

CONFERENCES IN RESEARCH AND PRACTICE IN  
INFORMATION TECHNOLOGY

VOLUME 72

# ADVANCES IN ONTOLOGIES 2006



AUSTRALIAN  
COMPUTER  
SOCIETY



# ADVANCES IN ONTOLOGIES 2006

Proceedings of the  
Second Australasian Ontology Workshop (AOW 2006),  
Hobart, Australia, 5 December 2006

Mehmet A. Orgun and Thomas Meyer, Eds.

Volume 72 in the Conferences in Research and Practice in Information Technology Series.  
Published by the Australian Computer Society Inc.



Published in association with the ACM Digital Library.

**Proceedings of the Second Australasian Ontology Workshop (AOW 2006), Hobart, Australia,  
5 December 2006**

**Conferences in Research and Practice in Information Technology, Volume 72.**

Copyright ©2006, Australian Computer Society. Reproduction for academic, not-for profit purposes permitted provided the copyright text at the foot of the first page of each paper is included.

Editors:

Mehmet A. Orgun  
Intelligent Systems Group (ISG)  
Department of Computing  
Macquarie University  
Sydney, NSW 2109, Australia  
E-mail: Mehmet.Orgun@mq.edu.au

Thomas Meyer  
Knowledge Representation and Reasoning Program  
Neville Roach Laboratory, Sydney  
National ICT Australia  
223 Anzac Parade  
Kensington, NSW 2052, Australia  
E-mail: Thomas.Meyer@nicta.com.au

Series Editors:

Vladimir Estivill-Castro, Griffith University, Queensland  
John F. Roddick, Flinders University, South Australia  
Simeon Simoff, University of Technology, Sydney, NSW  
crpit@infoeng.flinders.edu.au

Publisher: Australian Computer Society Inc.  
PO Box Q534, QVB Post Office  
Sydney 1230  
New South Wales  
Australia.

Conferences in Research and Practice in Information Technology, Volume 72.  
ISSN 1445-1336.  
ISBN 1-920-68253-8.

Printed, November 2006 by Flinders Press, PO Box 2100, Bedford Park, SA 5042, South Australia.  
Cover Design by Modern Planet Design, (08) 8340 1361.

The *Conferences in Research and Practice in Information Technology* series aims to disseminate the results of peer-reviewed research in all areas of Information Technology. Further details can be found at <http://crpit.com/>.

# Table of Contents

## Proceedings of the Second Australasian Ontology Workshop (AOW 2006), Hobart, Australia, 5 December 2006

Preface .....	vii
Programme Committee .....	viii
Acknowledgement of Support .....	ix

### Keynote Paper

The Semantics of Water .....	3
<i>Kerry Taylor</i>	

### Full Papers

A Process for Building a Domain Ontology: an Experience in Developing a Government Budgetary Ontology .....	7
<i>Graciela Brusa, María Laura Caliusco and Omar Chiotti</i>	
Trust Based Ontology Integration For The Community Services Sector .....	17
<i>Dennis Hooijmaijers and Markus Stumptner</i>	
Mephisto I. Towards a Formal Theory .....	25
<i>Dale Lambert and Chris Nowak</i>	
Towards Scalable Ontology Engineering Patterns: Lessons Learned from an Experiment based on W3Cs Part-whole Guidelines .....	31
<i>Laurent Lefort, Kerry Taylor and David Ratcliffe</i>	
Approaches for Semantic Interoperability between Domain Ontologies .....	41
<i>Bhavana Orgun, Mark Dras, Abhaya Nayak and Geoff James</i>	
Aspects of Automatic Ontology Extension: Adapting and Regeneralizing Dynamic Updates .....	51
<i>Ekaterina Ovchinnikova and Kai-Uwe Kühnberger</i>	
Metonymic, and Holonymic roles and Emergent Properties in the SNOMED CT Ontology .....	61
<i>Jon Patrick</i>	
Towards Semantic Interoperability in Healthcare: Ontology Mapping from SNOMED-CT to HL7 version 3 .....	69
<i>Amanda Ryan</i>	
Annotating Websites with Machine-processable Information in Controlled Natural Language .....	75
<i>Rolf Schwitter and Marc Tilbrook</i>	
A Native Ontology Approach for Semantic Service Descriptions .....	85
<i>Rajesh Thiagarajan and Markus Stumptner</i>	
Semantic Enrichment in Ontologies for Matching .....	91
<i>Nwe Ni Tun</i>	
Author Index .....	101



## Preface

Following the successful inaugural Australasian Ontology Workshop (AOW 2005) which was held in Sydney, NSW, Australia on 6 December 2005, AOW 2006 took place in Hobart, Tasmania, Australia on 5 December 2006, in conjunction with the 19th Australian Joint Conference on Artificial Intelligence (AI'06).

The use of formal ontologies in knowledge systems has many advantages. It allows for an unambiguous specification of the structure of knowledge in a domain, enables knowledge sharing and, as a result, makes it possible to perform automated reasoning about ontologies. In recent years there has been a worldwide increase in the use of ontologies, both in industry and in research laboratories. In parallel to the growth and interest in ontology research, there is also a growing community of researchers in Australia and New Zealand, working on various aspects of ontologies.

The purpose of this one-day workshop series on Advances in Ontologies is to bring together ontology researchers from both industry and academia in the Australasian region for interaction, discussion, sharing of results and initiation of new projects, and also to raise the awareness of the Australasian Artificial Intelligence community to the state-of-the-art ontology research conducted in the region. AOW 2006 has in particular provided a visible focal point for ontology research within the Australasian region, and connection with the international ontology community.

The keynote speaker, Kerry Taylor of the CSIRO ICT Centre, provided useful insights into why ontologies are an important part of the water information management solution, in addressing the pressing need for responsible management of Australia's scarce water resource. She also touched on the ongoing work in the ICT Centre which hosts the Australian Office for the World Wide Web Consortium (W3C Australia 2006).

A program committee of international standing reviewed all contributed papers (full papers were reviewed). Each paper was reviewed by three program committee members and additional reviews were also sought to identify those papers which propose the most promising ideas. As a result, eleven papers were selected for publication in these proceedings out of eighteen submitted papers involving authors from Argentina, Australia, Germany, Iran, Japan, Republic of Korea and Singapore.

The papers deal with all aspects of ontology research, including the theoretical foundations, description logics, design, implementation and prototype development issues, standards such as HL7, ontologies in multi-agent systems and the Semantic Web, comparative studies, and applications and development of real-world ontologies, as well as those describing new challenges arising out of applications.

We would like to thank the keynote speaker, Kerry Taylor, the authors and the members of the Program Committee of AOW 2006 and the additional reviewers for their contributions to the quality of the workshop and of this collection.

Thanks are also due to Peter Vampley, AI'06 workshop chair, for his help with the smooth organisation of this workshop event, and John Roddick, one of the editors of the CRPIT series, for facilitating the publication of the AOW 2006 workshop proceedings. We acknowledge the EasyChair conference management system which was used in all stages of the paper submission and review process and also in the collection of the final camera-ready papers.

Mehmet A. Orgun, Macquarie University  
Thomas Meyer, NICTA  
Organisers of AOW 2006  
December, 2006

# Programme Committee

## Programme Chairs

Mehmet A. Orgun (Macquarie University)  
Thomas Meyer (NICTA)

## Programme Committee

Mike Bain (UNSW)  
Robert Barta (Bond University)  
Richard Booth (Mahasarakham University)  
Werner Ceusters (SUNY Buffalo)  
Stephen Cranefield (University of Otago)  
Anne Cregan (UNSW and NICTA)  
Costas Mantratzis (University of Westminster)  
Abhaya Nayak (Macquarie University)  
Bhavna Orgun (Macquarie University)  
Maurice Pagnucco (UNSW)  
Debbie Richards (Macquarie University)  
Abdul Sattar (Griffith University)  
Rolf Schwitter (Macquarie University)  
Barry Smith (SUNY Buffalo)  
Markus Stumptner (University of South Australia)  
Kerry Taylor (CSIRO)  
Mary-Anne Williams (UTS)

## Additional Reviewers

Steve Cassidy (Macquarie University)  
Mark Dras (Macquarie University)  
Vadim Gerasimov (CSIRO)  
Quentin Reul (University of Aberdeen)



## Acknowledgement of Support

We wish to thank the National ICT Australia Limited and Macquarie University for their continuing support that made the organisation of this workshop possible.



---

National ICT Australia Limited  
Locked Bag 9013  
Alexandria, NSW 1435  
<http://www.nicta.com.au>



Macquarie University  
Sydney, NSW 2109, Australia  
<http://www.mq.edu.au/>



# KEYNOTE PAPER



## The semantics of water

**Kerry Taylor**

CSIRO ICT Centre  
GPO Box 664,  
Canberra ACT, 2601.  
[kerry.taylor@csiro.au](mailto:kerry.taylor@csiro.au)

### Abstract

The pressing need for responsible management of Australia's scarce water resources is now front page news. There are many challenging problems to be solved to meet this need. For example the science to understand the link between surface and groundwater flows, the monitoring to determine whether regional catchment projects are effective, the policy framework to balance agricultural and environmental benefits, the social and political will to establish efficient water markets, and the science that links climate change to water availability.

Underlying and unifying all these challenges is the crying need for information to support the development of knowledge and decision making. Current policy and regulations for water distribution cannot even be monitored for compliance, let alone for continuous improvement. In Australia, the information on water availability at a location is held by dozens of agencies, from irrigators, land managers, regional catchment management boards, water supply agencies, national commissions, state government departments, shire councils, and scientific agencies. A great deal of this data is not empirical, but simulated according to the best science and data available at the time of simulation. And plenty of it doesn't exist at all. Advances in sensor network technology are being sought to improve both the spatial and temporal resolution of data, as well as its currency. Advances in data management, analysis and visualisation are being sought to make sense of the highly heterogeneous data. A large scale multi-agency national resource for water information is envisaged (Water Resources Observation Network 2006).

In this talk I will give a personal view (probably uncontroversial to this audience!) on why ontologies are an important part of the water information management solution. It has been recently acknowledged in the Australian healthcare industry that they are crucial for information interoperability amongst many independent agencies, but in this area they are also important for furthering the scientific advances needed for hydrology and ecology. I will survey relevant international work in the area including our own work in CSIRO, and identify

some of the research challenges arising.

