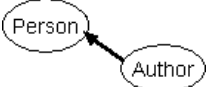
	... <Object xsi:type="NOLOT" Name="Person"/> <Object xsi:type="NOLOT" Name="Author"/> <Subtype Parent="Person" Child="Author"/> ...
---	---

Fig. 3.

Table 3. ORM-ML representation of Fig. 3

Predicates consist of at least one Object\_Role element. Such an element contains a reference to an object and may contain a role. They actually represent the rectangles in an ORM schema. Every Object\_Role element needs a generated attribute 'ID' which identifies the Object\_Role. Note that we did not put the 'ID' attribute to refer to predicates rather than to object\_role elements. The reason therefore is that we need to be able to refer to specific object-roles when we define constraints.

Predicates can have one or more rule elements. These elements can contain extra rules that are defined for the predicate. Predicates also have two boolean attributes that are optional: ‘Derived’ and ‘Derived\_Stored’ which indicate whether a predicate respectively is derived, or derived and stored, or not.

**Example 4.** This example shows a simple binary predicate as in Fig 4, and how it is represented in ORM-ML in Table 4.

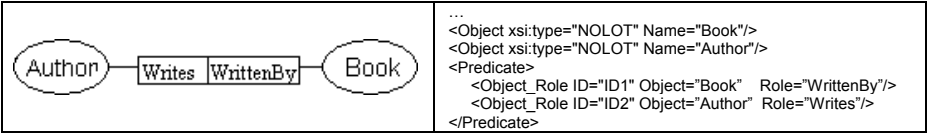


Fig. 4

Table 4. ORM-ML representation of Fig. 4

Constraint elements represent the ORM constraints. The Constraint element itself is abstract, but it is implemented by different types of constraints, viz. mandatory, uniqueness, subset, equality, exclusion, frequency, and ring constraints: irreflexive, anti-symmetric, asymmetric, symmetric, intransitive, and acyclic constraints. As mentioned above, we use the ID-s of the Object\_Role elements to define constraints (except for value constraints on an object type, since these are defined in the corresponding object element).

Uniqueness and mandatory constraint elements possess only Object\_Role elements (at least one). These elements are the object\_roles in the ORM diagram on which the constraint is placed. In this way, there is no need to make a distinction between the ORM-ML syntax of "external" and "internal" uniqueness constraints (see [H01]), or between mandatory and disjunctive mandatory constraints (see Example 5 below).

The representation for subset, equality and exclusion constraints is analogous, so we will only discuss them in general terms. Each of these latter constraints has exactly two elements that contain references to (combinations of) object\_role elements. For example, to represent an equality constraint between two predicates, we create a subset element, containing 2 elements ‘First’ and ‘Second’. In the first element we put references to the object\_roles from the first predicate, and in the second we put references to the object\_roles from the second predicate (see Example 5).

**Example 5.** This example shows the representation of the constraints from Fig. 6.

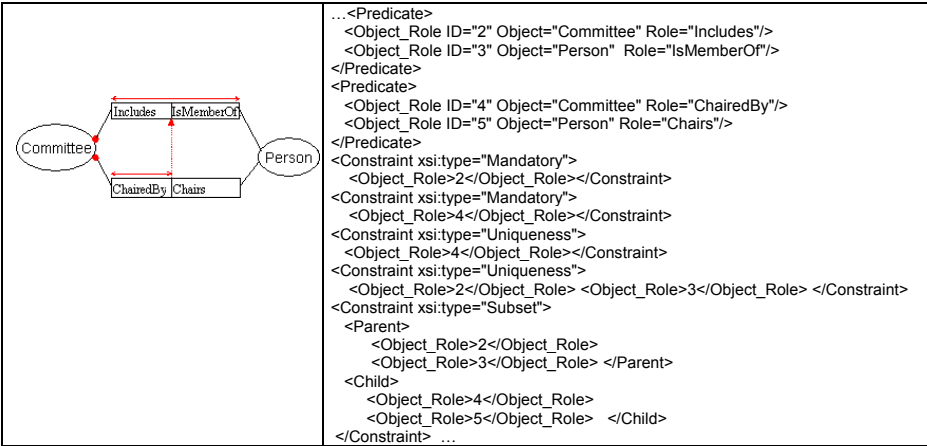


Fig. 5.

Table 5. ORM-ML representation of Fig. 5

Finally, *ring constraint* elements simply contain references to the object\_roles they are put on, and frequency constraints have two attributes: a reference to the



object\_role the constraint is placed on and an attribute called ‘Frequency’ which contains the declared frequency number.

A markup language such as ORM-ML has more advantages than just enabling the use of conceptual schemas at the runtime of applications. We name some of them:

- Building Style-sheets. As ORM-ML is very easy to parse and interpret, we can easily build style-sheets to translate ORM-ML files into other languages. For instance, we have already built a style-sheet to translate ORM-ML files into pseudo natural language sentences, which is used in DogmaModeler (see fig. 12 and fig. 15 in section 5). Moreover, it would be very useful to write style-sheets that translate ORM-ML into rule engine's languages. Experiments with Haley's Authorete™ [Haley] are being planned in the near future. Of course, such a style-sheet would open doors for ORM as a very powerful business rule modeling language. Other style-sheets can be built to translate ORM-ML into other types of conceptual schemes, ontology languages e.g. DAML, or into first order/description logic formalisms, etc.
- Easier schema integration and transformation. For integrating information systems, it is in general easier to integrate or *align* the conceptual models of these systems than integrate their logical or the physical internal representation as demonstrated by the literature on view- and schema integration (e.g. [SP94]). Therefore, ORM-ML as a standardized syntax for ORM models may assist interoperability tools to exchange, parse or understand the ORM schemes.
- Conceptual queries over the web. In open and distributed environments, building queries should be possible regardless of the internal representation of the data. Query languages based on ontologies (seen as shared conceptual models) help users not only to build queries, but also make them more expressive and understandable than corresponding queries in a language like SQL. Exchanging, reusing, or sharing such queries efficiently between agents over the web is substantially facilitated by a standardized markup language. Consequently, NIAM/ORM-based query languages (e.g. RIDL [VB82], [M81], ConQuer [BH96]) would gain from ORM-ML by representing queries in such an exchangeable representation.

Of course, *similar to ORM-ML, a markup language could be defined for any other conceptual modeling method.* We have chosen ORM to *illustrate* adopting conceptual modeling methods for ontology engineering purposes because ORM has several strengths over other methods [H01]: ORM is fairly comprehensive in its treatment of many “practical” and “standard” rules, ( e.g. identity, mandatory, uniqueness, subtyping, subset, equality, exclusion, frequency, transitive, acyclic, anti/a/symmetric... derivation rules, etc.). Furthermore, ORM has an expressive and stable graphical notation since it captures many rules graphically and it minimizes the impact of change to models<sup>5</sup>. ORM has well-defined formal semantics (see e.g. [H89]

---

<sup>5</sup> In comparison with other approaches (e.g. ER, UML), ORM models are attribute-free; so they are immune from changes that cause attributes to be remodeled as entity types or relationships.

[BHW91] [HPW93] [T96] [TM95] [HP95]). In addition, it is perhaps worthwhile to note that ORM derives from NIAM (Natural Language Information Analysis Method), which was explicitly designed to be a stepwise methodology arriving at "semantics" of a business application's data based on natural language communication.

In section 4 and 5, we show how ORM-ML is used as ontological commitment language.

## 4 DOGMA Approach for Ontology Engineering

In this section, we present the second topic of this paper: we first discuss some application-independency issues of ontologies and conceptual data schemes, i.e. domain vs. application conceptual modeling; then we present our DOGMA<sup>6</sup> ontology engineering framework that enables the use of conceptual modeling methods, such as ORM and its ORM-ML, for modeling and representing ontologies.

Similar to conceptual data schemes, ontologies consist of interrelated concepts and rules (e.g. identity, mandatory, value, cardinality, taxonomy, etc.) that constrain and specify the intended meaning of the concepts. However, since conceptual schemes were developed only for the use of an enterprise application(s), thus, building such schemes depends on the specific needs and tasks that are planned to be performed within a certain enterprise. In comparison, building ontologies is a challenging job, since ontologies are supposed to hold application-independent domain knowledge. The consensus level about ontological content is the main requirement in ontology modeling, and mainly distinguishes it from conceptual data modeling.

An expected question may arise, namely: how can one decide whether knowledge is application-independent? Certainly, ontology modelers will not manage to come to an agreement for all applications that can possibly exist in a domain. This is why we believe, like [CJ93], that there is no strict line between the levels of dependent and independent (or generic and specific) knowledge. *When building an ontology, there will always be intended or expected needs and tasks "at hand", which will influence the independency level of the ontology. In short, there is a clash between building an application-independent ontology and encountering the intended goals for building this ontology.* In the problem solving research community, this issue is called the *interaction problem*, which influences the independency of the problem-solver's domain knowledge [HSW97]. For example, Bylander and Chandrasekaran argued in [BC88] that:

"Representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem."

Solving such a clash requires a *principled methodology* to guide ontology modelers towards more genericity, as well as meeting requirements at hand. Notice that the methodologies that emphasize on the requirements "at hand", or that evaluate

---

<sup>6</sup> Developing Ontology-Guided Mediation for Agents

ontologies based only on how they fulfill specific requirements, will lead to *application ontologies*, similar to conceptual data schemas containing less reusable knowledge. Likewise, the methodologies that emphasize only on the genericity of the knowledge will lead to less usable ontologies, since they have no intended use by ignoring requirements at hand [BBH96][RVMS99].

In our approach, we introduce an essential principle that underpins the foundation of ontologies. Unlike ontology engineering proposals, which consider an ontology as one “unit”, holding both conceptual relations and rules together (e.g. [G95], [G98], [FHVDEK00]): we decompose an ontology into an ontology base and a layer of ontological commitments. The ontology base holds conceptual relations, as domain knowledge. The commitment layer consists of a set of ontological commitments, where each commitment holds ontology rules, which *formally and explicitly* provide an interpretation of an application or task in terms of the domain knowledge, (see fig. 6). We will show that ontology rules are mostly application/task-dependent knowledge, i.e. strongly influenced by the intended use of the knowledge and requirements at hand. Therefore, *as a result of the decomposition, the genericity of the knowledge in the ontology base level is increased, while rules influenced by requirements at hand are kept separated in the commitment layer. Hence, a conceptual schema can be seen as an ontological commitment defined in terms of the domain knowledge.* In what follows, we respectively describe the ontology base and the commitment layer, illustrating both by means of a detailed example. For further details about the DOGMA approach, see [JM02a] and [JM02b].

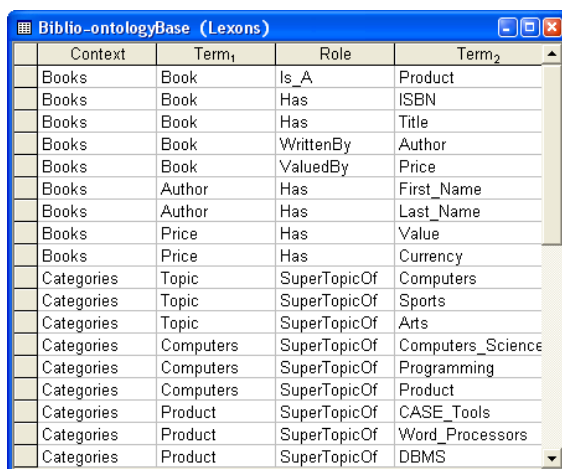


Fig. 6. Knowledge organization in the DOGMA approach

The ontology base consists of “plausible” intuitive domain fact types, represented and organized as a set of context-specific binary conceptual relations, called *lexons*. A lexon is formally described as a 4-tuple of the form  $\langle \gamma: \text{Term}_1, \text{Role}, \text{Term}_2 \rangle$ , where  $\gamma$  is a context identifier, used to group lexons that are intuitively and informally “related” in an intended conceptualization of a domain. For each context  $\gamma$  and Term  $T$ , the pair  $(\gamma, T)$  is assumed to refer to a *concept*.

Fig. 7. shows an example of an ontology base represented in a table format (see fig. 13 for its corresponding “tree” representation using DogmaModeler). The ontology base in this example consists of two contexts: “Books” and “Categories”. Notice that the term “Product” that appears within both contexts refers to two different concepts: the intended meaning of “Product” within the context “Categories” –that deal with the subject classification of books- refers to a topic or a subject of a book, while within

the context “Books”, it refers to a real world’s entity. More about our notion of context will be discussed at the end of this section.



Context	Term <sub>1</sub>	Role	Term <sub>2</sub>
Books	Book	Is_A	Product
Books	Book	Has	ISBN
Books	Book	Has	Title
Books	Book	WrittenBy	Author
Books	Book	ValuedBy	Price
Books	Author	Has	First_Name
Books	Author	Has	Last_Name
Books	Price	Has	Value
Books	Price	Has	Currency
Categories	Topic	SuperTopicOf	Computers
Categories	Topic	SuperTopicOf	Sports
Categories	Topic	SuperTopicOf	Arts
Categories	Computers	SuperTopicOf	Computers_Science
Categories	Computers	SuperTopicOf	Programming
Categories	Computers	SuperTopicOf	Product
Categories	Product	SuperTopicOf	CASE_Tools
Categories	Product	SuperTopicOf	Word_Processors
Categories	Product	SuperTopicOf	DBMS

**Fig. 7.** Example of an ontology base

The layer of ontological commitments mediates between the ontology base and its applications. Each commitment consists of: (1) an *ontological view* that specifies which lexons from the ontology base are *relevant*<sup>7</sup> to this commitment, i.e. selection of lexons, and (2) rules that formally constrain and specify the intended meaning of the selected lexons. For simplicity, one can see a commitment as a conceptual schema, where its conceptual relations correspond to lexons in the ontology base. Applications that use (or more precisely, “commit to”) a certain commitment must satisfy all rules declared in this commitment. In other words, any possible world, for an application, must conform to the rules declared in its commitment(s) (cf. model-theoretic semantics).

Each ontological commitment corresponds to an explicit *instance* of an (intensional) first order *interpretation* of the domain knowledge in the ontology base. In other words, it is the role of commitments to provide the formal interpretation(s) of the lexons. Therefore, the lexons in an ontology base are free of a *particular* formal interpretation. This allows different formalizations and interpretations, even if sometimes they disagree about certain things, to co-exist as different commitments in the commitment layer and to share what they have in common.