The CSIRO ICT Centre hosts the Australian Office for the World Wide Web Consortium (W3C Australia 2006). I will also advertise the current W3C activities that would be of interest to researchers and practitioners working on ontologies.

### References

W3C Australia (2006), <http://w3c.org.au>

Water Resources Observation Network (2006), <http://wron.net.au>



# FULL PAPERS





# A Process for Building a Domain Ontology: an Experience in Developing a Government Budgetary Ontology

Graciela Brusa

Dirección Provincial de Informática  
San Martín 2466, Santa Fe (Santa Fe)  
Argentina

[gracielabrusa@santafe.gov.ar](mailto:gracielabrusa@santafe.gov.ar)

Ma. Laura Caliusco

CIDISI, CONICET-UTN-FRSF,  
Lavaise 610, Santa Fe (Santa Fe)  
Argentina

[mcaliusc@frsf.utm.edu.ar](mailto:mcaliusc@frsf.utm.edu.ar)

Omar Chiotti

INGAR, CONICET-UTN-FRSF,  
Avellaneda 3657, Santa Fe (Santa Fe)  
Argentina

[chiotti@ceride.gov.ar](mailto:chiotti@ceride.gov.ar)

## Abstract

During the last years, there has been a growing concern on ontology due to its ability to explicitly describe data semantics in a common way, independently of data source characteristics, providing a schema that allows data interchanging among heterogeneous information systems and users. Several works have been aimed to improve ontology technological aspects, like representation languages and inference mechanisms, and less attention has been paid to practical results of development method application. This paper presents a discussion on the process and product of an experience in developing ontology for the Public Sector whose organization requires a strong knowledge management. Particularly, this process was applied to develop ontology for Budget Domain.

**Keywords:** ontology engineering, development methodology.

## 1 Introduction

Since an ontology has gained recognition from academy and industry, there are several definitions about what an ontology is. These definitions came from different disciplines and have been used for different purposes. In information science, an ontology can be seen as a dictionary of terms formulated in a canonical syntax and with commonly accepted definitions designed to yield a lexical or taxonomical framework for knowledge-representation which can be shared by different information systems communities (Smith, 2003). In order to define a complete commonly accepted definition, an agreement must be reached. This agreement has to follow a comprehensive ontology engineering process.

There are several mature methodologies that have been proposed to structure this process and thus to facilitate it. Moreover, the success of these methodologies has been demonstrated in a number of applications (Corcho et al, 2005). Nevertheless, the ontology development in some

areas has not been as expected. One example is the public sector area, which is characterized by a wide range of task and work arrangements. Some process can be fully automated but its scope is limited to simple processes of registering, accounting and calculating. Processes in which stakeholders participate because legal rules and knowledge play an important role are much more important (Klichewski, 2002).

Besides, decision-making in public administration occurs at organizational or policy level but it is also characteristic of its operative work. Thereby, public agents must be able to access information and knowledge to help their tasks.

The objective of this paper is to share with the ontology community the process followed to develop an ontology for Budgetary Domain. To this aim, this work is organized as follows: Section 2 discusses the main ontology development methodologies. Section 3 shows how we have adapted different methodologies to define the budgetary domain ontology. Section 4 discusses the implementation of the ontology. Section 5 presents a summary of results. Finally, Section 6 is devoted to the conclusions of this work.

## 2 Ontology Development Methodologies

Before starting to define the ontology, different development methodologies were studied (Wache et al, 2001). From this study, two main groups can be identified. On the one hand, there are experience-based methodologies, such as the methodology proposed for Gruninger and Fox (1995), based on TOVE Project or the other exposed by Uschold and King (1996) (Uschold & Gruninger, 1996) from Enterprise Model. Both were issued in 1995 and belong to the enterprise modeler domain. On the other hand, there are methodologies that propose evolutive prototypes models, such as METHONTOLOGY (Gómez-Pérez et al, 2004) that proposes a set of activities to develop ontologies based on its life cycle and the prototype refinement; and 101 Method (Noy & McGuinness, 2001) that proposes an iterative approach to ontology development.

On the one hand, there is not just one correct way or methodology for developing ontologies. Usually, the first ones are applied when the requirements are clearly known at the beginning; the second ones when the objectives are not clear from the beginning. Moreover, it is common to merge different methodologies since each of them provides design ideas that distinguish it from the

others. This merging depends on the ontology users and ontology goals.

On the other hand, like any other conceptual modeling activity, ontology construction must be supported by software engineering techniques (Falbo, 2004). Thus, we used methods and tools from software engineering to support ontology engineering activities.

In general terms, the ontology development can be divided into two main phases: specification and conceptualization. The goal of the specification phase is to acquire informal knowledge about the domain. The goal of the conceptualization phase is to organize and structure this knowledge using external representations that are independent of the implementation languages and environments. The objective of the next section is to show how we have adapted different ontology development methodologies to define the specification and conceptualization phases. Furthermore, it shows how different software engineering techniques were used to define different representations during these phases.

### 3 Building Government Ontology for Budgetary Domain

This section describes the process of an experience in developing a Government Ontology for Budgetary Domain.

#### 3.1 Specification: The Ontology Goal and Scope

The definition of ontology goal and scope was considered the first step in this study case as it is proposed in 101 Method and in the first activity in METHONTOLOGY.

The scope limits the ontology, specifying what must be included and what must not. It is an important step for minimizing the amount of data and concepts to be analyzed, especially for the extent and complexity of the budgetary semantics. In successive iterations for verification process, it will be adjusted if necessary.

This ontology only considers the needs for creating an analytic budget with concepts related to expenses. It does not consider the concepts related to other stages as budgetary executing, accounting, payments, purchases or fiscal year closure. Therefore, it includes general concepts for the budget life cycle and specific concepts for the formulation.

#### 3.2 Specification: Domain Description

Taking into account that this work was made from scratch and that 101 METHOD proposes the enumeration of important terms to continue as well as METHONTOLOGY plans to use intermediate representations for organizing knowledge domain in the conceptualization phase (Gómez-Pérez et al, 2004), it was necessary to make a previous domain analysis.

In this analysis, the application to formulating the provincial budget and its related documentations were studied and revised. Furthermore, meetings with a group

of experts were carried out. This group was conformed by public officials responsible for the whole budget formulation process in the Executive Power, expert professionals of Budget Committee in Legislative Power, public agents of the administrative area in charge of creating their own budget, and software engineers who bring informatics supports for these tasks. As it can be seen, the group of experts was very heterogeneous. In addition, they do not have much time to assign the meetings. This group was the support for knowledge acquisition during the ontology development. Then, we have to define different intermediate representations to communicate the knowledge acquired to the experts considering the background of each one and the time of meetings.

Following, a brief description of the domain is presented.

#### 3.2.1 Budgetary and Financial Domain

The budget of a government is a plan of the intended revenues and expenditures of that government. The budget is prepared by different entities in different government areas. Particularly, in Santa Fe Province (Argentina) these entities are:

- Executive Power: this government entity elaborates the Provincial Budget Draft. It is constituted by a Rector Organism (governing body) and several Executor Organisms. The first one define all activities for formulating a budget and the others execute these activities.
- Legislative Power: this government entity passes the Annual Budget Law.

Along with the budget life cycle the evaluation and control of actual and financial resources is made, and all of them are assigned to goods and services production. Table 1 shows the steps in detail.

1. Initiate Fiscal Year and Distribute Classifiers
2. Prepare Preliminary Budget and Resources Estimation
3. Define Budgetary Policy and Expenses Projection
4. Determine Expenses Top
5. Formulate Budget Project Draft
6. Present Budget Project Draft to Legislature
7. Approve Budget in Legislature
8. Elaborate new budget according to Budget Law
9. Distribute Budget for executing
10. Elaborate Budgetary Modifications
11. Program Budget executing
12. Reconduct Budget
13. Closure Fiscal Year

**Table 1: Budget Life Cycle Steps**

There is common information for all budget life cycle stages: Expense and Resource Classifiers. The classifiers used in this work are: Institutional, Expense Object,

Geographic Locate, Finality Function, Resource Item, Financing Source, and Programmatic Categories.

There are two situations where the availability of semantic information associated to budgetary data is critical: budget formulation and approval tasks. In the first case, only government staff with specific knowledge can be involved, concentrating a great responsibility on a few people. In the second case, semantics information is necessary for analyzing budgetary data and then having the budget law passed. Here, this is more complex because all legislators must vote and most of them have no specific knowledge. For simplicity purposes, only the Formulation stage for expenses budget was considered for this study case.

### 3.3 Specification: Motivating Scenarios and Competence Questions

We included this step taking into account the opinion of Gruninger and Fox (1995). The authors consider that for modeling ontologies, it is necessary to count on informal logic knowledge model in addition to requirements resulting from different scenarios. The motivation scenarios show problems that arise when people need information that the system does not provide. Besides, the scenario description contains a set of solutions to these problems that includes the semantic aspects to solve them. In order to define motivation scenarios and communicate them to the involved people, templates have been used. These templates were based on those proposed to specify case uses in object oriented methodology (Uschold & Gruninger, 1996). An example is shown in Table 2. The template describes: the name of the scenario, people who participate in the scenario, a brief scenario description, and a list of possible terms related to the scenario. Since this template shows the most important information in a concise way, it is useful when the experts do not have a lot of time to analyze the scenarios.

<b>Scenario:</b> Local Budget Formulation.
<b>Actors:</b> Participants of the budget formulation for next year.
<p><b>Description:</b> The scenario proposed here is a person who must participate in the budget formulation task for the next year. This task is carried out along the previous year because it is necessary to have the budget approved before the next year begins.</p> <p>Executor organisms of each government jurisdiction make their own formulation task. The Rector Organism defines policies conducting the budget draft elaboration as well as the main expenses and resources classifiers for the year. Then, each organism elaborates its jurisdictional budget draft.</p>
<p><b>Terms:</b> budgetary classifier, expense and resource classifier, Institutional, Programmatic Category, Geographic, Expenses Object, Financing Source and Finality Function Classifiers, among others, for working on the budget draft.</p>

**Table 2: Scenario Description**

Competency questions proceed from motivation scenarios. This allows deciding the ontology scope to verify if it contains enough information to answer these

questions and to specify the detail level required for the responses. Besides, it defines expressivity requirements for the ontology because it must be able to give answers using its own terms, axioms and definitions. The scope must define all the knowledge that should be in the ontology as well as those that should not. It means that a concept must not be included if there is not a competency question that uses it. This rule is also used to determine whether an axiom must be included in the ontology or not.