In our approach, ontological commitments are not restricted to be expressed in a certain specification language. However, as the goal of this paper is to enable the use of conceptual modeling methods, we illustrate representing ontological commitments using the ORM-ML. In the next example, as well as in the next section, we illustrate how commitments can be modeled using the ORM graphical notation, while the corresponding ORM-ML is being generated automatically.

<sup>7</sup> Notice that the relevancy is an application-dependent choice.

In the next example, we show how an ontology base -holding domain knowledge- may have different formal interpretations in different commitments, and that is because of the difference in the intended use of the same domain knowledge.

Example

The example given below is based on the Biblio-OntologyBase domain knowledge provided in Fig. 7. We present two different *kinds* of applications: library applications that need to interoperate with other libraries, and bookstore applications that additionally need to interoperate with other bookstores. Each kind of application has different rules (i.e. interpretation) that do not necessarily agree with the other’s rules. E.g., unlike bookstores, pricing information is not relevant for library applications. Likewise, bookstores identify a book by its ISBN, while in library applications, ISBN is not a mandatory property for every instance of a book, so it cannot be used for identity<sup>8</sup> purposes; instead, they identify a book by the combination of its title and authors. So for bookstores, instances of a “Thesis”, a ‘Manual’, etc. are not considered as books -since they do not have ISBN- while for libraries they are. However, suppose that both bookstore and library applications have the same agreement about categories (i.e. topics of books). Fig. 8, Fig. 9, and Fig. 10 illustrate the use of the ORM graphical representation of ontological commitments for ‘Bookstore’ ‘Library’, and ‘Categories’ respectively.

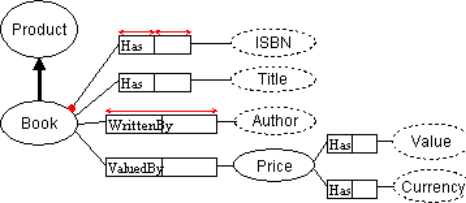


Fig. 8. ‘Bookstore’ Ontological Commitment

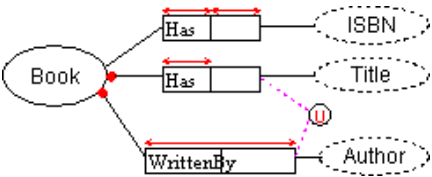


Fig. 9. ‘Library’ Ontological Commitment

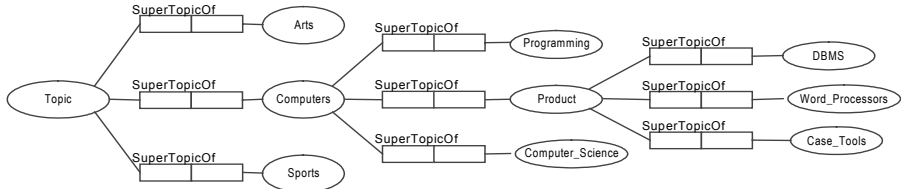


Fig. 10. ‘Categories’ Ontological Commitment<sup>9</sup>

<sup>8</sup> Notice that in ORM, the identity of a concept is based on the Uniqueness and Mandatory properties together. For example, if an ISBN is a mandatory and a unique property for all instances of the concept book, then one can use the ISBN as an identify property for the concept book. Furthermore, in ORM, one can use the combination of two (or more) properties to identify a concept, e.g. the combination of (title, author) of a book. In that case both properties must be unique and mandatory for every instance of a book, at any time.

<sup>9</sup> Notice that we do not use the subtype relationship in ORM to represent the “SuperTopicOf” relationship in Fig. 11; since they do not have the same formal semantics. For more information about the formalization of topics and subjects, see [WJ99].

All commitments share the same Biblio-OntologyBase shown in Fig. 7: all fact types in each commitment correspond to lexons in the ontology base. Notice that a commitment only uses the lexons that are relevant to it, i.e. the ontological view. For example, the lexons {<Books: Book, ValuedBy, Price>, <Books: Price, Has, Value>...} do not appear in the ‘Library’ commitment.

Fig. 11 shows commitments sharing the same Biblio-OntologyBase together with some bookstore applications committing to the ‘Bookstore’ and ‘Categories’ commitments, and some library applications committing to the ‘Library’ and ‘Categories’ commitments.

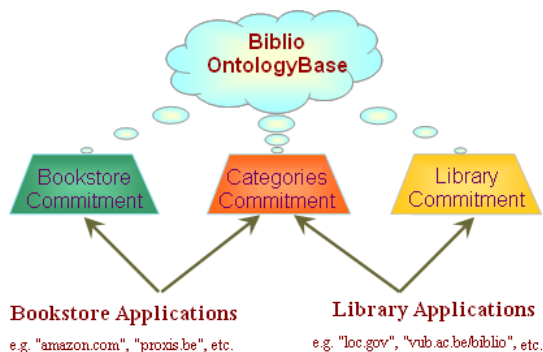


Fig. 11. Using ontological commitments

## Discussion

One can see from the previous example that application-kinds, even within the same domain, may have different formal interpretations of the same knowledge. For example, the set of possible instances of the concept “book” for library applications is formally not the same as for bookstore applications, since both have different identity criteria, etc. Nevertheless, in reality, both kinds of applications intuitively share the same concept of what is really a book. For example, suppose that one assigns an ISBN for an instance of a “Master Thesis”, then this instance can be considered as a book for bookstores, or that one removes an ISBN for an instance of a book, then this instance will no longer be a book, although, this instance remains the same real life object and is still being referred to and used as a book. Obviously, one of the main requirements at hand (i.e. intended use) for bookstores is “what can be sold”, which influences the identity criteria and thus the formal interpretation of the domain knowledge. However, there is more to “say”, than an ISBN, in order to identify a book in real world domains (e.g. structuring, formatting, authoring, publishing, printing, versioning, copying...). And, that the identity criteria should be based on essential properties that help to distinguish real world objects as being the same or not [G02]. In other words, for modelling ontologies at the domain level, the identity of a concept should serve to identify all of its instances in all applications.

Please note that the aim of this paper is not to discuss the identity rule, but we use this example to illustrate the difference (and requirements) between modeling knowledge at the domain level vs. modeling knowledge at the application level.

At the domain level, in our approach, we have introduced the notion of context; so that a term within a context refers *intuitively* to a concept. The intuition that a context provides here is: *a set of implicit or maybe tacit<sup>10</sup> assumptions that are necessary for all instances of a concept in all its applications*. In other words, a context is an abstract identifier that refers to implicit and tacit assumptions in a domain, and that maps a term to its intended meaning (i.e. concept) within these assumptions. Notice that a context in our approach is not explicit formal knowledge. In practice, we define a context by referring to a source (e.g. a set of documents, laws and regulations, informal description of “best practice”, etc.), which, by *human understanding*, is assumed to “contain” the necessary assumptions.

We suppose that the ontology base –as intuitive domain knowledge- is free of any particular formal interpretation; or rather, lexons are assumed (by human understanding) to be “true within their context’s source”. The formal interpretation of the lexons is provided through ontological commitments, which are explicit and formal (and thus machine-understandable) knowledge.

As a result and as we have illustrated before, we enable the use of conceptual modeling methods for modeling ontological commitments. The application-independency level of an ontology is increased, by separating the commitments (mostly application/task-dependent knowledge) from the ontology base (intuitive domain knowledge). In other words, the interaction problem has “neglectable” influence on the genericity of the ontology base level, because ontology modelers are *prevented* from entering their application-specific rules at this level.

**Remark:** In accordance to the given independency discussion, we emphasize that modeling ontological commitments should not be specific to certain needs, *they should be made more generic (e.g. describing application kinds, generic tasks, etc.), and seen as reusable components of knowledge*. In our approach, the ontological commitments that are specific to a limited number of applications do not affect the independency of other commitments in the same commitment layer. Rather, *Commitments -specially large ones- can even be modularized into smaller and interrelated commitments*, so that the general purpose –i.e. reusable- parts can be separated from the more specific parts [JM02b]. And therefore, not only the ontology base (i.e. lexons and the intuitive definitions of their terms) can be shared and reused among commitments, but also the ontological commitments themselves can be modularized and seen as a set of reusable knowledge components.

---

<sup>10</sup> The difference between implicit and tacit assumptions, is that the implicit assumptions, in principle, can be articulated but still they have not, while tacit assumptions are the knowledge that cannot be articulated: it consists partially of technical skills –the kind of informal, hard-to-pin-down skills captured in terms like “know-how”, “we know more than we can tell or put in words”, etc. Even if tacit assumptions cannot be articulated, but they can be transferred through other means over than verbal or formal descriptions [Innovanet03] [N94]. In many cases, the knowledge about the real world (such as the identity of a person, book, etc.) is tacit knowledge [P96].

## 5 DogmaModeler Ontology Engineering Tool

This section briefly outlines our DogmaModeler Tool prototype for ontology engineering. Its implementation is based on the approach described in this paper.

DogmaModeler supports functionalities for modeling, browsing, and managing both the ontology base and the commitments. It supports modeling ontological commitments using the ORM graphical notation, and it generates the corresponding ORM-ML automatically. In addition, DogmaModeler supports verbalization of ontological commitments into pseudo natural language. Fig. 12 shows a screenshot of DogmaModeler with demonstrates its three main windows: the ontology base window, the commitment modeling window, and the commitment library window. We will describe these windows in what follows.

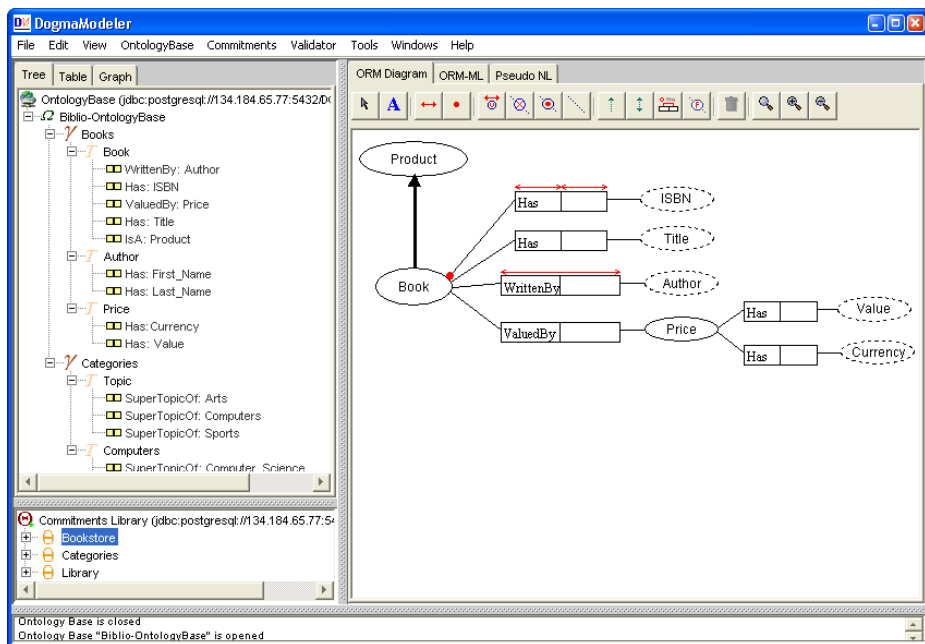


Fig. 12. A screenshot of DogmaModeler.

### Ontology base window (The top left side of Fig. 12)

Before building ontological commitments, ontology builders should define their lexons in the ontology base window, in case it is empty. This window presents the set of lexons –  $\{ \langle \gamma : \text{Term}_1, \text{Role}, \text{Term}_2 \rangle \}$  - in a tree-like structure<sup>11</sup>. The first level, ( $\Omega$ ) represents ontology bases (e.g. Biblio-Ontologybase). In the second level, each node ( $\gamma$ ) represents a context (e.g. Books). Within a context, each node ( $\mathcal{T}$ ), in the third level, represents a term; while nodes ( $\square$ ) in the fourth level, represent the set of (Role, Term<sub>2</sub>) for that term.

<sup>11</sup> The ontology base tree has advanced features, so it can also be browsed and seen as a graph.



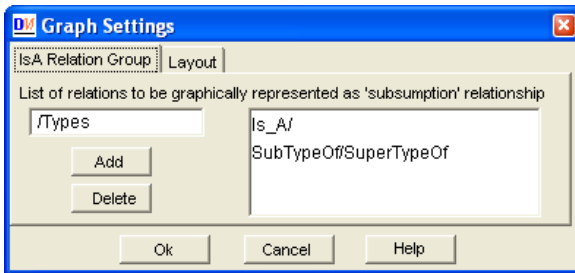
Notice that level 0 (🌐) in the tree represents an ontology base server, where the content of ontology bases is hosted and managed. All transactions on the ontology base (e.g. creating contexts, editing lexons, etc.) will be transmitted, verified and executed on the server side. As one can see in Fig. 12, DogmaModeler is connected with our DogmaServer<sup>12</sup>, which stores and serves the ontology base and the commitment layer.

#### Commitment modeling window (The right side of Fig. 12)

This window consists of three panels: ORM, ORM-ML, and Pseudo NL. To build an ontological commitment, ontology builders can drag and drop lexons from the ontology base window into the ORM panel (i.e. defining the ontological view). When doing so, lexons will be mapped automatically into ORM fact types. Then, in order to define rules on these lexons, ontology builders can use the ORM family of constraints; see icons in the top of the ORM panel.

*Remark:* mapping lexons as intuitive domain knowledge into ORM fact types that have predefined formal semantics [V82] is done as the following: a Term within a context is mapped directly into an Object Type in ORM, Roles within a context are also mapped directly into ORM Roles. While in case of ORM Subtype relations that have specific “build-in” semantics, commitment builders need to customize the “Graph settings” window, in order to specify which roles should be mapped, see Fig. 13. Further, DogmaModeler does not support ORM unary roles and nested fact types.

As we mentioned before, our approach is not restricted to ORM; the tool is designed with flexibility of adding new plug-ins in order to support modeling commitments in other languages, e.g. EER, UML, DAML, OWL etc.



**Fig. 13.** Mapping to ORM Subtyping relationship

Fig. 14 shows the corresponding ORM markup language of the ORM model in Fig. 12, which is automatically generated by the tool. DogmaModeler supports saving ORM-ML into text files, or uploading it into an ontology server.

<sup>12</sup> For more details, or to download DogmaServer, you can access:  
<http://www.starlab.vub.ac.be/research/dogma/OntologyServer.htm>.

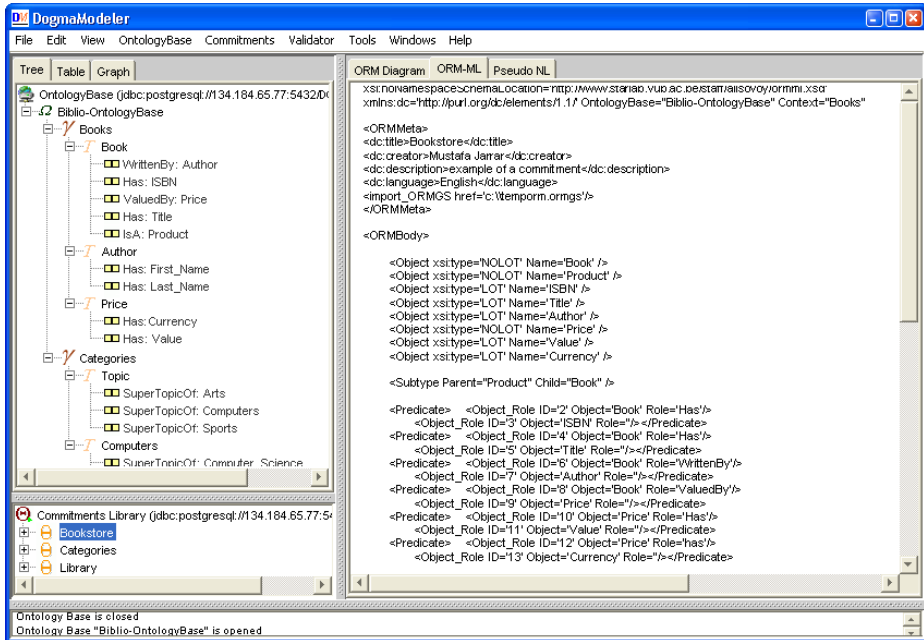


Fig. 14. The ORM-ML panel window

Fig. 15 shows the corresponding Pseudo Natural language (fixed-syntax English sentences) of the ORM model from Fig. 12. It is automatically generated by the tool by applying predefined templates to the commitments' content. We believe that this allows non-experts to (help to) check, validate or build the commitment rules and will simplify the modeling process.

#### Commitment library window (Under the ontology base window)

The purpose of this window is to enhance the reusability, management, and organization of ontological commitments. The current implementation allows ontology builders to access and browse ontological commitments stored in a library (📁). Each node (📁) in the first level of the tree represents a commitment. By expanding a commitment node, the set of lexons and the set of rules -subject to this commitment- will appear in the second level. Advanced features e.g. indexing, modularization/composing, versioning, etc. of ontological commitments are ongoing research issues.

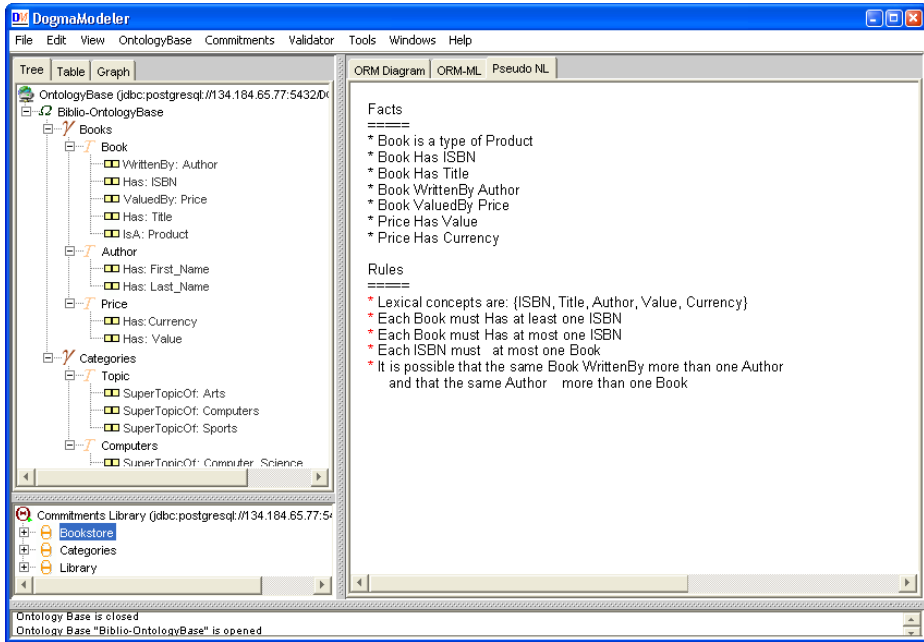


Fig. 15. The Pseudo NL panel window

## 6 Conclusions and Future Work

This paper has presented and discussed two main disparities between conceptual data schemes and ontologies, and we have shown how these disparities can be tackled when adopting conceptual data modeling techniques for ontology engineering purposes. First, we have presented how conceptual diagrams can be marked up and thus accessed and processed at run-time of applications. We have illustrated this by defining a conceptual markup language in order to textually represent ORM conceptual diagrams. Second, we have discussed and analyzed the differences between modeling knowledge at the application level vs. modeling knowledge at the domain level and thus conceptual data schemes vs. ontologies. We have presented an ontology engineering approach and tool that increase the application-independency of an ontology by decomposing it into an ontology base (that holds intuitive domain knowledge) and a set of ontological commitments (that hold application specific knowledge). Hence, we have enabled conceptual data modeling methods to be used for modeling and representing ontological commitments.

Future research on our approach concerns the development and engineering of commitment libraries, which open new several research issues such as e.g. indexing, versioning, distributed development, modularization, etc. of ontological commitments. Currently, our main priority is to develop and formalize a methodology for modularizing (and thus composing) ontological commitments. As a work in progress, we have defined an inclusion interrelationship between ontological

commitments, so that all concepts and constraints introduced in the included commitment will be inherited in the including commitment.

**Acknowledgments.** The authors are grateful to Jan De Bo, Peter Spyns, Sven Van Acker, Ruben Verlinden, Pieter De Leenheer, and Andriy Lisovoy for their comments and suggestions on an earlier version of this paper.

## References

- [BB03] Borgida, A., Brachman, R.: Conceptual Modeling with Description Logics. In: Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., (eds): *The Description Logic Handbook, Theory, Implementation and Applications*. ISBN: 0521781760 (2003).
- [BBH96] Beys, P., Benjamins, R., van Heijst, G.: Remedying the reusability-usability trade-off for problem solving methods. In B.R. Gaines and M. Mussen, (eds): *Proceedings of the KAW-96*, Banff, Ca, (1996)
- [BC88] Bylander, T., Chandrasekaran, B.: Generic tasks in knowledge-based reasoning: The right level of abstraction for knowledge acquisition. In: Gaines B., Boose, J. (eds): *Knowledge Acquisition for Knowledge Based Systems*. Vol. 1. Academic Press, London, (1988) 65–77
- [BCD01] Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML Class Diagrams using Description Logic Based Systems. In: *Workshop on Applications of Description Logics- ADL-2001*, (2001)
- [BH96] Bloesch, A., Halpin, T.: ConQuer: a Conceptual Query Language. In: Thalheim, B. (ed.): *Conceptual Modeling - ER'96 Proceedings, Lecture Notes in Compute Science*, Springer-Verlag, (1996) 121–33
- [BHW91] van Bommel, P., ter Hofstede, A., van der Weide, Th. P.: Semantics and verification of object-role models. *Information Systems*, 16(5), October (1991) 471–495
- [BKKHSHLA01] Baclawski, K., Kokar, M., Kogut, P., Hart, J., Smith, J., Holmes, W., Letkowski J., Aronson, M.: Extending UML to Support Ontology Engineering for the Semantic Web, 4th International Conference on UML, (2001) 342–360
- [CHP01] Cranefield, S., Hausteine, S., Purvis, M.: UML-Based Ontology Modelling for Software Agents. In: *Proceedings of the Workshop on Ontologies in Agent Systems*, 5th International Conference on Autonomous Agents, Montreal (2001) 21–28
- [CJ93] Chandrasekaran, B., Johnson T.: Generic Tasks and Task Structures: History, Critique and New Directions. In: David, J., Krivine, J., Simmons, R. (eds.): *Second Generation Expert Systems*, Springer, (1993) 233–272
- [CP99] Cranefield, S., Purvis, M.: UML as an ontology modelling language. In: *Workshop on Intelligent Information Integration*, 16th International Joint Conference on Artificial Intelligence, IJCAI-99, (1999)
- [F02] Franconi, E.: Tutorial on Description Logics for Conceptual Design, Information Access, and Ontology Integration: Research Trends. 1st International Semantic Web Con. (2002)
- [FHVDEK00] Fensel, D., Horrocks, I., Van Harmelen, F., Decker, S., Erdmann, M., Klein, M.: Oil in a nutshell. 12th International Conference on Knowledge Engineering and Knowledge Management EKAW 2000, Juan-les-Pins, France, (2000)
- [G95] Gruber T.R.: Toward principles for the design of ontologies used for knowledge sharing, *International Journal of Human-Computer Studies*, 43(5/6) (1995)
- [G98] Guarino, N.: Formal Ontology in Information Systems. In: *Proceedings of FOIS'98*, IOS Press, Amsterdam, (1998) 3–15

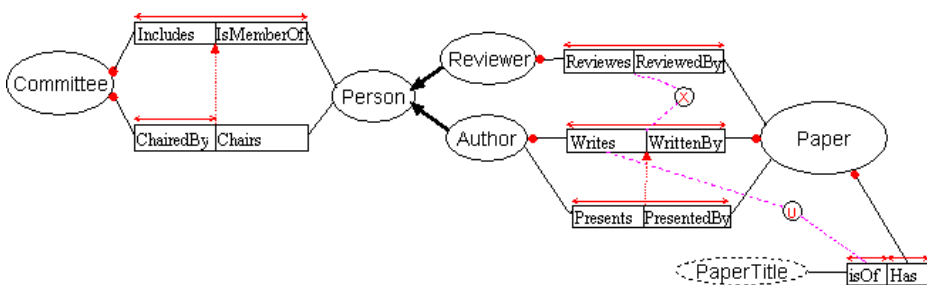
- [GHW02] Guizzardi, G., Herre, H., Wagner G.: Towards Ontological Foundations for UML Conceptual Models. 1st International Conference on Ontologies, Databases and Application of Semantics (ODBASE'02), Lecture Notes in Computer Science, Vol. 2519, Springer-Verlag, Berlin (2002) 1100–1117
- [GW02] Guarino, N. and Welty, C.: Evaluating Ontological Decisions with OntoClean. Communications of the ACM, 45(2): 61–65 (2002).
- [H89] T.A. Halpin. A logical analysis of information systems: static aspects of the data-oriented perspective. PhD thesis, University of Queensland, Brisbane, Australia, 1989.
- [H01] Halpin, T.: Information Modeling and Relational Databases. 3rd edn, Morgan-Kaufmann [Haley] The HaleyEnterprise, <http://www.haley.com>.
- [HP95] Halpin, T., Proper, H.: Subtyping and polymorphism in object-role modeling. Data & Knowledge Engineering 15(3), (1995) 251–281
- [HPW93] ter Hofstede, A., Proper, H., van der Weide, T.: Formal definition of a conceptual language for the description and manipulation of information models. In: Information Systems 18(7), October (1993) 471–495
- [HSW97] van Heijst, G., Schreiber, A., Wielinga, B.: Using Explicit Ontologies in KBS Development, International Journal of Human-Computer Studies, 46, (1997) 183–292.
- [Innovanet03] Persidis A., Niederée C., Muscogiuri C., Bouquet P., & Wynants M.: Innovation Engineering for the Support of Scientific Discovery. Innovanet Project (IST-2001-38422), deliverable #D1, 2003.
- [JM02a] Jarrar, M., Meersman, R.: Formal Ontology Engineering in the DOGMA Approach. In: 1st International Conference on Ontologies, Databases and Application of Semantics (ODBASE'02), Lecture Notes in Computer Science, Vol. 2519, Springer-Verlag, Berlin (2002) 1238–1254
- [JM02b] Jarrar, M., Meersman, R.: Scalability and Knowledge Reusability in Ontology Modeling, In: Proceedings of the International conference on Infrastructure for e-Business, e-Education, e-Science, and e-Medicine (SSGRR2002s) (2002)
- [M81] Meersman, R.: Languages for the High-Level End User. In: InfoTech State of the Art Report, Pergamon Press (1981)
- [M99] Meersman R.: The Use of Lexicons and Other Computer-Linguistic Tools. In Zhang Y., Rusinkiewicz M., & Kambayashi Y. (eds.): Semantics, Design and Cooperation of Database Systems, in The International Symposium on Cooperative Database Systems for Advanced Applications (CODAS 99), Springer Verlag, Heidelberg, (1999) 1–14
- [M01] Meersman, R.: Ontologies and Databases: More than a Fleeting Resemblance. In d'Atri A., Missikoff, M. (eds): OES/SEO 2001 Rome Workshop, Luiss Publications (2001)
- [MC02] Meisel, H., Compatangelo, E.: EER-ConceptTool: a “reasonable” environment for schema and ontology sharing. In: Proc. of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'2002), IEEE Computer Society Press (2002) 527–534
- [N94] I. Nonaka, A dynamic theory of organizational knowledge creation, In: Organizational Science, Vol. 5, No. 1 (1994), pp. 14–37
- [ORMML-XMLS] <http://www.starlab.vub.ac.be/ORMML/ormml.xsd>
- [P96] Polany, M.: The Tacit Dimension. Doubleday, Garden City-N.Y. (1996)
- [RFC2413] <http://www.ietf.org/rfc/rfc2413.txt> (Dublin Core definition)
- [RVMS99] Russ, T., Valente, A., MacGregor, R., Swartout, W.: Practical Experiences in Trading Off Ontology Usability and Reusability. In Proceedings of the Twelfth Banff Knowledge Acquisition for Knowledge-based Systems Workshop, pp. 4–11–1 to 4–11–20, (1999)
- [SOVZJSSM02] Spyns, P., Oberle, D., Volz, R., Zheng, J., Jarrar, M., Sure, Y., Studer, R., Meersman, R.: OntoWeb - a Semantic Web Community Portal. In Karagiannis, D., Reimer, U., (eds.): Proceedings of the Fourth International Conference on Practical Aspects of Knowledge Management (PAKM02), LNAI 2569, Springer Verlag, (2002) 189–200