Moreover, competency questions allow defining a hierarchy so that an answer to a question may also reply to others with a more general scope by means of composition and decomposition processes. As an example, some of them are shown in Table 3.

<b>Simple Questions</b>
Which are the budget states?
Which are the budgetary classifiers?
Which are the expenses classifiers?
Which are the resources classifiers?
Which are the executor organisms for Health Minister?
Which are the Health Minister Programs?
<b>Complex Questions</b>
Which is the institutional code for the Education Minister?
Which are the sector and subsector for Central Administration?
Which is the character code for "Decentralized Organism"?
Which properties have an Institution?
Which is the institutional code for "Pharmacological Producer Laboratory" SAF?
Which Institutions have Program Code = 16?

**Table 3: Competency Questions**

### 3.4 Specification: Ontology Granularity and Type

According to the level of conceptualization and granularity (Gómez-Pérez et al, 2004), the ontology proposed here is domain ontology. Domain ontology describes the vocabulary related to a specific domain. In this case study, the ontology describes the budgetary domain of Santa Fe Province. And, the ontology objective is to facilitate communication among the members of the central administration staff that must deal with the local budget, bringing adequate terminology to non-expert users.

The term ontology can be used to describe models with different degrees of structure. Particularly, the ontology defined in this paper is a formal structure expressed in artificial formally defined languages.



same ontology. Therefore, the work is concentrated on Domain Ontology development. This Ontology of general concepts will be able to be used in all budget states facilitating term reusability. Then, we can see that it ontology will be able usability too.

### 3.6 Conceptualization: Identification of Classes, Relations and Attributes

At this step, we considered 101 METHOD guide and recommendations. Besides, we used representations proposed by METHONTOLOGY to knowledge organization as concepts classifier trees (Fig. 2) to analyze hierarchies and attributes, binary relations, axioms and instances tables. For determining classes, we identified those terms of independent existence from the key terms list and the glossary.

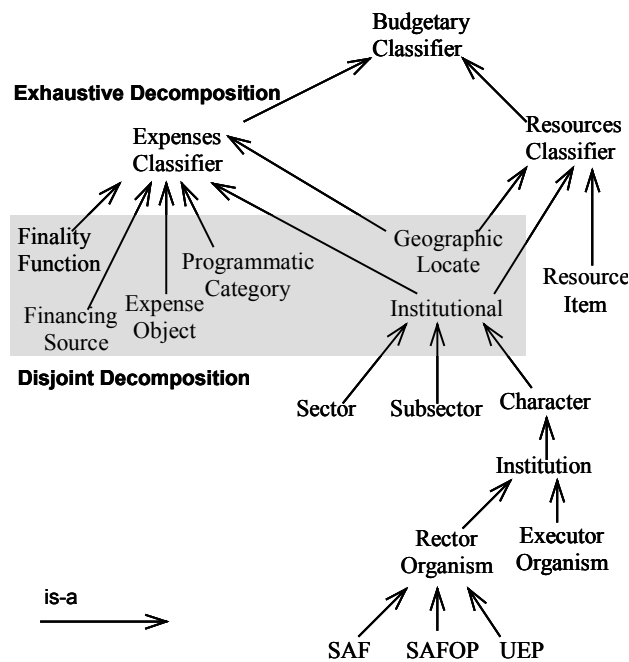


Fig. 2. Concepts Classifier Tree

Disjoint classes, exhaustive decompositions and partitions (Horridge et al, 2001) may be identified in these graphic representations:

- A Disjoint-Decomposition of a concept C is a set of subclasses of C that do not have common instances and do not cover C, that is, there can be instances of the concept C that are not instances of any of the concepts in the decomposition. As an example (see Fig. 2), Finality Function, Financing Source, Expense Object, Programmatic Category, Geographic Locate and Institutional can be mentioned as disjoint.
- An Exhaustive-Decomposition of a concept C is a set of subclasses of C that cover C and may have common instances and subclasses, that is, there cannot be instances of the concept C that are not instances of at least one of the concepts in the decomposition. For example (see Fig. 2), the concepts Expenses Classifier and Resource Classifier make up an exhaustive decomposition of the concept Budgetary Classifier because there are no classifiers

that are not instances of at least one of those concepts, and those concepts can have common instances.

- A Partition of a concept C is a set of subclasses of C that do not share common instances and that cover C, that is, there are not instances of C that are not instances of one of the concepts in the partition. In this scenario there are no partitions.

It is always convenient to begin with primitive classes, analyzing which of them are disjoint and verifying if that condition does not produce instances absents.

Once the hierarchies and their features have been identified a table to reflect bidirectional relations may be elaborated by means of assigning names using uniform criteria (or a uniform criterion), identifying domain and range, cardinality and inverse relations. An example is shown in Table 5. Shaded rows are bidirectional relations between concepts shown in the Concepts Classifier Tree. The relation direction depends on competence questions to be solved and the possible conflicts with other defined classes restrictions. A restriction list identifies those necessary and sufficient conditions and those only necessary to work later on their formalization. We analyzed the axioms both individually and in a group of classes to verify if closure restrictions are required.

Concept	Relation	Cardinality	Concept	Inverse Relation
Institutional	inst-include-sec	1	Sector	sec-isPartOf-Inst
Institutional	inst-include-sbsec	1	Subsector	sbsec-isPartOf-Inst
Institutional	inst-include-char	1	Character	char-isPartOf-Inst
Sector	sec-isPartOf-Inst	1,n	Institutional	inst-include-sec
Subsector	sbsec-isPartOf-Inst	1,n	Institutional	inst-include-sbsec
Character	char-isPartOf-Inst	1,n	Institutional	inst-include-char
Character	char-has-Inst	1,n	Institution	inst-correspond-char
Institution	ins-has-SAF	1	SAF	SAF-correspond-inst

Table 5. Bidirectional Relations

### 3.7 Conceptualization: Instance Definition

Once the conceptual model of the ontology has been created, the next step is to define relevant instances inside an instance table.

According to METHONTOLOGY, each instance should be provided a definition of: its name, the name of the concept it belongs to, and its attribute values if known.

An excerpt of the Instance Table of the Budgetary Ontology is shown in Table 6.

Concept Name	Instance Name	Property	Value
Institutional	Institutional_111	cod-institutional	1.1.1
		has-fiscal-year	2004
		inst-include-sec	1-No Financial Local Public Sector
		inst-include-sbsec	1- Local Administration
		inst-include-char	1- Main Administration
	Institutional_212	cod-institutional	2.1.2
		has-fiscal-year	2004
		inst-include-sec	2-Financial Local Public Sector
		inst-include-sbsec	1-Official Banking System
		inst-include-char	2- Official Banks

Table 6. An excerpt of the Instance Table of the Budgetary Ontology.

#### 4 Implementing the Budget Ontology with PROTÉGÉ 3.1

In order to implement the ontology, we chose Protégé 3.1 because of the fact that it is extensible and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development (Knublauch et al, 2005). Protégé ontologies can be exported into different formats including RDF Schema (RDFS) (Brickley & Guha, 2004), and Web Ontology Language (OWL) (Smith et al, 2004). Particularly, we have implemented the Budgetary Ontology in OWL and verified its consistency by using Racer (Haarslev & Möller, 2001). It was very useful for determining unsatisfiability problems and their propagation causes. An OWL class is deemed to be unsatisfiable (inconsistent) if, because of its description, it cannot possibly have any instances (Wang, 2005).

During the verification process, we have taken into account experience of CO-ODE Project (Knublauch et al, 2005), and practical experience of teaching OWL-DL reported by (Rector et al, 2004).

To compare the ontology implementation with its conceptualization, graphics using the OWLViz and Ontoviz plug-ins were generated and compared with UML diagrams. On the one hand, OWLViz enables the class hierarchies in OWL Ontology to be viewed, allowing comparison of the asserted class hierarchy and the inferred class hierarchy. OWLViz integrates with the Protege-OWL plugin, using the same color scheme so that primitive and defined classes can be distinguished, computed changes to the class hierarchy may be clearly

seen, and inconsistent concepts are highlighted in red. Fig. 3 shows the Domain Ontology taxonomy.



Fig. 3. Domain Ontology Taxonomy.

On the other hand, OntoViz generates diverse combinations of graphics with all relations defined in the ontology, instances and attributes. OntoViz allows visualizing several disconnected graphs at once. These graphs are suitable for presentation purposes, as they tend to be of good clarity with no overlapping nodes. An example of them in Fig. 4 shows The main relations of the concept Institutional with other concepts, and an instance of this concept, Local Administration.

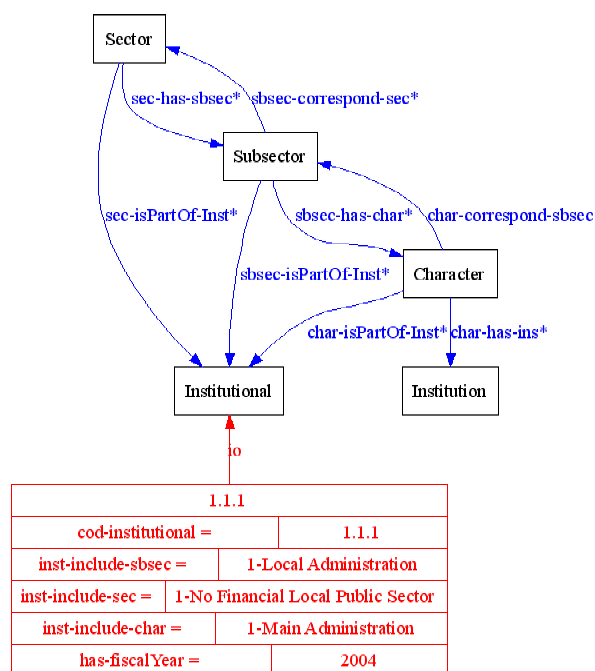


Fig. 4. Main Relations Between Concepts of Institutional Classifier



## 4.1 Ontology Querying

In order to verify and validate the ontology as regards competency questions, we used the RDF Data Query Language (RDQL) (Seaborne, 2004). RDQL is an implementation of an SQL-like query language for RDF. It treats RDF as data and provides queries with triple patterns and constraints over a single RDF model. Another query language is OWL-QL (Fikes et al, 2003), which was designed for query-answering dialogues among agents using knowledge in OWL. Then, OWL-QL is suitable when it is necessary to carry out an inference in the query. This is not the case of the major competency questions; then, RDQL is enough. Hence, RDF ontology was created from Protégé Project. Following the RDQL query that models the competency question “Which are the sector and subsector for Main Administration?” is shown.

```
SELECT ?x ?y ?z ?nsec ?nsbsec
WHERE (x,<adm:rdfssec-hasbsec>,?y)
      (?y,<adm:rdfssec-has-char>,?z)
      (?z,<rdfs:label>,'1-Main Administration')
      (?x,<rdfs:label>, ?nsec),
      (?y,<rdfs:label>, ?nsbsec)
USING rdfs FOR
      http://www.w3.org/2000/01/rdf-schema#
adm FOR http://protege.stanford.edu/
```

To implement the queries, Jena framework has been used. Jena is a Java toolkit which provides an API for creating and manipulating RDF models. Jena sources can be retrieved at <http://jena.sourceforge.net/>.

## 5 Discussion

In order to develop the ontology presented in this paper, the methodology outlined in Fig. 8 has been followed. This methodology was divided into three phases: Specification, Conceptualization and Implementation according to the METHONTOLOGY Framework. These phases constitute an iterative process. This framework provided the idea of support activities: Knowledge Acquisition and Validation/Verification.

The innovation of the methodology presented in this paper consists of the tasks that compose each phase. These tasks were imported from 101 Method and Grüniger & Fox Methodology. In addition, they were enriched with some Software Engineering techniques.

The most important task in the methodology is the definition of a Domain Conceptual Model. Then, it is important to assign all the necessary time to carry out a good conceptual analysis. The conceptual model resumes the knowledge acquired during the specification phase and it is the basis of conceptualization. This conceptualization has to be agreed on by domain experts. Then, the use of a graphical representation is essential in order to facilitate communication between ontology engineers and experts. So, software engineering techniques that could be familiar for the domain experts, such as UML, can be useful. Although UML in its

standard form is not suitable for ontology representation, we cannot ignore that UML is a standard and its use is widely spread among different communities.

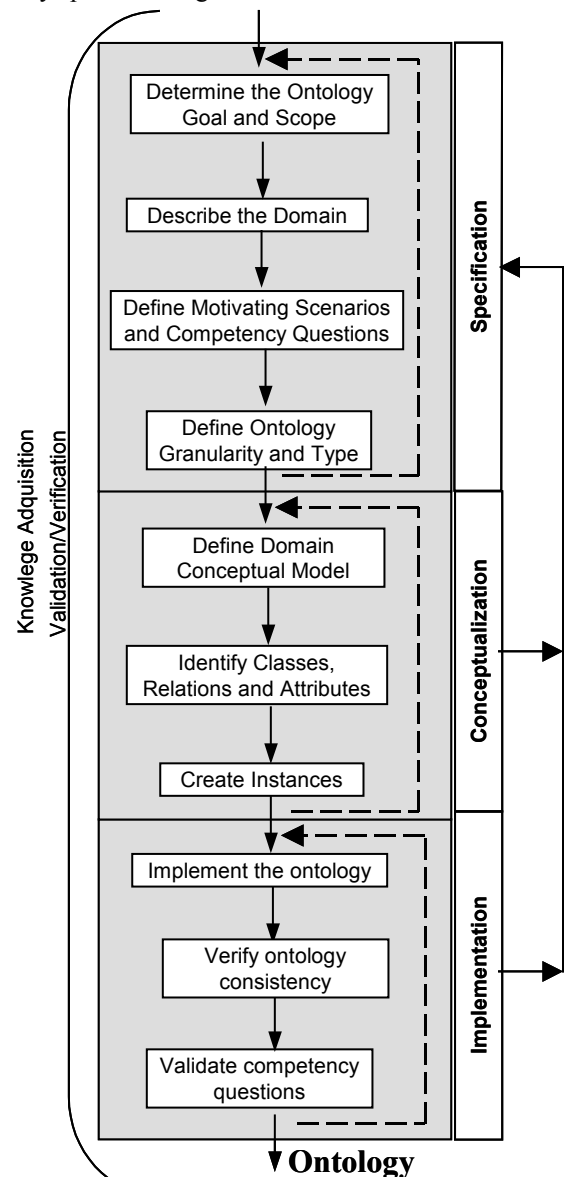


Fig. 8 . A Domain Ontology Development Process.

Another important aspect to consider in developing a good ontology is to carry out a permanent and iterative validation process, taking into account that partial verifications allow identifying errors propagation between sets of classes.

Furthermore, for the purpose of making the ontology more flexible and allowing extensibility and reuse, it is important to modularize the ontology if possible. This modularization can be made through relations and attributes observation of conceptual aspects involved.

## 6 Conclusions

Building domain ontologies is not a simple task when domain experts have no background knowledge on engineering techniques and/or they have not much time to invest in domain conceptualization.

In this paper, we have shown how ontologists could develop domain ontologies merging different methodologies and software engineering techniques, taking advantages of them. Particularly, this approach has been used to define a Domain Ontology for a Budgetary and Financial System, which could be extended by Task Ontologies and used by different government applications.

Sharing the best practice on ontology building can be useful for the whole community. Then, the contribution of this paper is the implementation and improvement of a systematic process for the development of domain ontologies.

## 7 References

- Brickley, D., Guha, R.V. (2004) RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. <http://www.w3.org/TR/rdf-schema/>
- Caliusco M. L. (2005) A Semantic Definition Support of Electronic Business Documents in e-Colaboration. PhD thesis. UTN - F.R.S.F. Santa Fe, Argentina.
- Corcho O, Fernández-López M, Gómez-Pérez A, López-Cima A. (2005) Building legal ontologies with METHONTOLOGY and WebODE. Law and the Semantic Web. Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications.
- Cranefield, S. (2001) UML and the Semantic Web. Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA:113-130.
- Cranefield, S., Purvis, M. (1999) UML as an Ontology Modelling Language. Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration, Held in conjunction with the Sixteenth International Joint Conference on Artificial Intelligence City Conference Center, Stockholm, Sweden.
- Falbo, R.A. (2004). Experiences in Using a Method for Building Domain Ontologies. Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering, SEKE'2004, pp. 474-477, International Workshop on Ontology In Action, OIA'2004. Banff, Alberta, Canada.
- Fikes, R., Hayes, P., Horrocks, I. (2003) OWL-QL - A Language for Deductive Query Answering on the Semantic Web. KL Laboratory, Stanford University, Stanford, CA.
- Gómez-Pérez A., Fernández López M. and Corcho O. (2004) Ontological Engineering with examples from the areas of knowledge management, e-commerce and the semantic web. London: Springer.
- Gruninger M. and Fox M. S. (1995) Methodology for the Design and Evaluation of Ontologies, IJCAI Workshop on Basic Ontological in Knowledge Sharing, Montreal, Canada.
- Haarslev V. and Möller R. (2001). RACER System Description. In Proceedings of the First international Joint Conference on Automated Reasoning. IJCAR.
- Horridge M., Knublauch H., Rector A., Stevens R., Wroe C. (2004) A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0, The University Of Manchester Stanford University.
- Jarrar M. (2005) Towards Methodological Principles for Ontology Engineering. PhD Thesis, Vrije Universiteit Brusell.
- Jones D., Bench-Capon T. y Visser P. (1998) Methodologies for Ontology Development, en Proc. IT&KNOWS Conference, XV IFIP World Computer Congress, Budapest.
- Klischewski R., Lenk K., (2002), Understanding and Modelling Flexibility in Administrative Processes. In: Proceedings of EGOV 2002, Ed. Traummüller R., and Lenk K., pp.129-136.
- Knublauch H., Horridge M., Musen M., Rector A., Stevens R., Drummond N., Lord P., Noy N., Seidenberg J., Wang H. (2005) The Protégé OWL Experience, Workshop on OWL: Experiences and Directions, Fourth International Semantic Web Conference (ISWC2005), Galway, Ireland.
- Noy N., McGuinness D., (2001) Ontology Development 101: A Guide to Creating Your First Ontology.
- Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C (2004) OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns (2004) in E Motta and N Shadbolt, et al. (eds) Proceedings of the European Conference on Knowledge Acquisition, Northampton, England, LNAI3257, pp 63-81
- Seaborne A., (2004) RDQL - A Query Language for RDF, W3C Member Submission <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>
- Smith, B. "Ontology", (2003), in L. Floridi (ed.), *Blackwell Guide to the Philosophy of Computing and Information*, Oxford: Blackwell, 155-166.
- Smith, M., Welty C., McGuinness D. (2004) OWL Web Ontology Language Guide, W3C Recommendation 10 <http://www.w3.org/TR/owl-guide/>
- UML (2006) Unified Modeling Language. <http://www.uml.org/>
- Uschold, M., Building Ontologies: Towards a Unified Methodology, (1996) 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems, Cambridge, UK.
- Uschold, M., Gruninger M. (1996) Ontologies: Principles, Methods and Applications. Knowledge Engineering Review.
- Wache H., Vögele T., Visser U., Stuckenschmidt H., Schuster G., Neumann H., Hübner S. (2001) Ontology-Based Integration of Information –A Survey of Existing Approaches. Proc. IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA, 108-117.



Wang H., Horridge M., Rector A., Drummond N. and Seidenberg J., (2005), Debugging OWL Ontologies: A Heuristic Approach. 4th International Semantic Web Conference (ISWC'05), Galway, Ireland.