- [SP94] Spaccapietra, S., Parent, C.: View Integration: A Step Forward in Solving Structural Conflicts. In: IEEE Transactions on Data and Knowledge Engineering 6(2), (1994)
- [T96] de Troyer, O.: A Formalization of the Binary Object-Role Model based on Logic. In: Data & Knowledge Engineering 19, North-Holland Elsevier (1996) 1–37
- [TM95] de Troyer, O., Meersman, R.: A Logic Framework for a Semantics of Object-Oriented Data Modelling. In: Papazoglou, M.P. (eds): Proceedings of 14th International Conference Object-Oriented and Entity-Relationship Modelling (OO-ER'95), Lecture Notes in Computer Science 1021, Springer (1995) 238–249
- [TSC01] Tzitzikas, Y., Spyrtos, N., Constantopoulos, P.: Mediators over Ontology-based Information Sources. In: Second International Conference on Web Information Systems Engineering, WISE'01, (2001)
- [V82] Van Griethuysen, J.J., (Ed.): Concepts and Terminology for the Conceptual Schema and Information Base. International Standard for Standardization, Publication No. ISO/TC97/SC5- N695, (1982)
- [VB82] Verheijen, G., van Bekkum, P.: NIAM, an Information Analysis Method. In: Olle, T.W., Sol, H. and Verrijn-Stuart A. (eds.), IFIP Conference on Comparative Review of Information Systems Methodologies, North-Holland, (1982) 537–590
- [W90] Wintraecken, J.J.V.R.: The NIAM Information Analysis Method: Theory and Practice. Kluwer, Deventer. (1990)
- [WJ99] Welty, C., Jessica, J.: An Ontology for Subject. In J. Data and Knowledge Engineering. 31(2)155–181. Elsevier. (1999)
- [WSW99] Wand, Y., Storey, V., Weber, R.: An Ontological Analysis of the relationship Construct in Conceptual Modelling. In: ACM Transactions on Database Systems, Vol. 24, No. 4, (1999) 494–528

## Appendix A

A complete example of an ORM Schema with the associated ORM-ML file, ORM pseudo NL generated by the DogmaModeler tool.

### ORM Schema



### ORM-ML

```
<?xml version="1.0" encoding="UTF-8"?>
<ORMSchema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://starlab.vub.ac.be/ORMML/omml.xsd"
xmlns:dc="http://purl.org/dc/elements/1.1/">
```

```

<ORMMeta>
  <dc:title>ORM ML example</dc:title>
  <dc:creator> Mustafa Jarrar </dc:creator>
  <dc:description>A complete example of an ORM ML file</dc:description>
  <dc:contributor>Jan Demey</dc:contributor>      </ORMMeta>
<ORMBody>
  <Object xsi:type='NOLOT' Name='Committee' />
  <Object xsi:type='NOLOT' Name='Person' />
  <Object xsi:type='NOLOT' Name='Author' />
  <Object xsi:type='NOLOT' Name='Reviewer' />
  <Object xsi:type='NOLOT' Name='Paper' />
  <Object xsi:type='LOT' Name='PaperTitle' />
  <Subtype Parent='Person' Child='Author'/>
  <Subtype Parent='Person' Child='Reviewer'/>
  <Predicate>
    <Object_Role ID='2' Object='Committee' Role='Includes'/>
    <Object_Role ID='3' Object='Person' Role='IsMemberOf'/> </Predicate>
  <Predicate>
    <Object_Role ID='4' Object='Committee' Role='ChairedBy'/>
    <Object_Role ID='5' Object='Person' Role='Chairs'/> </Predicate>
  <Predicate>
    <Object_Role ID='6' Object='Reviewer' Role='Reviews'/>
    <Object_Role ID='7' Object='Paper' Role='ReviewedBy'/> </Predicate>
  <Predicate>
    <Object_Role ID='8' Object='Author' Role='Writes'/>
    <Object_Role ID='9' Object='Paper' Role='WrittenBy'/> </Predicate>
  <Predicate>
    <Object_Role ID='10' Object='Author' Role='Presents'/>
    <Object_Role ID='11' Object='Paper' Role='PresentedBy'/> </Predicate>
  <Predicate>
    <Object_Role ID='12' Object='PaperTitle' Role='isOf'/>
    <Object_Role ID='13' Object='Paper' Role='Has'/> </Predicate>
  <Constraint xsi:type='Mandatory'> <Object_Role>2</Object_Role></Constraint>
  <Constraint xsi:type='Mandatory'> <Object_Role>4</Object_Role></Constraint>
  <Constraint xsi:type='Mandatory'> <Object_Role>6</Object_Role></Constraint>
  <Constraint xsi:type='Mandatory'> <Object_Role>8</Object_Role></Constraint>
  <Constraint xsi:type='Mandatory'> <Object_Role>9</Object_Role></Constraint>
  <Constraint xsi:type='Mandatory'> <Object_Role>13</Object_Role></Constraint>
  <Constraint xsi:type='Uniqueness'> <Object_Role>4</Object_Role></Constraint>
  <Constraint xsi:type='Subset'>
    <Parent><Object_Role>2</Object_Role><Object_Role>3</Object_Role> </Parent>
    <Child> <Object_Role>4</Object_Role> <Object_Role>5</Object_Role> </Child>
  </Constraint>
  <Constraint xsi:type='Uniqueness'>
    <Object_Role>2</Object_Role><Object_Role>3</Object_Role></Constraint>
  <Constraint xsi:type='Uniqueness'>
    <Object_Role>10</Object_Role><Object_Role>11</Object_Role></Constraint>
  <Constraint xsi:type='Uniqueness'>
    <Object_Role>8</Object_Role><Object_Role>9</Object_Role></Constraint>
  <Constraint xsi:type='Uniqueness'>
    <Object_Role>6</Object_Role><Object_Role>7</Object_Role></Constraint>
  <Constraint xsi:type='Exclusion'>
    <First><Object_Role>8</Object_Role><Object_Role>9</Object_Role></First>
    <Second><Object_Role>6</Object_Role><Object_Role>7</Object_Role></Second>
  </Constraint>
  <Constraint xsi:type='Uniqueness'>
    <Object_Role>12</Object_Role><Object_Role>8</Object_Role></Constraint>
  <Constraint xsi:type='Uniqueness'> <Object_Role>13</Object_Role></Constraint>

```

```

<Constraint xsi:type='Uniqueness'><Object_Role>12</Object_Role></Constraint>
<Constraint xsi:type='Subset'>
  <Parent> <Object_Role>8</Object_Role> <Object_Role>9</Object_Role> </Parent>
  <Child> <Object_Role>10</Object_Role> <Object_Role>11</Object_Role> </Child>
</Constraint>
</ORMBody>
</ORMSchema>

```

## ORM Verbalization

(Pseudo NL sentences, generated by DogmaModeler)

●	Each Committee must ChairedBy at least one Person.
●	Each Committee must Includes at least one Person.
●	Each Reviewer must Reviewes at least one Paper.
●	Each Author must Writes at least one Paper.
●	Each Paper must WrittenBy at least one Author.
↔	Each Paper must Has at most one PaperTitle.
↔	Each PaperTitle must isOf at most one Paper.
↔	Each Committee must ChairedBy at most one Person.
↔	It is disallowed that the same Committee Includes the same Person more then once, and it is disallowed that the same Person IsMemberOf the same Committee more then once.
↔	It is disallowed that the same Author Presents the same Paper more then once, and it is disallowed that the same Paper PresentedBy the same Author more then once.
↔	It is disallowed that the same Author Writes the same Paper more then once, and it is disallowed that the same Paper WrittenBy the same Author more then once.
↔	It is disallowed that the same Reviewer Reviewes the same Paper more then once, and it is disallowed that the same Paper ReviewedBy the same Reviewer more then once.
↑	Each Person who Chairs a Committee must also IsMemberOf that Committee.
↑	Each Paper who WrittenBy a Author must also PresentedBy that Author.
⊗	Each Paper which is WrittenBy a Person must not ReviewedBy with that Person.
⊙	Each (PaperTitle, Author) as a combination refers to at most one Paper.



# The DAQUINCIS Broker: Querying Data and Their Quality in Cooperative Information Systems

Massimo Mecella<sup>1</sup>, Monica Scannapieco<sup>1,2</sup>, Antonino Virgillito<sup>1</sup>,  
Roberto Baldoni<sup>1</sup>, Tiziana Catarci<sup>1</sup>, and Carlo Batini<sup>3</sup>

<sup>1</sup> Università di Roma “La Sapienza”

Dipartimento di Informatica e Sistemistica

Via Salaria 113, I-00198 Roma, Italy

{mecella, monscan, virgi, baldoni, catarci}@dis.uniroma1.it

<sup>2</sup> Consiglio Nazionale delle Ricerche

Istituto di Analisi dei Sistemi ed Informatica

<sup>3</sup> Università di Milano “Bicocca”

Dipartimento di Informatica, Sistemistica e Comunicazione

Edificio U7, Via Bicocca degli Arcimboldi 8, I-20126 Milano, Italy

batini@disco.unimib.it

**Abstract.** In cooperative information systems, the quality of data exchanged and provided by different data sources is extremely important. A lack of attention to data quality can imply data of low quality to spread all over the cooperative system. At the same time, improvement can be based on comparing data, correcting them and disseminating high quality data. In this paper, a framework and a related architecture for managing data quality in cooperative information systems is proposed, as developed in the context of the DAQUINCIS research project. Then the focus concentrates *(i)* on an XML-based model for data and quality data, and *(ii)* on the design of a broker, which selects the best available data from different sources; such a broker also supports the improvement of data based on feedbacks to data sources. The broker is the basic component of the DAQUINCIS architecture.

## 1 Introduction

A *Cooperative Information System (CIS)* is a large scale information system that interconnects various systems of different and autonomous organizations, geographically distributed and sharing common objectives. Among the different resources that are shared by organizations, data are fundamental; in real world scenarios, an organization  $\mathcal{A}$  may not request data from an organization  $\mathcal{B}$  if it does not “trust”  $\mathcal{B}$  data, i.e., if  $\mathcal{A}$  does not know that the quality of the data that  $\mathcal{B}$  can provide is high. As an example, in an *e-Government* scenario in which public administrations cooperate in order to fulfill service requests from citizens and enterprises [1], administrations very often prefer asking citizens for data, rather than other administrations that have stored the same data, because the

quality of such data is not known. Therefore, lack of cooperation may occur due to lack of quality certification.

Uncertified quality can also cause a deterioration of the data quality inside single organizations. If organizations exchange data without knowing their actual quality, it may happen that data of low quality spread all over the CIS.

On the other hand, CIS's are characterized by high data replication, i.e., different copies of the same data are stored by different organizations. From a data quality perspective this is a great opportunity: improvement actions can be carried out on the basis of comparisons among different copies, in order either to select the most appropriate one or to reconcile available copies, thus producing a new improved copy to be notified to all involved organizations.

In this paper, we introduce a framework and an overall architecture for managing data quality in CIS's, as developed in the context of the DAQUINCIS research project<sup>1</sup>. The architecture aims at avoiding dissemination of low qualified data through the CIS, by providing a support for data quality diffusion and improvement. At the best of our knowledge, this is a novel contribution in the information quality area, that aims at integrating, in the specific context of CIS, both modeling and architectural issues. To enforce this vision, the current work, beside presenting the general architecture, focuses on two specific elements of the problem, that we consider of primary importance. More specifically:

- we first face the problem of lack of quality certification by proposing a model for each organization to export data with associated quality information. The model is XML-based in order to address interoperability issues existing in cooperative information systems;
- then, we present the distributed design of a single architectural element, based on the model cited above, namely the Data Quality Broker, which allows each organization involved in the CIS to retrieve data specifying their quality requirements. The Data Quality Broker offers only data that satisfy the given requirements and notifies organizations about the highest quality values found within the CIS. Its design takes into account reliable communication issues and shows the feasibility of our approach in practical scenarios.

The structure of the paper is as follows. In Section 2, the proposed framework for Cooperative Information Systems is introduced, and the DAQUINCIS architecture specifically addressing quality related issues is described. In Section 3, a model for both the exchanged data and their quality is presented. In Section 4 the Data Quality Broker is described and its distributed design is discussed. In Section 5, related research work is discussed and compared, and finally Section 6 concludes the paper by drawing future work and possible extensions.

---

<sup>1</sup> “DAQUINCIS – Methodologies and Tools for Data Quality in Cooperative Information Systems” is an Italian research project carried out by Università di Roma “La Sapienza”, Università di Milano “Bicocca” and Politecnico di Milano (<http://www.dis.uniroma1.it/~dq/>).

## 2 A Cooperative Architecture for Data Quality

### 2.1 Definition of Cooperative Information System

In current business scenarios, organizations need to cooperate in order to offer services to their customers and partners. Organizations that cooperate have business links (i.e., relationships, exchanged documents, resources, knowledge, etc.) connecting each other. Specifically, organizations exploit business services (e.g., they exchange data or require services to be carried out) on the basis of business links, and therefore the network of organizations and business links constitutes a cooperative business system.

As an example, a supply chain, in which some enterprises offer basic products and some others assemble them in order to deliver final products to customers, is a cooperative business system. As another example, a set of public administrations which need to exchange information about citizens and their health state in order to provide social aids, is a cooperative business system derived from the Italian *e-Government* scenario [1].

A cooperative business system exists independently of the presence of a software infrastructure supporting electronic data exchange and service provisioning. Indeed cooperative information systems are software systems supporting cooperative business systems; in the remaining of this paper, the following definition of CIS is considered:

*A cooperative information system is formed by a set of organizations  $\{ Org_1, \dots, Org_n \}$  which cooperate through a communication infrastructure  $\mathbb{N}$ , which provide software services to organizations as wells as reliable connectivity. Each organization  $Org_i$  is connected to  $\mathbb{N}$  through a gateway  $G_i$ , on which services offered by  $Org_i$  to other organizations are deployed.*

### 2.2 The Architecture for Data Quality

A typical feature of CIS's is the high degree of data replicated in different organizations. As an example, in an *e-Government* scenario, the personal data of a citizen are stored by almost all administrations. On the basis of the proposed definition of CIS, more than one organization can implement the same service providing access to data; but several organizations can provide the same data though with different quality levels. Any requester of data may want to have the data with the highest quality level, among the provided ones. Thus only the highest quality data are returned to the requester, limiting the dissemination of low quality data. Moreover, the comparison of the gathered data values can be used to enforce a general improvement of data quality in all organizations.

In the context of the DAQUINCIS project, we propose an architecture for the management of data quality in CIS's; such an architecture allows the diffusion of data and related quality and exploits data replication to improve the overall quality of cooperative data. According to the logical model of a CIS presented in the previous section, we need to define both a model for the organizations

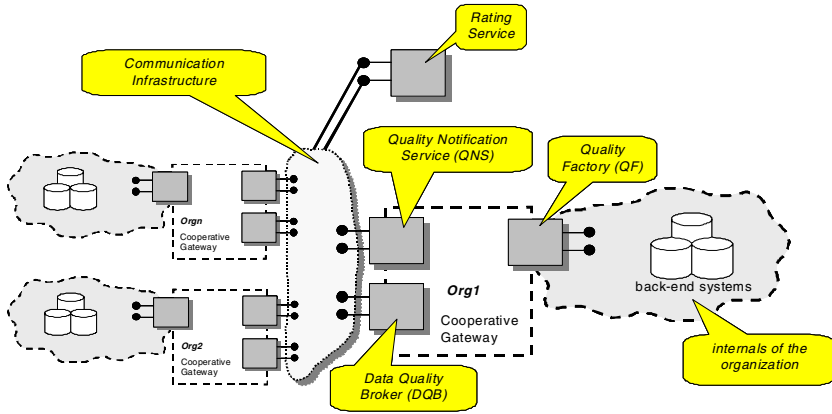


Fig. 1. An architecture for data quality diffusion and improvement

to exchange data and quality data and a set of services that realize quality management functions.

The model for data quality we propose in this paper is called *Data and Data Quality ( $D^2Q$ ) model*. It includes the definitions of (i) constructs to represent data, (ii) a common set of data quality properties, (iii) constructs to represent them and (iv) the association between data and quality data. The  $D^2Q$  model is described in Section 3.

In order to produce data and quality data according to the  $D^2Q$  model, each organization holds a **Quality Factory** that is responsible for evaluating the quality of its own data. The overall architecture is depicted in Figure 1. In the following we give a description of each element:

- **Data Quality Broker:** it is the core of the architecture. It performs, on behalf of a requesting organization, a data request on all organizations, also specifying a set of quality requirements that the desired data have to satisfy (*quality brokering function*). Different copies of the same data received as responses to the request are reconciled and a best-quality value is selected and proposed to organizations, that can choose to discard their data and adopt higher quality ones (*quality improvement function*). Quality brokering and improvement are described in Section 4. If the requirements specified in the request cannot be satisfied, then the broker initiates a negotiation with the requester that can optionally weaken the constraints on the desired data. The Data Quality Broker is in essence a data integration system [2] which allows to pose quality-enhanced query over a global schema and to select data satisfying such requirements.
- **Quality Notification Service:** it is a publish/subscribe engine used as a general message bus between architectural components and/or organizations. More specifically, it allows quality-based subscriptions for organizations to be notified on changes of the quality of data. For example, an organization

may want to be notified if the quality of data it uses degrades below a certain acceptable threshold, or when high quality data are available.

- **Rating Service:** it associates trust values to each data source in the CIS. These are used to determine the reliability of the quality evaluation performed by organizations. In this paper, for sake of simplicity, we assume that all organizations are reliable, i.e., provide trustable values with respect to quality of data.

In this paper, we only focus on the architectural design of the data quality broker, which is detailed in Section 4. The reader interested in other elements can refer to [3,4,5].

### 3 The $D^2Q$ Model

All cooperating organizations export their application data and quality data (i.e., data quality dimension values evaluated for the application data) according to a specific data model. The model for exporting data and quality data is referred to as *Data and Data Quality ( $D^2Q$ ) model*. In this section, we first introduce the data quality dimensions used in this paper (Section 3.1), then we describe the  $D^2Q$  model with respect to the data features (Section 3.2) and the quality features (Section 3.3).

#### 3.1 Data Quality Dimensions

Data quality dimensions are properties of data such as correctness or degree of updating. The data quality dimensions used in this work concern only data values; instead, they do not deal with aspects concerning quality of logical schema and data format [6].

In this section, we propose some data quality dimensions to be used in CIS's. The need for providing such definitions stems from the lack of a common reference set of dimensions in the data quality literature [7] and from the peculiarities of CIS's; we propose four dimensions, inspired by the already proposed definitions, and stemming from real requirements of CIS's scenarios that we experienced [1].

In the following, the general concept of *schema element* is used, corresponding, for instance, to an entity in an Entity-Relationship schema or to a class in a Unified Modeling Language diagram. We define: (i) accuracy, (ii) completeness, (iii) currency, and (iv) internal consistency.

**Accuracy.** In [6], accuracy refers to the proximity of a value  $v$  to a value  $v'$  considered as correct. More specifically:

*Accuracy is the distance between  $v$  and  $v'$ , being  $v'$  the value considered as correct.*

Let us consider the following example: **Citizen** is a schema element with an attribute **Name**, and **p** is an instance of **Citizen**. If **p.Name** has a value  $v = \text{JHN}$ ,

while  $v' = \text{JOHN}$ , this is a case of low accuracy, as JHN is not an admissible value according to a dictionary of English names.

Accuracy can be checked by comparing data values with reference dictionaries (e.g., name dictionaries, address lists, domain related dictionaries such as product or commercial categories lists). As an example, in the case of values on string domain, edit distance functions can be used to support such a task [8]; such functions consider the minimum number of operations on individual characters needed in order to transform a string into another.

Another interesting definition of accuracy is provided in [7], on the basis of ontological considerations; in such a case, the difference is between what is observed in the system state vs. that in the “real world” state. As we are in a multi-system scenario, we prefer a definition not including the concept of system.

**Completeness.** *Completeness is the degree to which values of a schema element are present in the schema element instance.*

According to such a definition, we can consider: (i) the completeness of an attribute value, as dependent from the fact that the attribute is present or not; (ii) the completeness of a schema element instance, as dependent from the number of the attribute values that are present.

A recent work [9] distinguishes other kinds of completeness, namely: schema completeness, column completeness and population completeness. The measures of such completeness types can give very useful information for a “general” assessment of the data completeness of an organization, but, as it will be clarified in Section 3.3, our focus in this paper is on associating a quality evaluation on “each” data a cooperating organization makes available to the others.

In evaluating completeness, it is important to consider the meaning of null values of an attribute, depending on the attribute being mandatory, optional, or inapplicable: a null value for a mandatory attribute is associated with a lower completeness, whereas completeness is not affected by optional or inapplicable null values.

As an example, consider the attribute **E-mail** of the **Citizen** schema element; a null value for **E-mail** may have different meanings, that is (i) the specific citizen has no e-mail address, and therefore the attribute is inapplicable (this case has no impact on completeness), or (ii) the specific citizen has an e-mail address which has not been stored (in this case completeness is low).

**Currency.** The currency dimension refers only to data values that may vary in time; as an example, values of **Address** may vary in time, whereas **DateOfBirth** can be considered invariant. Currency can be defined as:

*Currency is the distance between the instant when a value changes in the real world and the instant when the value itself is modified in the information system.*

More complex definitions of currency have been proposed in the literature [10]; they are especially suited for specific types of data, i.e., they take into

account elements such as “volatility” of data (e.g., data such as stock price quotes that change very frequently). However, in this work, we stay with the provided general definition of currency, leaving its extensions to future work.

Currency can be measured either by associating to each value an “updating time-stamp” [11] or a “transaction time” in temporal databases [12].

**Internal Consistency.** Consistency implies that two or more values do not conflict each other. Internal consistency means that all values being compared in order to evaluate consistency are within a specific instance of a schema element.

A semantic rule is a constraint that must hold among values of attributes of a schema element, depending on the application domain modeled by the schema element. Then, internal consistency can be defined as:

*Internal Consistency is the degree to which the values of the attributes of an instance of a schema element satisfy the specific set of semantic rules defined on the schema element.*

As an example, if we consider **Citizen** with attributes **Name**, **DateOfBirth**, **Sex** and **DateOfDeath**, some possible semantic rules to be checked are:

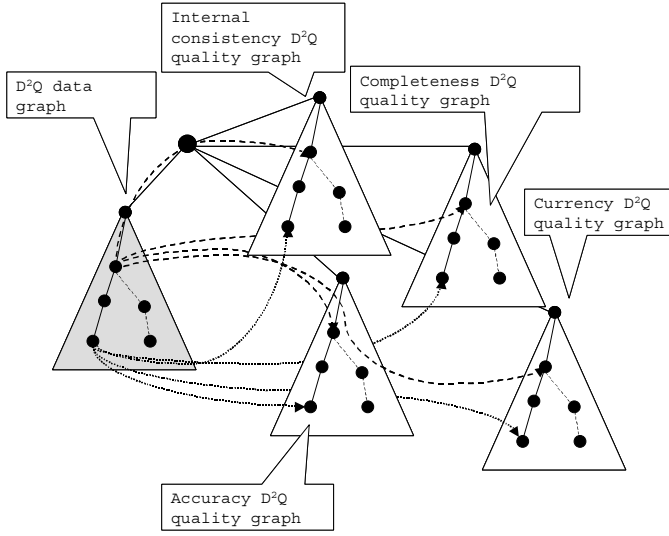
- the values of **Name** and **Sex** for an instance **p** are consistent. If **p.Name** has a value  $v = \text{JOHN}$  and the value of **p.Sex** is **FEMALE**, this is a case of low internal consistency;
- the value of **p.DateOfBirth** must precede the value of **p.DateOfDeath**.

Internal consistency has been widely investigated both in the database area through integrity constraints (e.g., [13]) and in the statistics area, through edits checking (e.g., [14]).

### 3.2 Data Model

The  $D^2Q$  model is inspired by the data model underlying XML-QL [15]. A database view of XML is adopted: an XML Document is a set of data items, and a Document Type Definition (DTD) is the schema of such data items, consisting of *data* and *quality classes*. In particular, a  $D^2Q$  XML document contains both application data, in the form of a  $D^2Q$  *data graph*, and the related data quality values, in the form of four  $D^2Q$  *quality graphs*, one for each quality dimension introduced in Section 3.1. Specifically, nodes of the  $D^2Q$  data graph are linked to the corresponding ones of the  $D^2Q$  quality graphs through links, as shown in Figure 2. Organizations in the CIS exchange each other data and quality data as  $D^2Q$  XML documents.

A  $D^2Q$  XML document corresponds to a set of conceptual data items, which are instances of conceptual schema elements; schema elements are data and quality classes, and instances are data and quality objects. Data classes and objects are straightforwardly represented as  $D^2Q$  data graphs, as detailed in the following of this section, and quality classes and objects are represented as  $D^2Q$  quality graphs, as detailed in Section 3.3.



**Fig. 2.** The generic structure of a  $D^2Q$  XML document

As a running example, consider the document `citizens.xml`, shown in Figure 3, which contains entries about citizens with the associated quality data. Such a document corresponds to a set of conceptual data items, which are instances of conceptual schema elements; schema elements are data and quality classes, and instances are data and quality objects. Specifically, an instance of `Citizen` and the related `Accuracy` values are depicted.

A data class  $\delta (\pi_1, \dots, \pi_n)$  consists of:

- a name  $\delta$ ;
- a set of properties  $\pi_i = \langle \text{name}_i : \text{type}_i \rangle, i = 1 \dots n, n \geq 1$ , where  $\text{name}_i$  is the name of the property  $\pi_i$  and  $\text{type}_i$  can be:
  - either a basic type<sup>2</sup>;
  - or a data class;
  - or a type set-of  $\langle X \rangle$ , where  $\langle X \rangle$  can be either a basic type or a data class.

We define a  $D^2Q$  data graph as follows:

A  $D^2Q$  data graph  $G$  is a graph with the following features:

- a set of nodes  $\mathcal{N}$ ; each node (i) is identified by an object identifier and (ii) is the source of 4 different links to quality objects, each one for a different

<sup>2</sup> Basic types are the ones provided by the most common programming languages and SQL, that is Integer, Real, Boolean, String, Date, Time, Interval, Currency, Any.



```

<Citizens>
  <Citizen Accuracy="o00" ... >
    <Name Accuracy="o01" ... >Maria</Name>
    <Surname Accuracy="o02" ... >Rossi</Surname>
    <ResidenceAddress Accuracy="o03" ... >
      <Street Accuracy="o04" ... >
        Via Salaria 113
      </Street>
      <City Accuracy="o05" ... >Roma</City>
      <ZIPCode Accuracy="o06" ... >
        00198
      </ZIPCode>
      <Country Accuracy="o07" ... >
        Italy
      </Country>
    </ResidenceAddress>
    <TelephoneNumber Accuracy="o08" ... >
      +390649918479
    </TelephoneNumber>
    <TelephoneNumber Accuracy="o09" ... >
      +393391234567
    </TelephoneNumber>
  </Citizen>
  <Accuracy_Citizen OID="o00">
    <Accuracy_Name OID="o01">
      0.7
    </Accuracy_Name>
    <Accuracy_Surname OID="o02">
      0.7
    </Accuracy_Surname>
    <Accuracy_ResidenceAddress OID="o03">
      <Accuracy_Street OID="o04">
        0.3
      </Accuracy_Street>
      <Accuracy_CityName OID="o05">
        0.9
      </Accuracy_CityName>
      <Accuracy_ZIPCode OID="o06">
        0.9
      </Accuracy_ZIPCode>
      <Accuracy_Country OID="o07">
        0.9
      </Accuracy_Country>
    </Accuracy_ResidenceAddress>
    <Accuracy_TelephoneNumber OID="o08">
      0.5
    </Accuracy_TelephoneNumber>
    <Accuracy_TelephoneNumber OID="o09">
      0.2
    </Accuracy_TelephoneNumber>
  </Accuracy_Citizen>
  ... ..
</Citizens>

```

**Fig. 3.** The XML document of the running example

*quality dimension. A link is a pair attribute-value, in which attribute represents the specific quality dimension for the element tag and value is an IDREF link<sup>3</sup>;*

<sup>3</sup> The use of links will be further explained in Section 3.3, when quality graphs are introduced.