# Trust Based Ontology Integration For The Community Services Sector

Dennis Hooijmaijers

Markus Stumtner

Advanced Computing Research Centre  
University of South Australia,  
Mawson Lakes Blvd, Mawson Lakes, South Australia 5095,  
Email: dennis@cs.unisa.edu.au

## Abstract

As ontologies become more prevalent for information management the need to manage the ontologies increases. In the community services sector multiple organisations often combine to tender for funding. When separate organisations come together to generate reports for funding bodies an alignment of terminology and semantics is required. Ontology creation is privatised for these individual organisations to represent their view of the domain. This creates problems with alignment and integration, making it necessary to consider how much each ontology should influence the current decision to be made.

To assist with determining influence a trust based approach on author and the ontologies provides a mechanism for ranking reasoning results. A representation of authors and the individual resources they provide for the merged ontology becomes necessary. The authors are then weighted by trust and trust for the resources the author provides to the ontology is calculated. This is then used to assist the integration process allowing for an evolutionary trust model to calculate the level of belief in the resources. Once the integration is complete the semantic agreement between the ontologies allows for the recalculation of the author's trust.

**Keywords:** Ontology, Trust, Belief Revision, Ontology Integration

## 1 Introduction

Disparate information sources exist in most domains (Noy, Mitra & Jaiswal 2004). Terminology and semantics vary between individuals within the domain. This creates difficulty when attempting to integrate work from multiple sources. It becomes necessary to create an appropriate mapping so that integration can proceed.

In certain domains, such as the *community services sector*, it is imperative that communication and understanding can occur. The community services sector is a conglomeration of distinct projects. The disparate projects each tender for funding from external bodies based on the services they can provide. The project groups often band together to tender for funding. Each funding body requires reports to ensure the projects are providing the support stipulated in the financial contracts. Each project sets up its own data collection including

selecting what, how and when to collect. This makes it difficult to integrate the data for reporting to the funding bodies. Terminology and attributes are often used with different semantic descriptions and the data is collected for different purposes. This merging may involve the alignment and merging of the *private ontologies* to allow for reporting and reasoning to assist in achieving mutual, and/or individual, goals.

Integrating ontologies is not a trivial task, and can never be fully automated (Dou, McDermott & Qi 2003). Many researchers have provided tools for assisting in finding appropriate mappings and using the mapping to merge the ontologies, such as PROMPT (Noy & Musen 2000), Chimera (McGuinness, Fikes, Rice & Wilder 2000) and OntoMerge (Dou et al. 2003), to assist the user in merging these ontologies. These tools assume the ontologies are error free and the creator is not attempting to mislead or sabotage their results.

Trust is a major concern in most enterprise interactions and the Internet allows for utilising information from strangers (Resnick, Kuwabara, Zeckhauser & Friedman 2000). Social networking is an approach to apply trust propagation between entities. The *Friend of a Friend* (FOAF) (Golbeck, Parsia & Hendler 2003) and *Web of Trust* (Guha 2003) combine direct trusts of individuals to create a graph of nodes to capture indirect trust, or reputation. This allows an individual to use a stranger's reputation to base their belief on the knowledge that stranger provides. When deciding on whether to believe an individual reputation is often augmented by accordance between our beliefs and the individuals.

The main premise in this work is the following: If a trust value for an individual exists, then this should be taken that into account when integrating that individual's ontology. Furthermore if an individual's ontology is in accordance (that is provide no contradictions) with our ontology, then the trust in that individual should increase.

This paper describes a model to capture and update author trust for ontology integration. The preliminary results show an ability to differentiate between resources within an ontology, and authors, based on their believability and to show *mistrust* of an author.

## 2 Ontology Model

An *ontology* can be described as a collection of resources that explicitly and formally conceptualise a domain model (Guarino 1998). This work uses the OWL DL ontology language as defined by the W3C (McGuinness & van Harmelen 1994-2006)

**Definition 1** (Ontology). An OWL ontology,  $\Omega$ , is considered to be a 4-tuple of resources  $\langle C, S, I, A \rangle$

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at The Australasian Ontology Workshop (AOW2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 72. M. A. Orgun and T. Meyer, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

where  $C$  is a set of *classes*,  $S$  is a set of *semantic relations*,  $I$  is a set of *instances*, and  $A$  is a set of *axioms*. In OWL DL  $C$  and  $I$  are disjoint (the same term can not denote both a class and an instance)

Any ontology that a user adopts for their use is considered to be a *private ontology*. Private ontologies are a specific version of an ontology that a particular user has selected. This is important as an user may use a publicly available ontology but will decide whether to evolve the ontology themselves or to adopt newer versions. This will create an environment of fragmented ontology versions that may no longer be compatible for merging. To assist in managing *private ontologies* it is necessary to apply mechanisms for author and version discovery. An author is defined as any entity that contributes to a resource to a private ontology.

**Definition 2** (Private Ontology). A *private ontology* is an extension of an ontology that relates an author, as a provider,  $\mathcal{P}$ , to the provided Resource  $\mathcal{R}$ .  $\mathcal{P} \mapsto \mathcal{R}$ , where  $\mathcal{R} \subset C, S, I, A \in \Omega$

In this work we identify authors by a unique identifier and an affiliation to an organisation and project that they work on. These were identified as the necessary factors for basing integration within the community services sector. Initially we only calculate trust based on the organisation and the individual themselves. Further experiments are necessary to discover if the project, an author is affiliated with, will influence trust in that individual. To capture multiple individual authors and map them to their provided resources it is necessary to provide additional classes and semantic relationships as shown in Figure 1. By creating the necessary classes as subclasses of ‘owl:thing’ it allows for all ontologies to be integrated to these classes by taking all subclasses of ‘owl:thing’ in the ontology and making them a subclass and instance of ‘trust-rated class’ and all other resources become instances of the appropriate trusted version. By making classes instances of ‘trust-rated class’ the ontology is no longer a valid ‘OWL DL’ ontology but an ‘OWL FULL’ ontology. Although the ontology is now ‘OWL FULL’ it is only used to generate reasoning rules based on the trust values and the authors. The reasoning will occur over the original resources, ignoring the instances of ‘trust-rated class’ and retaining computational completeness (McGuinness & van Harmelen 1994-2006).

### 3 Trust Model

Numerous trust models have been proposed in recent years to rate or quantify belief in information provided by a person. Jøsang (Jøsang, Ismail & Boyd 2005) provides the following definition:

Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends.

A key distinction is between *direct trust* (local trust) and *reputation* (global trust) (Ziegler & Lausen 2004). Reputation is an individual,  $\mathcal{P}$ , using another individual’s,  $\mathcal{P}'$  trust of an individual,  $\mathcal{P}''$ , to determine to determine their trust of  $\mathcal{P}''$ . Direct trust is based on previous experience with the source of the information, while reputation requires a social network model to provide a similar measure. Some trust researchers (Bertino, Ferrari & Squicciarini 2004) also aim to provide a metric for authenticity. This *security trust* aims to prevent

document tampering (encryption) and ascertain authorship (*digital signatures*). These metrics often ignore the source’s credibility and reliability (Golbeck et al. 2003). Credibility is the concept of trusting a source, in a domain, to provide information that is most likely to be correct (Ding, Zhou & Finin 2003, Golbeck et al. 2003), while reliability is the concept of ranking sources by their credibility ratings (Golbeck et al. 2003, Huang & Fox 2005).

This work focuses on the credibility of an ontology and an author, and trust is considered a value that represents the belief that the author produces credible work. Author trust is the subjective probability that an author will provide resources for an ontology,  $\Omega$  that will not conflict (or be incorrect for the purpose of  $\Omega$ ) with those already contained within  $\Omega$ .

**Definition 3** (Author Trust). Author trust,  $\tau : \mathcal{P} \times \mathcal{P} \rightarrow f(\mathcal{P})$ , is a partial function that corresponds to author ratings for other authors.

The *Web of Trust* (Guha 2003) uses trust levels and provides a technique to model trust and distrust between objects (ontologies or knowledge bases), users, reviews, ratings of reviews, and trust relations between users. The model allows for trust to be captured and rated between users. Each review can be ranked and allows for the propagation of trust changes. When a user’s trust rank is altered, their reviews of other users will gain additional value within the model to reflect their improved reputation. One problem is that the approach does not take into account the reliance between objects. If an ontology,  $\Omega$ , is built incorporating another ontology,  $\Omega'$ , then the rating of  $\Omega'$  can not be higher than  $\Omega$ . Belief of statements made by an author are calculated from trust and distrust propagation.

Friend of a Friend (FOAF) (Dumbill 2002) is an approach aiming to mark up trust relations between people in XML and RDF to enrich personal web pages. It provides a simple XML schema to allow an author to define people they know. This creates a directed graph with nodes representing people and edges representing direct trust. The FOAF approach has been extended to provide levels of trust, ranking from 1 **distrusts absolutely** to 9 **trusts absolutely** (Golbeck et al. 2003), and also allowing for the trust ratings to be related to specific topics. Trust can be calculated in three ways, via path capacity, path length and weighted averages. A similar approach is followed by Ziegler and Lausen (Ziegler & Lausen 2004), but this does not model distrust but rather *lack of trust*.

Marsh and Dibben (Marsh & Dibben 2005) break trust into *trust*, *un-trust*, *distrust* and *mistrust*. Where un-trust is considered to be how little the individual is trusted. Distrust is a measure of how much the individual is believed to actively work against the trustee. Mistrust is the level to which the trustee was wrong in their trust or distrust of an individual. In this work distrust and trust are actively modelled while un-trust is a direct calculation from the trust and distrust. Mistrust is modelled by using an evolutionary trust model where author and resource trust levels are considered to be dynamic during the ontology merging process.

Ontology trust is calculated from the belief ratings of each individual resource. Resource belief is the extent to which the user will depend on the resource in a given situation with a feeling of relative security, even though negative consequences are possible.