- a set of edges  $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ ; each edge is labeled by a string, which represents an element tag of an XML document;
- a single root node  $\mathcal{R}$ ;
- a set of leaves; leaves are nodes that (i) are not identified and (ii) are labeled by strings, which represent element tag values, i.e., the values of the element tags labeling edges to them.

Data class instances can be represented as  $D^2Q$  data graphs, according to the following rules.

Let  $\delta(\pi_1, \dots, \pi_n)$  be a data class with  $n$  properties, and let  $\mathcal{O}$  be a data object, i.e., an instance of the data class. Such an instance is represented by a  $D^2Q$  data graph  $\mathbb{G}$  as follows:

- The root  $\mathcal{R}$  of  $\mathbb{G}$  is labeled with the object identifier of the instance  $\mathcal{O}$ .
- For each  $\pi_i = \langle \text{name}_i : \text{type}_i \rangle$  the following rules hold:
  - if  $\text{type}_i$  is a basic type, then  $\mathcal{R}$  is connected to a leaf  $lv_i$  by the edge  $\langle \mathcal{R}, lv_i \rangle$ ; the edge is labeled with  $\text{name}_i$  and the leaf  $lv_i$  is labeled with the property value  $\mathcal{O}.\text{name}_i$ ;
  - if  $\text{type}_i$  is a data class, then  $\mathcal{R}$  is connected to the  $D^2Q$  data graph which represents the property value  $\mathcal{O}' = \mathcal{O}.\text{name}_i$  by an edge labeled with  $\text{name}_i$ ;
  - if  $\text{type}_i$  is a set-of  $\langle \mathbf{X} \rangle$ , then:
    - \* let  $C$  be the cardinality of  $\mathcal{O}.\text{name}_i$ ;  $\mathcal{R}$  is connected to  $C$  elements as it follows: if (i)  $\langle \mathbf{X} \rangle$  is a basic type, then the elements are leaves (each of them labeled with a property value of the set); otherwise if (ii)  $\langle \mathbf{X} \rangle$  is a data class, then the elements are  $D^2Q$  data graphs, each of them representing a data object of the set;
    - \* edges connecting the root to the elements are all labeled with  $\text{name}_i$ .

In Figure 4, the  $D^2Q$  data graph of the running example is shown: an object instance **Maria Rossi** of the data class **Citizen** is considered. The data class has **Name** and **Surname** as properties of basic types, a property of type **set-of  $\langle \text{TelephoneNumber} \rangle$**  and another property of data class type **ResidenceAddress**; the data class **ResidenceAddress** has all properties of basic types.

### 3.3 Quality Model

So far the data portion of the  $D^2Q$  model has been described. However, organizations export XML documents containing not only data objects, but also quality data concerning the four dimensions introduced in Section 3.1.

The assumption that each organization is able and is willing to export quality values is motivated by the cooperative scenarios that we experienced; as an example, in a recent Italian *e-Government* project concerning toponymic data, each organization participating in the project has to export quality values (calculated on the basis of statistical considerations). In our architecture, a specific

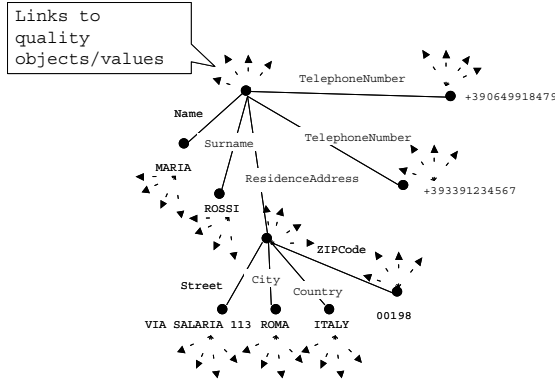


Fig. 4. The  $D^2Q$  data graph of the running example

element, i.e., the Quality Factory, is in charge of providing such values; the design of such an element is out of the scope of this paper, and has been addressed in [16].

Quality data are represented as graphs, too; they correspond to a set of conceptual quality data items, which are instances of conceptual quality schema elements; quality schema elements are referred to as *quality classes* and instances as *quality objects*. A quality class models a specific quality dimension for a specific data class: the property values of a quality object represent the quality dimension values of the property values of a data object. Therefore, each data object (i.e., node) and value (i.e., leaf) of a  $D^2Q$  data graph is linked to respectively four quality objects and values.

Let  $\delta (\pi_1, \dots, \pi_n)$  be a data class. A quality class  $\delta^D (\pi_1^D, \dots, \pi_n^D)$  consists of:

- a name  $\delta^D$ , with  $D \in \{ \text{Accuracy, Completeness, Currency, InternalConsistency} \}$ ;
- a set of tuples  $\pi_i^D = \langle \text{name}_i^D : \text{type}_i^D \rangle, i = 1 \dots n, n \geq 1$ ,

where:

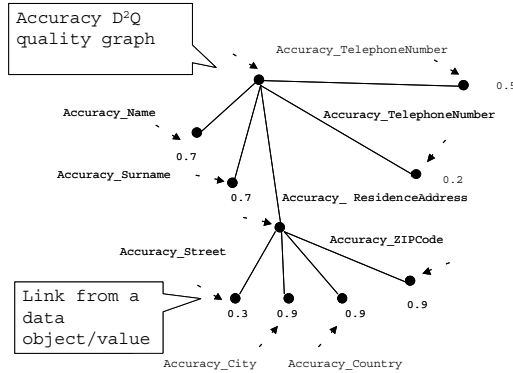
- $\delta^D$  is associated to  $\delta$  by a one-to-one relationship and corresponds to the quality dimension  $D$  evaluated for  $\delta$ ;
- $\pi_i^D$  is associated to  $\pi_i$  of  $\delta$  by a one-to-one relationship, and corresponds to the quality dimension  $D$  evaluated for  $\pi_i$ ;
- $\text{type}_i^D$  is either a basic type or a quality class or a **set-of** type, according to the structure of the data class  $\delta$ .

In order to represent quality objects, we define a  $D^2Q$  quality graph as follows:

A  $D^2Q$  quality graph  $\mathbb{G}^D$  is a  $D^2Q$  data graph with the following additional features:

- no node nor leaf is linked to any other element;
- labels of edges are strings of the form  $D\_name$  (e.g., `Accuracy_Citizen`);
- labels of leaves are strings representing quality values;
- leaves are identified by object identifiers.

A quality class instance can be straightforwardly represented as a  $D^2Q$  quality graph, on the basis on rules analogous to the ones previously presented for data objects and  $D^2Q$  data graphs. As an example, in Figure 5, the  $D^2Q$  quality graph concerning accuracy of the running example is shown, and links are highlighted; for instance, the accuracy of `Maria` is 0.7.



**Fig. 5.** The accuracy  $D^2Q$  quality graph of the running example

Data and quality data are exchanged as  $D^2Q$  XML documents in the CIS. Specifically, for each data class instance there is a  $D^2Q$  data graph linked to four  $D^2Q$  quality graphs, expressing the quality of the data objects for each dimension introduced in Section 3.1.

Let  $\{ \mathcal{O}_1, \dots, \mathcal{O}_m \}$  be a set of  $m$  objects which are instances of the same data class  $\delta$ ; a  $D^2Q$  XML document is a graph consisting of:

- a root node  $ROOT$ ;
- $m$   $D^2Q$  data graph  $\mathcal{G}_i$ ,  $i = 1 \dots m$ , each of them representing the data objects  $\mathcal{O}_i$ ;
- $4 * m$   $D^2Q$  quality graph  $\mathcal{G}_i^D$ ,  $i = 1 \dots m$ , each of them representing the quality graph related to  $\mathcal{O}_i$  concerning the quality dimension  $D$ ;
- $ROOT$  is connected to the  $m$   $D^2Q$  data graphs by edges labeled with the name of the data class, i.e.,  $\delta$ ;
- for each quality dimension  $D$ ,  $ROOT$  is connected to the  $m$   $D^2Q$  quality graph  $\mathcal{G}_i^D$  by edges labeled with the name of the quality class, i.e.,  $\delta^D$ .

The model proposed in this work adopts several graphs instead of embedding metadata within the data graph. Such a decision increases the document size,

but on the other hand allows a modular and “fit-for-all” design: (i) extending the model to new dimensions is straightforward, as it requires to define the new dimension quality graph, and (ii) specific applications, requiring only some dimension values, will adopt only the appropriate subset of the graphs.

## 4 The Data Quality Broker

The *Data Quality Broker (DQB)* is the core component of the architecture. It is implemented as a peer-to-peer distributed service: each organization hosts a copy of the Data Quality Broker that interacts with other copies. We first provide an overview of the high-level behavior of the Data Quality Broker in Sections 4.1 and 4.2; then, in Sections 4.3 and 4.4, we explain the details of its design and implementation.

### 4.1 Overview

The general task of the Data Quality Broker is to allow users to select data in the CIS according to their quality. Specifically, a user inside an organization can issue a query on a *D<sup>2</sup>Q global schema*, specifying constraints on quality values. Then, the Data Quality Broker selects, among all the copies of the requested data that are in the CIS, only those satisfying all specified constraints.

Quality constraints can be related to quality of *data* and/or quality of *service* (QoS). Specifically, besides the four data quality dimensions previously introduced, two QoS dimensions are defined, namely: (i) *responsiveness*, defined by the average round-trip delay evaluated by each organization periodically pinging gateway in the CIS; and (ii) *availability* of a source, evaluated by pinging a gateway and considering if a timeout expires; in such a case the source is considered not available for being currently queried.

A user can request data with a specified quality (e.g., “return citizens with accuracy greater than 0.7”) or declare a limit on the query processing time, restricting the query only to available sources (e.g., “return only citizens from the first responding source”) or include both kinds of constraints (e.g., “return citizens with accuracy greater than 0.7 provided by the first responding source”).

The Data Quality Broker performs two specific tasks:

- *query processing* and
- *quality improvement*.

The semantics of query processing is described in Section 4.2, and its realization is detailed in Section 4.3. The quality improvement feature consists of notifying organizations with low quality data about higher quality data that are available in the CIS. This step is enacted each time copies of the same data with different quality are collected during the query processing activity. The best quality copy is sent to all organizations having lower quality copies. Organizations involved in the query have the choice of updating or not their data. This gives a non-invasive feedback that allows to enhance the overall quality of data

in the CIS, while preserving organizations' autonomy. The details of how the improvement feature is realized are provided in Section 4.3.

## 4.2 Query Processing

The Data Quality Broker performs query processing according to a *global-as-view* (GAV) approach, by unfolding queries posed over a global schema, i.e., replacing each atom of the original query with the corresponding view on local data sources [17,2]. Both the global schema and local schemas exported by cooperating organizations are expressed according to the  $D^2Q$  model. The adopted query language is XQuery [18]. The unfolding of an XQuery query issued on the global schema can be performed on the basis of well-defined mappings with local sources. Details concerning such a mapping are beyond the scope of this paper (details can be found in [19]).

Under our hypothesis, data sources have different copies of the same data at different quality levels, i.e., there are *instance-level conflicts*. We resolve these conflicts at query execution time by relying on quality values associated to data: when a set of different copies of the same data are returned, we look at the associated quality values, and select the copy to return as a result on the basis of such values. In [20] a technique with a query processing semantics similar to the one we propose can be found; the main difference of that work with respect to our approach is that we have quality metadata associated to data that allows us to resolve conflicts; instead, in [20] conflict resolution functions are explicitly provided.

Specifically, the detailed steps we follow to answer a query with quality constraints are:

1. let  $Q$  be a query posed on the the global schema  $\mathbb{G}$ ;
2. The query  $Q$  is unfolded according to the static mapping that defines each concept of the global schema in terms of the local sources; such a mapping is defined in order to retrieve all copies of same data that are available in the CIS. Therefore, the query  $Q$  is decomposed in  $Q_1, \dots, Q_n$  queries to be posed over local sources;
3. The execution of the queries  $Q_1, \dots, Q_n$  returns a set of results  $\mathcal{R}_1, \dots, \mathcal{R}_n$ . On such a set the following property is checked: given  $\mathcal{R}_i$  and  $\mathcal{R}_j$ ,  $\mathcal{R}_i$  is *extensionally correspondent* to  $\mathcal{R}_j$ , iff for each tuple  $\mathbf{t}$  in  $\mathcal{R}_i$ , a tuple  $\mathbf{t}'$  exists in  $\mathcal{R}_j$ , such that  $\mathbf{t}$  and  $\mathbf{t}'$  are related to a same object but have inconsistent values. Similarly, for each tuple  $\mathbf{t}'$  in  $\mathcal{R}_j$ , a tuple  $\mathbf{t}$  in  $\mathcal{R}_i$  should exist that refers to the same object. This step is performed by using a well-known algorithm for record matching [21].
4. On the basis of quality constraints specified in the query, a (set of) admissible results is constructed, selecting all  $\mathcal{R}_i$  satisfying the quality constraints.

The global schema and the local schemas exported by organizations have similar structural properties, in order to compare the values of the quality descriptors of the respective data items. Wrappers in each organization (see Section

4.3) are in charge of the translation and transformation between local schemas and internal database schemas.

If data graphs are re-structured (for data integration purposes, e.g., when a new organization adheres to the CIS), the data quality graphs must be re-structured as well. In general, a GAV approach imposes some constraints on the opportunity of frequently restructuring the global schema. In the cooperative scenarios that we experienced [1], such constraints are not critical, as the CIS is built for including the most of organizations (i.e., public administrations) since the beginning, and the entrance of a new organization in the cooperative system is very rare (indeed, the creation of the CIS and the participating administrations are regulated by laws, Government directives, etc.).

### 4.3 Internal Architecture and Deployment

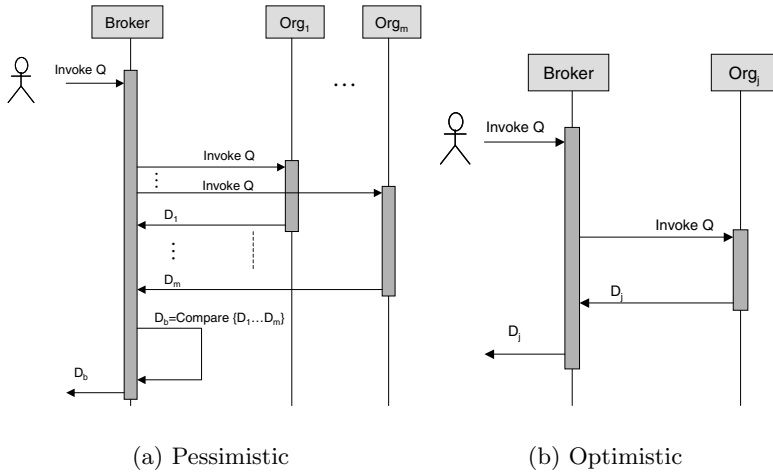
In this section we detail the design of the Data Quality Broker, by considering its interaction modes and its constituting modules; then, an example of query processing execution is shown.

The Data Quality Broker can operate either in *pessimistic* or in *optimistic* mode. In the former, the broker accesses all sources that provide data satisfying the query and builds the result, whereas in the latter the broker accesses only those organizations that *probably* will provide good quality data. Such information can be derived on the basis of statistics on quality values. In the optimistic mode, the broker does not contact all sources and does not have to wait for all responses; however, this optimization can lead to non accurate results. Conversely, in the pessimistic mode, all sources are always contacted; it takes more time to retrieve the final result, but it is always the most accurate one.

Figure 6 shows the sequence of operations performed in each of the two modes: in pessimistic mode (Figure 6(a)), all organizations storing data satisfying a query are queried and all results have to be waited for before proceeding. After results are retrieved, a comparison phase is required to choose the best data. In optimistic mode (Figure 6(b)) only the organization  $Org_j$  that has the best overall statistical quality evaluation is involved in the query and no comparison is required.

Clients of the broker can invoke one of the following operations: (i) **invoke(query, mode)**, for submitting a query  $Q$  over the global schema including quality constraints, and (ii) **propose(data)**, for notifying all sources, that have previously sent data with lower quality than the selected one, with higher quality results. Indeed each time a query is posed, different copies of same data are received as answers; the broker submits to organizations the best quality copy  $\mathcal{R}$  selected among the received ones.

Internally the Data Quality Broker is composed of five interacting modules (Figure 7). The Data Quality Broker is deployed as a peer-to-peer service: each organization hosts an independent copy of the broker and the overall service is realized through an interaction among the various copies, performed via the Transport Engine module. The modules Query Engine, Transport Engine and



**Fig. 6.** Broker invocation modes

Schema Mapper are general and can be installed without modifications in each organization. The module Wrapper has to be customized for the specific data storage system. The module Comparator can also be customized to implement different criteria for quality comparison.

*Query Engine (QE)*: it receives the query and splits the query in sub-queries (local to the sources) by interacting with the Schema Mapper (which manages the mapping between the global schema and the local schemas). Then, the *QE* also interacts with the *TE* in order to: (i) send local queries to other copies of the *QE* and receive the answers; (ii) be notified about QoS parameters to be used for optimization.

*Wrapper (Wr)*: translates the query from the language used by the broker to the one of the specific data source. In this work the wrapper is a read-only access module, that is it returns data stored inside organizations without updating them.

*Transport Engine (TE)*: it provides general connectivity among all Data Quality Broker instances in the CIS. Copies of the *TE* interact with each other in two different scenarios, corresponding to two different interaction types:

- Query execution (request/reply interaction): the requesting *TE* sends a query to the local *TE* of the target data source and remains blocked waiting for the result. A request is always synchronous but it can be always stopped from the outside, for example when the *QE* decides to block the execution of a query to enforce QoS constraints. In this case, the results are discarded.
- Quality feedback (asynchronous interaction): when a requesting *QE* has selected the best quality result of a query, it contacts the local *TE*'s to enact



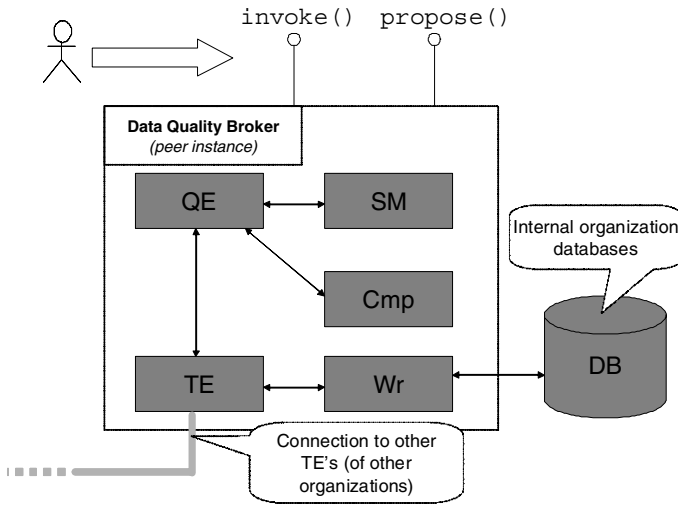


Fig. 7. Broker modules

quality feedback propagation. The **propose()** operation is executed as a call-back on each organization, with the best quality selected data as a parameter. The **propose()** can be differently implemented by each organization: a remote *TE* simply invokes this operation, whose behaviour depends from the strategy that an organization chooses for the improvement of its own data on the basis of external notifications.

The other function performed by the *TE* is the evaluation of the QoS parameters. This feature is encapsulated into the *TE* as it can be easily implemented exploiting *TE*'s communication capabilities. Specifically, the *TE* periodically pings data sources in order to calculate responsiveness. Moreover, the timeout value necessary to calculate the availability QoS parameter is configured in the *TE*.

*Schema Mapper (SM)*: stores the static mappings between the global schema and the local schemas of the various data sources.

*Comparator (CMP)*: compares the quality values returned by the different sources to choose the best one. *CMP* compares quality value vectors according to ranking methods known in literature, such as Simple Additive Weighting (SAW) [22] or Analytical Hierarchy Process (AHP) [23]

As an example, we give details of pessimistic mode query invocations, by describing the interaction among peer Data Quality Broker instances and among modules inside each Data Quality Broker when a query with quality requirements is submitted by a client.

After receiving the query, the *QE* (local to the client) interacts with the *SM* and generates a set of sub-queries, which in turn are sent through the local *TE*

to all  $TE$ 's of the involved organizations. All  $TE$ 's transfer the query to local  $QE$ 's that execute local queries by interacting with local  $Wr$ 's. Query results are passed back to the local  $TE$  that passes all results to the local  $QE$ . This module selects the best quality result through the  $CMP$  and passes it to the client. Once the best quality data is selected the feedback process starts.

This process involves all sources that have previously sent data with lower quality than the selected one; each source may adopt the proposed data (updating its local data store) or not. Indeed criteria to decide whether updating the local data store with the feedback data may vary from one organization to another; such criteria can be set by providing proper implementations of the `propose()`.

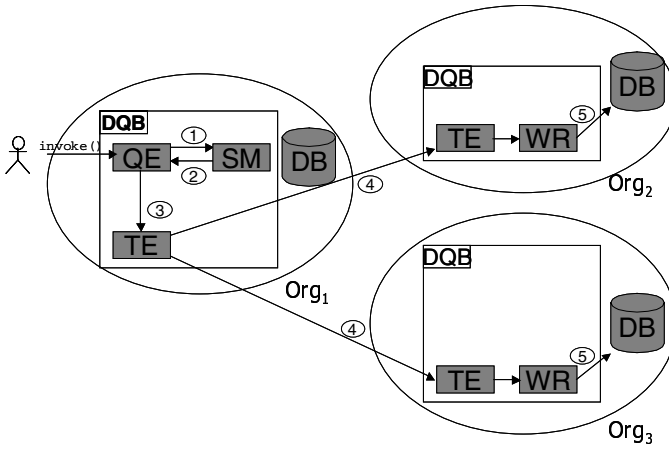
Figure 8 depicts an invocation by a client of the organization  $Org_1$ , specifically the `invoke('Select Mecella from Citizen with accuracy  $\geq 0.7$ ', PESSIMISTIC)`. For the sake of clarity, only involved modules are depicted.

The sequence of actions performed during the query invocation is described in the following. We indicate with a subscript the specific instance of a module belonging to a particular organization: for example,  $QE_i$  indicates the Query Engine of organization  $Org_i$ .

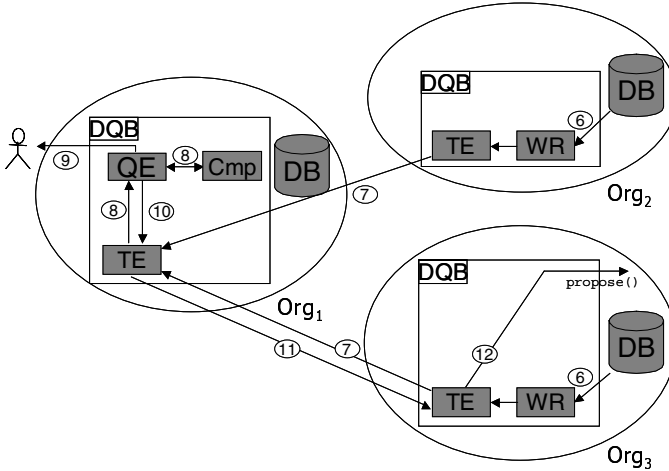
1.  $QE_1$  receives the query  $Q$  posed on the global schema, as a parameter of the `invoke()` function;
2.  $QE_1$  retrieves the mapping with local sources from  $SM_1$ ;
3.  $QE_1$  performs the unfolding that returns queries for  $Org_2$  and  $Org_3$  that are passed to  $TE_1$ ;
4.  $TE_1$  sends the request to  $Org_2$  and  $Org_3$ ;
5. The recipient  $TE$ 's pass the request to the  $Wr$ 's modules. Each  $Wr$  accesses the local data store to retrieve data;
6.  $Wr_3$  retrieves a query result, e.g.,  $\mathcal{R}_a$  with accuracy 0.7, and  $Wr_2$  retrieves another query result  $\mathcal{R}_b$  with accuracy 0.9;
7. Each  $TE$  sends the result to  $TE_1$ ;
8.  $TE_1$  sends the collected results ( $\mathcal{R}_a, \mathcal{R}_b$ ) to  $QE_1$ .  $QE_1$  selects through  $CMP_1$  the result with the greatest accuracy, i.e.,  $Org_2$ 's result ( $\mathcal{R}_b$  with accuracy 0.9);
9.  $QE_1$  sends the selected result to the client;
10.  $QE_1$  starts the feedback process sending the selected result ( $\mathcal{R}_b$  with accuracy 0.9) to  $TE_1$ ;
11.  $TE_1$  sends it to  $Org_3$ ;
12.  $TE_3$  receives the feedback data and makes a call `propose( $\mathcal{R}_b$  with accuracy 0.9)`.

#### 4.4 Implementation

We have developed a prototype of the broker [19,24], implementing the pessimistic query invocation mode. In our prototype, the Data Quality Broker is realized as a Web Service [25], deployed on all the organizations. A Web Service-based solution is suitable for integrating heterogeneous organizations as it allows



(a) Step 1 – 5



(b) Step 6 – 12

**Fig. 8.** Details of query invocation in the pessimistic mode

to deploy the Data Quality Broker on the Internet, regardless from the access limitations imposed by firewalls, interoperability issues, etc. Moreover, by exploiting standard Web Service technologies, the technological integration among different organizations is straightforward; specifically, we tested interoperability among different versions of the Data Quality Broker, realized using respectively the JAX-RPC<sup>4</sup> framework and the .NET one<sup>5</sup>.

<sup>4</sup> <http://java.sun.com/xml/jaxrpc/>

<sup>5</sup> <http://msdn.microsoft.com/netframework/>

As we discussed above, the component responsible for the communication between the Data Quality Broker instances is the Transport Engine. Besides the operations previously described (i.e., `propose()` and `invoke()`), it exposes also the `ping()` operation, which is used to estimate the query responsiveness. Such operations are invoked through standard SOAP messages sent over HTTP; specifically, invocations to multiple Transport Engines are performed in parallel. Remote invocations to the `invoke()` operation require the client Transport Engine to wait for all results.

If we assume that Data Quality Broker instances are subject to faults this can cause the client to wait indefinitely for responses from faulty instances. From the distributed system theory stems that in communication systems, such as the Internet, where delays on message transfer cannot be predicted, it is impossible to distinguish a faulty software component from a very slow one [26,27]. Then, the client Transport Engine sets a timeout for each invoked source. The timeout value is estimated for each single source on the basis of its responsiveness. After the timeout expires, no result is returned for that query. This is a best-effort semantics that does not guarantee to always return the best quality value but rather enforces the correct termination of each query invocation. Replication techniques can be exploited in order to enhance the availability of each web service, ensuring more accurate results even in presence of faults [28].

## 5 Related Work

Data Quality has been traditionally investigated in the context of single information systems. Only recently, there is a growing attention towards data quality issues in the context of multiple and heterogeneous information systems [29,30,31].

In cooperative scenarios, the main data quality issues regard [30]: *(i)* assessment of the quality of the data owned by each organization; *(ii)* methods and techniques for exchanging quality information; *(iii)* improvement of quality within each cooperating organization; and *(iv)* heterogeneity, due to the presence of different organizations, in general with different data semantics.

For the assessment *(i)* issue, some of the results already achieved for traditional systems can be borrowed, e.g., [32,33].

Methods and techniques for exchanging quality information *(ii)* have been only partially addressed in the literature. In [29], an algorithm to perform query planning based on the evaluation of data sources' quality, specific queries' quality and query results' quality is described. The approach of [29] is different from our approach mainly because we assume to have quality metadata associated to each quality value: this gives us the possibility of more accurate answers in selecting data on the basis of quality. Specifically, the granularity of quality data in our framework is finer than the one proposed in [29], as our framework is also targeted to improvement actions.