An Agent social model, introduced by Zheng et al. (Zheng, Chen, Wu & Zhang 2006) provides mechanisms for calculating trust for individual authors. The model uses a utility function to

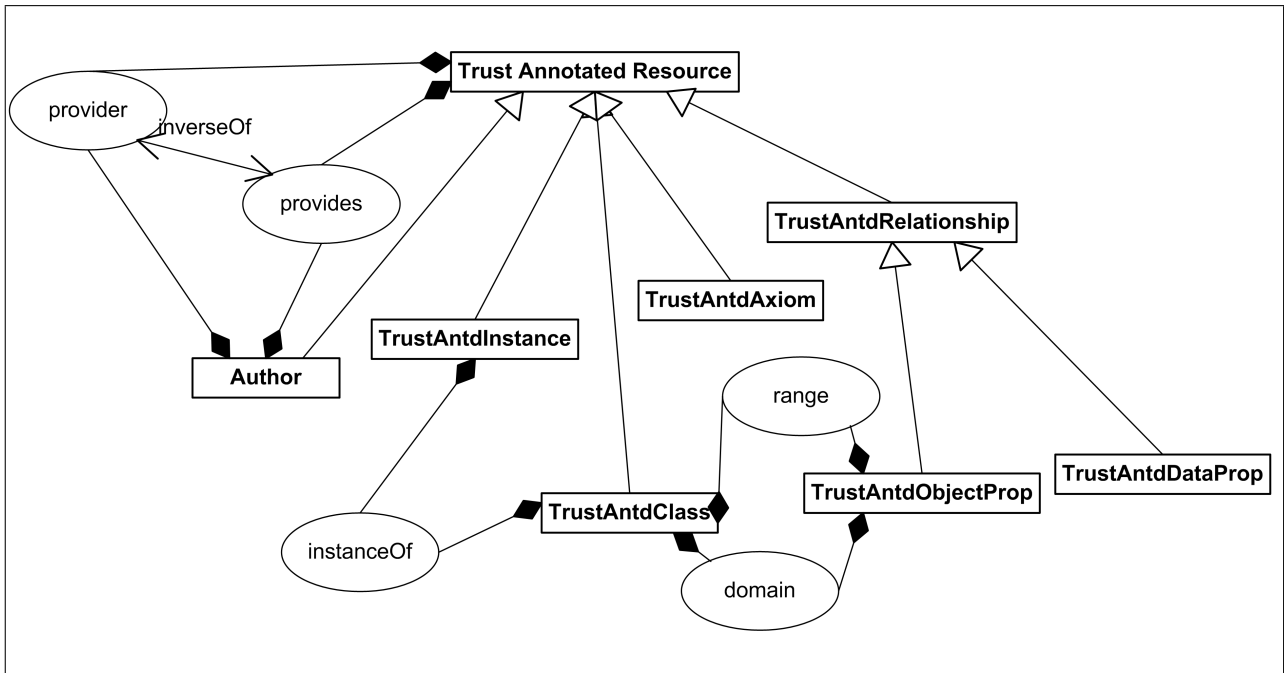


Figure 1: Semantic Structure for Trusted Ontologies

create a *preference* based system for selecting the appropriate author. This preference is a numerical value created from other agents beliefs, in regards to *operational cost*, *opportunity cost*, and *service charge*, of an author. The limitation of this approach is that the author preference is applied to all services that the author provides equally. This minimises the ability to evaluate what area is the authors expertise and also minimises the ability to calculate mistrust (Marsh & Dibben 2005) and how it relates to the areas of expertise.

By assigning a name to an RDF graph, which an OWL ontology is a subset of, Carroll et. al (Carroll, Bizer, Hayes & Stickler 2005) introduce the ability to link an author to an ontology as well as to express meta-information about the graph. This is achieved by extending the current Semantic Web approach with an additional node, reference, or ID. This allows for different authors to supply a resource with different descriptions and to select the appropriate resource for a given situation. These descriptions could be used to link resource to authors contained within an existing social network, such as the Web of Trust (Guha 2003) or FOAF (Dumbill 2002). In our work we decided to capture this meta-information within the current OWL standard by expressing it via a set of predefined top level concepts and roles.

**Definition 4** (Resource Belief). Resource belief,  $\rho: \mathcal{P} \times \mathcal{R} \rightarrow f(\mathcal{R})$  is a partial function that corresponds to resource ratings calculated from the providing authors.

In this paper trust levels are used for author trust and resource belief is boolean based. Trust levels for authors correspond to the 9 levels used by Golbeck, Parsia and Hendler's FOAF extension (Golbeck et al. 2003):

1. Distrusts absolutely
2. Distrusts highly
3. Distrusts moderately
4. Distrusts slightly
5. Trusts neutrally (un-trust)

6. Trusts slightly
7. Trusts moderately
8. Trusts highly
9. Trusts absolutely

This allows ranking authors by reliability and filtering based on the various levels of trust. Resources are used in a binary fashion, they are either used for the current situation or not. Therefore the state of the resource is represented by a boolean value, True meaning the resource is *believed* to be accurate and will be used and False meaning that the believability of the resource is in *doubt*. In this work trust can be separated into initial author trust assignment, resource belief calculation and the updating of author trust. Author trust assignment is performed prior to ontology integration while resource belief and the author trust update are performed after the integration is complete.

### 3.1 Author Trust Assignment

Calculating initial trust can be based on multiple factors. In this work initial trust can be selected from any previous metric of *direct trust*, *connected trust*, *organisational trust* and un-trust. Where the value of direct trust subsumes connected trust which subsumes organisational trust. Direct trust is captured within the user's private ontology if they have integrated an ontology previously from an individual. Connected trust is captured within the user's private ontology when they have integrated an ontology that contains direct trust to the individual. While organisational trust is captured by predefining a trust value for an organisation and propagating it to each individual within the organisation. Finally un-trust is the default trust value that is given to any individual that does not belong to an organisation and does not exist in the private ontology.

### 3.2 Resource Belief Calculation

We calculate resource belief by combining the trust of each provider of that resource, as shown in Figure 2.

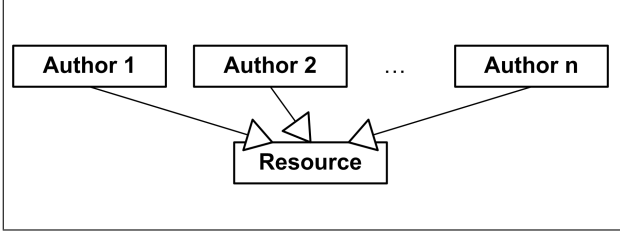


Figure 2: Resource Belief Calculation

A belief threshold,  $n$ , applied to the combined trust determines if the resource is believable. Guha (Guha, Kumar, Raghavan & Tomkins 2004) shows that it is not possible to simply subtract distrust from trust for transitive trust (Jøsang & Pope 2005) and uses two separate ranks for trust and distrust for an entity based on the average value between the agents. Since belief for the resource is calculated from direct links to the providers this is not an issue here and the resource belief value can be calculated as follows:

$$\rho(\mathcal{R}) = \sum_{\mathcal{P} \mapsto \mathcal{R}} \frac{\tau(\mathcal{P})}{N_{\mathcal{P}}} \quad (1)$$

If  $\rho(\mathcal{R})$  is greater than  $n$  the resource is *believed* else it remains *doubtful*.

### 3.3 Author Trust Updating

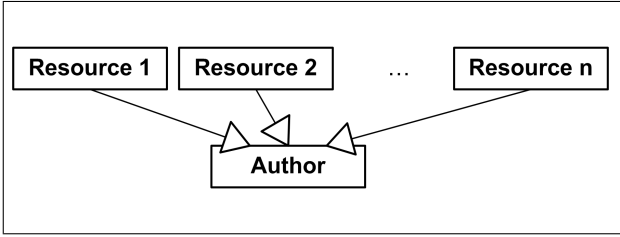


Figure 3: Author Trust Evolution

Once the integration process is complete resources that have been merged have multiple authors and the believability of the resource may have changed. This will effect the trust rating of any author that has provided that resource. The initial author trust assignment is then updated by taking into account each  $\rho(\mathcal{R})$  that the author provides, as shown in Figure 3 and calculated using:

$$\tau(\mathcal{P}) = \tau(\mathcal{P}) + \frac{\sum_{\mathcal{P} \mapsto \mathcal{R}} \rho(\mathcal{R})}{N_{\mathcal{R}}} - \left( \frac{\sum_{\mathcal{P} \mapsto \mathcal{R}} \rho(\mathcal{R})}{N_{\mathcal{R}}} \times \tau(\mathcal{P}) \right) \quad (2)$$

## 4 Private Ontology Integration

Ontology merging is the process of creating a mapping between two ontologies and then combining them to create a new ontology. Automated knowledge integration has been an active research area for some time (McGuinness et al. 2000, Noy & Musen 2000, Dou et al. 2003), but so far has mostly concentrated on knowledge assumed to be stable, certain, and (for a particular problem or domain) complete. Automated advisors have been suggested to overcome the ‘knowledge acquisition bottleneck’ (Gonzalez & Dankel 1993) caused by reliance on human experts. Such systems, including PROMPT (Noy & Musen 2000), OMEN (Noy et al. 2004) and Chimaera (McGuinness et al. 2000), use simple heuristics to assist users in making the best decisions.

In real world situations, knowledge is subjective and the creators of the knowledge are not always completely certain of the correct semantics between the classes.

The Chimaera ontology integration tool (McGuinness et al. 2000) uses a suggestion algorithm based on pattern matching of class labels. It searches through the first ontology and then, using a set of reasoning rules, searches through the second ontology for a match. Chimaera mainly focuses on the subsumption hierarchies within ontologies. What Chimaera attempts to do is to integrate two hierarchies contained within multiple ontologies. This is achieved by determining if two similar classes are the same or one is a subclass of the other.

OntoMerge, proposed by Dou (Dou et al. 2003), was originally designed to convert entity relationship models from database schema to an ontology, in a process called ‘Ontolization’. Dou states that this process can not be fully automated since only domain experts know the meanings and relationships of the terms. Manual mapping is adopted by OntoMerge to provide a mechanism for merging ontologies, which was proven to be a requirement for the translation of schema to ontologies (Noy & Musen 2000). OntoMerge uses first-order logic to provide predicate axioms to map the source ontologies onto the target ontology. Predicate axioms provide rules for locating equivalent classes and for locating possible conflicts that require resolution.