When considering the issue of exchanging data and the associated quality, a model to export both data and quality data needs to be defined. Some concep-

tual models to associate quality information to data have been proposed that include an extension of the Entity-Relationship model [34], and a data warehouse conceptual model with quality features described through the Description Logic formalism [33]. Both models are thought for a specific purpose: the former to introduce quality elements in relational database design; the latter to introduce quality elements in the data warehouse design. Whereas, in the present paper the aim is to enable quality exchanging in a generic CIS, independently of the specific data model and system architecture. Moreover, previous models are not well suited for semi-structured data.

Our work is similar to the extensions of the relational model described in [35], chapters 2 and 3; in particular, the attribute-based model can be considered as a possible mechanism for effectively storing data and quality data inside organizations (by assuming that back-end systems are on relational database), to be then converted and exported as  $D^2Q$  documents.

In [36], the problem of the quality of web-available information has been faced in order to select data with high quality coming from distinct sources: every source has to evaluate some pre-defined data quality parameters, and to make their values available through the exposition of metadata. Our proposal is different as we propose an ad-hoc service that brokers data requests and replies on the basis of data quality information. Moreover, we also take into account improvement features (*iii*) that are not considered in [36].

The heterogeneity (*iv*) issues are taken into account by data integration works. Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [2]. As described in [20], when performing data integration two different types of conflicts may arise: *semantic conflicts*, due to heterogeneous source models, and *instance-level conflicts*, due to what happens when sources record inconsistent values on the same objects. The Data Quality Broker described in this paper is a system solving instance-level conflicts.

Other notable examples of data integration systems within the same category are AURORA [20] and the system described in [37]. AURORA supports conflict tolerant queries, i.e. it provides a dynamic mechanism to resolve conflicts by means of defined conflict resolution functions. The system described in [37] describes how to solve both semantic and instance-level conflicts. The proposed solution is based on a multidatabase query language, called FraQL, which is an extension of SQL with conflict resolution mechanisms. Similarly to both such systems, the Data Quality Broker supports dynamic conflict resolution, but differently from them the Data Quality Broker relies onto quality metadata for solving instance-level conflicts.

A system that also takes into account metadata (i.e., not necessarily and specifically regarding quality of data) for instance-level conflict resolution is described in [38]; such a system adopts the ideas of the COIN framework [39]: context dependent and independent conflicts are distinguished and accordingly to this very specific direction, conversion rules are discovered on pairs of systems.

Moreover, differently from the COIN prototype, the Data Quality Broker is not a centralized system, but it is based on a distributed architecture.

## 6 Conclusions and Future Work

Managing data quality in CIS's merges issues from many research areas of computer science such as databases, software engineering, distributed computing, security, and information systems. This implies that the proposal of integrated solutions is very challenging. In this paper, an overall architecture to support data quality management in CIS's has been proposed. The specific contribution of the paper are (i) a model for data and quality data exported by cooperating organizations, and (ii) the design of a service for querying and improving quality data and.

Up to now, the pessimistic mode of the Data Quality Broker has been implemented. We plan to investigate in future work the research issues described throughout the paper concerning the optimistic invocation mode. We are currently investigating techniques for query optimization based on both data quality parameters and QoS parameters. The optimization based on data quality parameters can be performed on the basis of statistics on quality values. As an example, statistic values could be calculated on current data exported by organizations (e.g., "the medium accuracy of citizens currently provided by organization  $\mathcal{A}$  is high"). As another example, statistic values could be calculated on the basis of past performances of organizations in providing good quality data (e.g., "the accuracy of citizens provided by organization  $\mathcal{A}$  in the last  $N$  queries is high"). In such a way, the number of possible query answers is reduced by eliminating those that may not be conform to quality requirements according to the statistical evaluation.

The complete development of a framework for data quality management in CIS's requires the solution of further issues. An important aspect concerns the techniques to be used for quality dimension measurement. In both statistical and machine learning areas, some techniques could be usefully integrated in the proposed architecture. The general idea is to give to each data value a quality evaluation with a certain estimated probability, instead of a deterministic quality evaluation. In this way, the task of assigning quality values to each data value could be considerably simplified.

**Acknowledgments.** The authors would like to thank all people involved in the DAQUINCIS project, in particular Sara Tucci Piergiovanni, the anonymous reviewers for their useful suggestions, and Maurizio Lenzerini, Domenico Lembo and Carlo Marchetti for interesting discussions on some of the topics discussed in the paper.

This work is supported by MIUR, COFIN 2001 Project "DaQuinCIS - Methodologies and Tools for Data Quality in Cooperative Information Systems" (<http://www.dis.uniroma1.it/~dq/>).

## References

1. C. Batini and M. Mecella, "Enabling Italian e-Government Through a Cooperative Architecture," *IEEE Computer*, vol. 34, no. 2, 2001.
2. M. Lenzerini, "Data Integration: A Theoretical Perspective," in *Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS 2002)*, Madison, Wisconsin, USA, 2002.
3. C. Marchetti, M. Mecella, M. Scannapieco, A. Virgillito, and R. Baldoni, "Data Quality Notification in Cooperative Information Systems," in *Proceedings of the First International Workshop on Data Quality in Cooperative Information Systems*, Siena, Italy, 2003.
4. L. De Santis, M. Scannapieco, and T. Catarci, "A Trust Model for Tightly Coupled P2P Systems," in *Proceedings of the 11<sup>o</sup> Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD 2003)*, Cetraro (CS), Italy, 2003.
5. P. Bertolazzi, L. De Santis, and M. Scannapieco, "Automatic Record Matching in Cooperative Information Systems," in *Proceedings of the ICDT'03 International Workshop on Data Quality in Cooperative Information Systems (DQCIS'03)*, Siena, Italy, 2003.
6. T.C. Redman, *Data Quality for the Information Age*, Artech House, 1996.
7. Y. Wand and R.Y. Wang, "Anchoring Data Quality Dimensions in Ontological Foundations," *Communications of the ACM*, vol. 39, no. 11, 1996.
8. P.A.V. Hall and G.R. Dowling, "Approximate String Matching," *ACM Computing Surveys*, vol. 12, no. 4, 1980.
9. L.L. Pipino, Y.W. Lee, and R.Y. Wang, "Data Quality Assessment," *Communications of the ACM*, vol. 45, no. 4, 2002.
10. D.P. Ballou, R.Y. Wang, H. Pazer, and G.K. Tayi, "Modeling Information Manufacturing Systems to Determine Information Product Quality," *Management Science*, vol. 44, no. 4, 1998.
11. P. Missier, M. Scannapieco, and C. Batini, "Cooperative Architectures: Introducing Data Quality," Technical Report 14-2001, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy, 2001.
12. A. Tansell, R. Snodgrass, J. Clifford, S. Gadia, and A. Segev (eds.), *Temporal Databases*, Benjamin-Cummings, 1993.
13. E.F. Codd, "Relational Database: a Practical Foundation for Productivity (1981 ACM Turing Award Lecture)," *Communications of the ACM*, vol. 25, no. 2, 1982.
14. R. Bruni and A. Sassano, "Errors Detection and Correction in Large Scale Data Collecting," in *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, Cascais, Portugal, 2001.
15. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu, "XML-QL: A Query Language for XML," in *Proceedings of the 8th International World Wide Web Conference (WWW8)*, Toronto, Canada, 1999.
16. C. Cappiello, C. Francalanci, B. Pernici, P. Plebani, and M. Scannapieco, "Data Quality Assurance in Cooperative Information Systems: a Multi-dimension Quality Certificate," in *Proceedings of the ICDT'03 International Workshop on Data Quality in Cooperative Information Systems (DQCIS'03)*, Siena, Italy, 2003.
17. J.D. Ullman, "Information Integration Using Logical Views," in *Proceedings of the Sixth International Conference on Database Theory (ICDT '97)*, Delphi, Greece, 1997.
18. S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simèon, "XQuery 1.0: An XML Query Language," W3C Working Draft, November 2002.

19. D. Milano, *Progetto DaQuinCIS: Estensione di XQuery e Query processing in un Sistema di Integrazione Basato sulla Qualità dei Dati*, Tesi di Laurea in Ingegneria Informatica, Università di Roma "La Sapienza", Facoltà di Ingegneria, 2003 (in Italian) (the thesis is available by writing an e-mail to: [monscan@dis.uniroma1.it](mailto:monscan@dis.uniroma1.it)).
20. L.L. Yan and M.T. Ozsu, "Conflict Tolerant Queries in AURORA," in *Proceedings of the Fourth International Conference on Cooperative Information Systems (CoopIS'99)*, Edinburgh, Scotland, UK, 1999.
21. M.A. Hernandez and S.J. Stolfo, "Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem," *Journal of Data Mining and Knowledge Discovery*, vol. 1, no. 2, 1998.
22. C. Hwang and K. Yoon, *Multiple Attribute Decision Making*, Lectures Notes in Economics and Mathematical Systems-Springer Verlag 186, 1981.
23. T.L. Saaty, *The Analytic Hierarchy Process*, McGraw-Hill, 1980.
24. G. Palmieri, *Progetto DaQuinCIS: Architettura Basata su Tecnologie Web Service ed Ottimizzazione di Query XML*, Tesi di Laurea in Ingegneria Informatica, Università di Roma "La Sapienza", Facoltà di Ingegneria, 2003 (in Italian) (the thesis is available by writing an e-mail to: [monscan@dis.uniroma1.it](mailto:monscan@dis.uniroma1.it)).
25. A. Buchmann, F. Casati, L. Fiege, M.C. Hsu, and M.C. Shan, Eds., *Proceedings of the 3rd VLDB International Workshop on Technologies for e-Services (VLDB-TES 2002)*, Hong Kong, China, 2002.
26. T.D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM (JACM)*, vol. 43, no. 2, pp. 225–267, 1996.
27. M.J. Fischer, N.A. Lynch, and M.S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.
28. R. Guerraoui and A. Schiper, "Software-Based Replication for Fault Tolerance," *IEEE Computer*, vol. 30, no. April 1997, 1997.
29. F. Naumann, U. Leser, and J.C. Freytag, "Quality-driven Integration of Heterogeneous Information Systems," in *Proceedings of 25th International Conference on Very Large Data Bases (VLDB'99)*, Edinburgh, Scotland, UK, 1999.
30. P. Bertolazzi and M. Scannapieco, "Introducing Data Quality in a Cooperative Context," in *Proceedings of the 6th International Conference on Information Quality (IQ'01)*, Boston, MA, USA, 2001.
31. L. Berti-Equille, "Quality-Extended Query Processing for Distributed Processing," in *Proceedings of the ICDT'03 International Workshop on Data Quality in Cooperative Information Systems (DQCIS'03)*, Siena, Italy, 2003.
32. H. Galhardas, D. Florescu, D. Shasha, and E. Simon, "An Extensible Framework for Data Cleaning," in *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*, San Diego, CA, USA, 2000.
33. M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, Eds., *Fundamentals of Data Warehouses*, Springer Verlag, 1999.
34. R.Y. Wang, H.B. Kon, and S.E. Madnick, "Data Quality Requirements: Analysis and Modeling," in *Proceedings of the 9th International Conference on Data Engineering (ICDE '93)*, Vienna, Austria, 1993.
35. R.Y. Wang, M. Ziad, and Y.W. Lee, *Data Quality*, Kluwer Academic Publisher, 2001.
36. G. Mihaila, L. Raschid, and M. Vidal, "Querying Quality of Data Metadata," in *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.



37. K. Sattler, S. Conrad, and G. Saake, "Interactive Example-driven Integration and Reconciliation for Accessing Database Integration," *Information systems*, vol. 28, 2003.
38. K. Fan, H. Lu, S.E. Madnick, and D. Cheung, "Discovering and Reconciling Value Conflicts for Numerical Data Integration," *Information systems*, vol. 28, 2003.
39. "The COntext INterchange (COIN) Project," <http://context.mit.edu/~coin/>, 1996–1999.

# Table of Contents

Formal Reasoning Techniques for Goal Models .....	1
<i>Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, Roberto Sebastiani</i>	
Attribute-Based Semantic Reconciliation of Multiple Data Sources .....	21
<i>Jeffrey Parsons, Yair Wand</i>	
Data Quality in Web Information Systems .....	48
<i>Barbara Pernici, Monica Scannapieco</i>	
Reasoning about Anonymous Resources and Meta Statements on the Semantic Web .....	69
<i>Guizhen Yang, Michael Kifer</i>	
IF-Map: An Ontology-Mapping Method Based on Information-Flow Theory .....	98
<i>Yannis Kalfoglou, Marco Schorlemmer</i>	
OntoEdit: Multifaceted Inferencing for Ontology Engineering .....	128
<i>York Sure, Juergen Angele, Steffen Staab</i>	
Distributed Description Logics: Assimilating Information from Peer Sources .....	153
<i>Alex Borgida, Luciano Serafini</i>	
On Using Conceptual Data Modeling for Ontology Engineering .....	185
<i>Mustafa Jarrar, Jan Demey, Robert Meersman</i>	
The DAQUINCIS Broker: Querying Data and Their Quality in Cooperative Information Systems .....	208
<i>Massimo Mecella, Monica Scannapieco, Antonino Virgillito, Roberto Baldoni, Tiziana Catarci, Carlo Batini</i>	
<b>Author Index</b> .....	233



# Author Index

Angele, Juergen 128

Baldoni, Roberto 208

Batini, Carlo 208

Borgida, Alex 153

Catarci, Tiziana 208

Demey, Jan 185

Giorgini, Paolo 1

Jarrar, Mustafa 185

Kalfoglou, Yannis 98

Kifer, Michael 69

Mecella, Massimo 208

Meersman, Robert 185

Mylopoulos, John 1

Nicchiarelli, Eleonora 1

Parsons, Jeffrey 21

Pernici, Barbara 48

Scannapieco, Monica 48, 208

Schorlemmer, Marco 98

Sebastiani, Roberto 1

Serafini, Luciano 153

Staab, Steffen 128

Sure, York 128

Virgillito, Antonino 208

Wand, Yair 21

Yang, Guizhen 69