PROMPT (Noy & Musen 2000) was developed to semi-automatically merge ontologies. The merge occurs automatically where attainable, and by guiding the user where it is unable to decide upon the correct results. Automated identification of likely class mappings will reduce a domain expert’s effort in searching through the ontologies. OMEN (Noy et al. 2004) is an extension of PROMPT that uses Bayesian networks to assign belief values to likely matches, thereby providing a finer graduation of candidate matchings. When a user selects a match, the beliefs are updated by propagating the new evidence through the network. The major evidence that increases probabilities for a match, is caused by ‘down flow’, starting at the point where a class’s ancestor in a hierarchy was matched.

All these systems assume that the ontologies to be matched are accurate and contain only definite classes and semantics. Yet, since the reason to perform an ontology matching process is presumably the need to incorporate from all sources involved, any long term usage scenario will find agents possessing and working with ontologies that have already undergone this matching process.

Ontology merging is the process of combing an ontology,  $\Omega$  with another ontology  $\Omega'$  to create a merged ontology  $\Omega_f$ . In this work we are combing *private ontologies* and thus have sets of providers,  $P_1$  and  $P_2$ , contained in each ontology with their mappings onto the resources that each individual has provided for the ontology. Ontology merging can be divided into the following operations derived from PROMPT (Noy & Musen 2000):

- perform a shallow copy of a resource
- merge resource
- perform a deep copy of a resource

**Definition 5** (Shallow Copy Resource). Let  $\mathcal{R}$  be a resource in either  $\Omega$  or  $\Omega'$ , create a resource  $\mathcal{R}'_f \in \Omega_f$  and add as instance of the appropriate *trusted resource*. Add all providers  $\mathcal{P}(\mathcal{R})$  as providers  $\mathcal{P}(\mathcal{R}'_f) \in \Omega_f$ . If  $\mathcal{R}$  is a subclass of ‘owl:thing’, add

the subclass of *trust-rated class* relationship to the set of semantic relations,  $S$ .

**Definition 6** (Merge Resource). Let  $\mathcal{R} \in \Omega$  and  $\mathcal{R}' \in \Omega'$  be two resources that are found to be equivalent, and let  $\mathcal{R}$  (w.l.o.g.) be the resource with the higher belief value according to eq 1. Shallow copy  $\mathcal{R}$  into  $\Omega_f$  and add as instance of the appropriate *trust-rated resource*. Add all providers  $\mathcal{P}(\mathcal{R}) \in \Omega$  and all providers  $\mathcal{P}(\mathcal{R}') \in \Omega'$  as providers  $\mathcal{P}(\mathcal{R}_f) \in \Omega_f$ . Such that  $\mathcal{P}(\mathcal{R}_f)$  is equivalent to  $\mathcal{P}(\mathcal{R}) \cup \mathcal{P}'(\mathcal{R}')$ . We refer to this as merging  $\mathcal{R}'$  into  $\mathcal{R}$  in  $\Omega_f$  ( $\Omega_f$  is omitted if it is understood).

Unlike PROMPT, which expects a user to select a preferred ontology among those being merged, and selects the label of a merged resource from that ontology, we use the label of the resource which has the higher belief value. A choice is only necessary if the belief values are the same

A deep copy is the process of performing a shallow copy for all parents of a resource (Noy & Musen 2003). Where a parent is a resource  $\mathcal{R}$  that is linked to a resource  $\mathcal{R}'$  by a semantic relationship,  $S$ , where  $S$  is ‘`rdfs:subClassOf`’ or ‘`rdfs:subPropertyOf`’ as defined by the W3C (McGuinness & van Harmelen 1994-2006). It is also necessary to perform the operations in order of type of resource:

1. classes
2. semantic relationships
3. axioms
4. instances

## 4.1 Conflict Resolution

Conflicts are any semantic inconsistencies in the merged ontology,  $\Omega_f$ . Noy and Musen (Noy & Musen 2000) indicate potential problems need to be addressed by the user and list the following:

- name conflict
- dangling references
- hierarchy redundancy
- property value restriction violations

These conflicts need to be resolved to ensure a consistent ontology. Name conflicts occur when two resources have the same name. Resolution involves renaming the resource, with a lower trust threshold. Currently in this work renaming is considered a ‘minor edit’ and will not involve altering the belief of the resource and the trust in providers. Dangling references occur during the merge process and are addressed by copying the appropriate resource into  $\Omega_f$ . Hierarchy redundancy occurs when multiple paths occur between a class and a superclass, deleting one of the ‘`rdfs:subClassOf`’ from the class. Property value restriction violations occur when a copied instance violates the defined restriction. This can be resolved by deleting the restriction or by editing the instance. Additionally it was found necessary to be able to perform the following operations:

- delete resource
- edit resource
- delete author

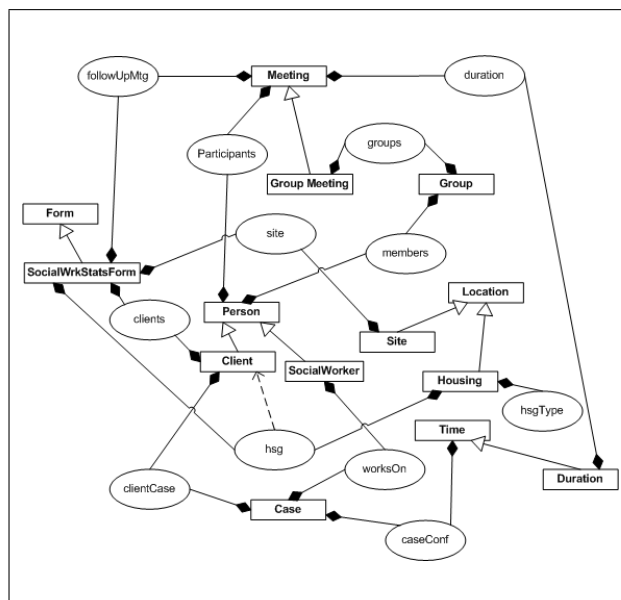


Figure 4: A Simple Social Work Counselling Ontology

Deleting a resource is the user stating that they do not believe the resource and this reflects a lack of trust in the authors that have provided that resource. If a resource is ‘doubtful’ any resource that directly relates to it will also be in doubt, (i.e if a class is in doubt then and object property related to, or instance of, that class will be ‘doubtful’)

**Definition 7** (Resource Deletion). To delete a resource,  $\mathcal{R}$ , set  $\rho(\mathcal{R})$  to ‘doubtful’. For each  $\mathcal{R}$ , such that  $\mathcal{R} \mapsto \mathcal{R}_1$  and  $\mathcal{R}_1 \notin C_f$ , set  $\rho(S)$  to ‘doubtful’.

Editing a resource is the process of deleting a resource and creating a new resource to replace it. To create a new resource add the resource as an instance of the appropriate ‘trust-rated resource’ and add the user as a providing author of that resource. If the resource is a subclass of ‘owl:thing’ it is necessary to add the subclass of *‘trust-rated class’* relationship to the set of semantic relations, *S*. and is performed similar to a *shallow copy*.

**Definition 8** (Delete Author). To delete an author,  $\mathcal{P}'$  and all mappings  $\mathcal{P}' \mapsto \mathcal{R}$ , for each  $\mathcal{R} \in \Omega_f$  if  $\mathcal{P}(\mathcal{R}) = \{\emptyset\}$  remove  $\mathcal{R}$  and all mappings  $\mathcal{R}' \mapsto \mathcal{R}$  from  $\Omega_f$ .

Author deletion is the actual removal of an author and all of their resources. This provides a mechanism for ontology evolution. Klein and Noy (Klein & Noy 2003) define a set of tasks that ontology evolution process should perform such as data transformation, data access, ontology update, consistent reasoning and Verification and Approval. Currently our approach provides support for all these tasks except verification and approval, which currently is handled manually.

## 5 Ontology Filtering

To provide benefit to the user from private ontologies it is necessary to provide different techniques to reason based on this additional information. A filter,  $\mathcal{F}$ , is a mechanism in which to select a partial set of resources from an ontology for which to base a decision on. By using filters we can compare results based on different selection criteria allowing for a more informed decision. Initially three filters have been implemented: *by author*, *by affiliation* and *by trust*.

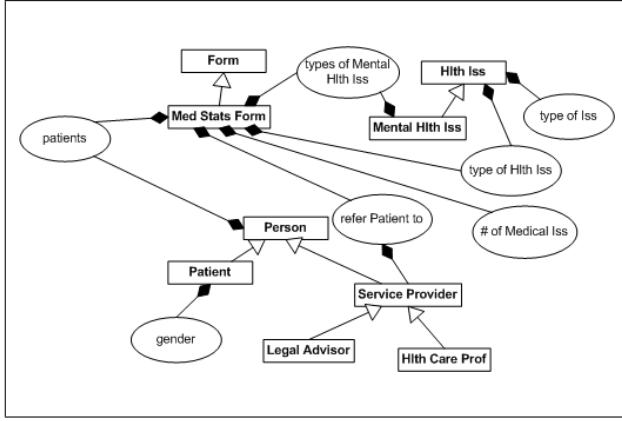


Figure 5: A Simple Medical Counselling Ontology

One problem when filtering is that a superclass may become ‘doubtful’, causing all subclasses to no longer have inherited properties that may be necessary. For example if ‘Service Provider’ in figure 5 were to become ‘doubtful’ then ‘Legal Advisor’ and ‘Health Care Professional’ will no longer have access to properties that differentiate ‘Service Provider’ from ‘Person’. This will occur in situations where multiple authors agree on the ‘subclass’ but do not agree on the ‘superclass’ and an instance of the ‘subclass’ instantiates the property. There are two possible solutions to this problem, create redundancy within the ontology, or migrate the properties to subclasses as part of the filtering process. To keep our ontology redundancy free the later approach is used creating independent filtered private ontologies for current situation analysis.

**Definition 9** (Ontology Filtering). Let  $\mathcal{R}$  be a resource in  $\Omega_f$ . If  $\mathcal{F}$  does not include  $\mathcal{R}$  then set  $\rho(\mathcal{R})$  to ‘doubtful’, for  $\mathcal{R} \in C$  set all  $\rho(I)$ , where  $I \mapsto \mathcal{R} \in \Omega_f$  to ‘doubtful’ and  $\forall \mathcal{R}' \in C$  that are subclasses of  $\mathcal{R}$  ensure that  $I'$ , where  $I' \mapsto \mathcal{R}' \in \Omega_f$ , (a) does not instantiate  $S$ , where  $S \mapsto \mathcal{R}$ . (b)  $I'$  instantiates  $S$ , then create  $S \mapsto \mathcal{R}'$ .

### 5.1 Filter by Author

Filtering by author allows for decisions to be made based on information that a certain author or set of authors provides. Once the user has selected the author all resources that do not have that author as a provider are temporarily removed for the purpose of querying the ontology.

### 5.2 Filter by Affiliation

By supplying an organisational affiliation to each author we are able to create views of the ontology which reflect that of a specific organisation. This allows for the comparison of results between separate organisations.

### 5.3 Filter by Trust

By selecting a threshold value the ontology can be filtered to only show resources that have a positive belief value as described in Section 3.2. This allows for a best case and worst case scenario for a query to be generated. Where a comparison can be made between a query where every author can supply their resources, and the same query over the resource that meet the threshold value.

## 6 Case Study

In the community services sector disparate organisations often combine to apply for tenders. The tender providers need reports to ensure the projects are running according to contracts. In this work we will generate a report after integrating two ontologies. Figure 4 is a simple ontology used to capture data about clients that require counselling, while Figure 5 provides basic overview information on types and numbers of conditions that an individual may be suffering.

An organisation needs to generate a report on the effects of housing conditions on an individuals health. Initially it is necessary to integrate data from two distinct sections of the organisation. Once this is achieved an internal report can be generated. Next it is necessary to integrate an inter-organisational level.

Both ontologies are annotated with the author information,  $Author_1$  and  $Author_2$ , and supplied initial trust values, ‘trust moderately’ and ‘trust neutrally’. The integration process currently uses the author trust value when solving naming conflicts, by selecting the most trusted author’s label as the preferred name (Noy & Musen 2000), thus Client will be selected instead of Patient.

Once the ontology integration process is completed belief propagation for resources occurs, followed by the recalculation of author trust. Figure 6 shows the merged concepts and properties. The ontologies overlapped on the following classes and properties:

- Class Form in both
- Class Person in both
- Class Client and Class Patient
- Property client subclass of Person and Patient subclass of Person
- Property client

These were given mappings from both authors,  $Author_1$  and  $Author_2$ . While the other resources would only have links to their respective authors. The major disagreements that were resolved were that patient, in Figure 5, was renamed to client and Social Worker, in Figure 4 was moved to a subclass of Service Provider. The renaming of patient is a minor edit and did not need belief revision. While the changing of Social Worker would delete the resource, Social Worker subclass of Person, which would give the belief a ‘doubtful’ rating.

Once the resource beliefs were calculated on the threshold, ‘trust neutrally’, it can be seen that only one resource, was considered ‘doubtful’, both author’s trust ratings increase as the number of agreements outweighs the number of disagreements.

## 7 Future Work

In the future, we intend to perform experiments to determine whether a trust model based on Bayesian Probabilities provides better results than a ‘trust level’ approach. Furthermore we will investigate various reasoning approaches that will allow for the incorporation of an ‘evolutionary ontology model’ and an ‘uncertain knowledge model’. By defining an appropriate reasoning algorithm we aim to provide a mechanism for comparing trust models. We also plan to investigate the use of trust in assisting in the ‘ontology mapping’ process to discover if trust can be used to assist in the semi-automation and improved suggestions of current mapping approaches, such as PROMPT (Noy & Musen 2000).



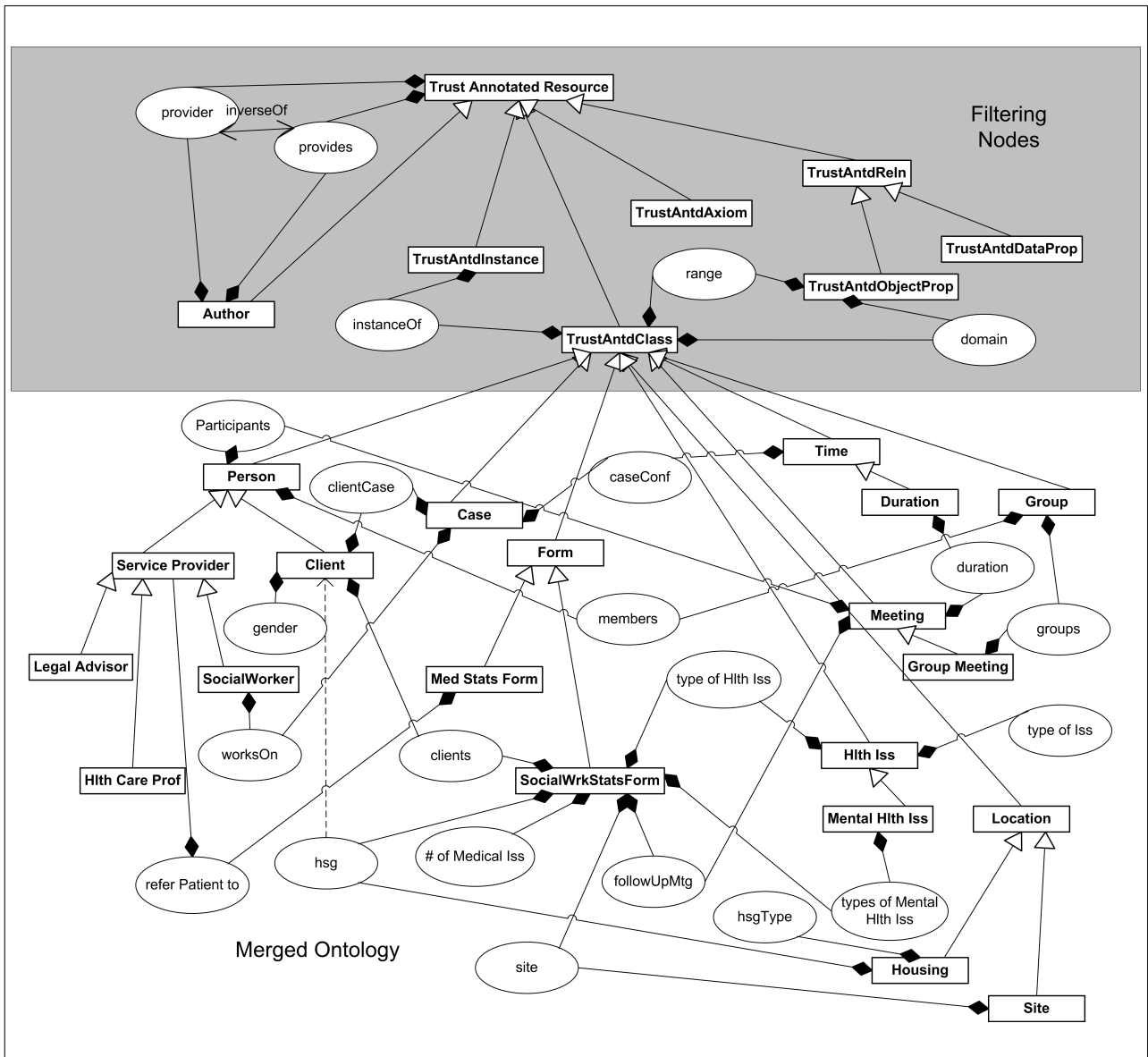


Figure 6: Merged Private Ontology

## 8 Conclusion

It can be seen that by annotating ontologies with authors and trust values the user is provided with extra flexibility when making decision based on queries to ontology. By supplying additional information the ability to compare and contrast results can be achieved. By combining the two ontologies an integrated result is also possible.

Although we are still in the process of implementing our solution it can be seen from initial experiments that the additional information obtained by this solution provides insights to an uncertain domain. It can also be seen that the author and trust annotations can be shown to give a decision maker additional insight into the knowledge. Filters have been implemented to supply differing views of the information contained within.

Additionally by providing an evolutionary model for trust we are able to monitor authors and corporations and ensure that our trust remains valid. By implementing deletion by author we are able to evolve the ontology or regress to a previous version when an author is proved to be providing spurious knowledge.

## References

- Bertino, E., Ferrari, E. & Squicciarini, A. (2004), 'Trust negotiations: concepts, systems, and languages', *Computing in Science and Engineering* **06**(4), 27–34.
- Carroll, J. J., Bizer, C., Hayes, P. & Stickler, P. (2005), 'Named graphs.', *J. Web Sem.* **3**(4), 247–267.
- Ding, L., Zhou, L. & Finin, T. (2003), Trust based knowledge outsourcing for Semantic Web agents, in 'Proc. IEEE/WIC Web Intelligence', pp. 379–387.
- Dou, D., McDermott, D. & Qi, P. (2003), Ontology translation on the semantic web., in 'On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003'.
- Dumbill, E. (2002), 'XML watch: Finding friends with XML and RDF', *IBM's XML Watch*.
- Golbeck, J., Parsia, B. & Hendler, J. (2003), Trust networks on the Semantic Web, in '7th International Workshop, CIA 2003, Aug 27-29 2003', LNAI 2782, Springer-Verlag, Helsinki, pp. 238–249.
- Gonzalez, A. J. & Dankel, D. D. (1993), *The engineering of knowledge-based systems : theory and practice*, Prentice Hall.
- Guarino, N. (1998), Formal ontology and information systems, in 'International Conference on Formal Ontology in Information Systems'.
- Guha, R. V. (2003), Open rating systems, Technical report, Stanford Knowledge System Laboratory.
- Guha, R. V., Kumar, R., Raghavan, P. & Tomkins, A. (2004), Propagation of trust and distrust, in 'Proc.WWW'.
- Huang, J. & Fox, M. (2005), Trust judgment in knowledge provenance, in 'Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on', pp. 524–528.
- Jøsang, A., Ismail, R. & Boyd, C. (2005), A survey of trust and reputation systems for online service provision, in 'Decision Support Systems'.
- Jøsang, A. & Pope, S. (2005), Semantic constraints for trust transitivity., in S. Hartmann & M. Stumptner, eds, 'APCCM', Vol. 43 of *CRPIT*, Australian Computer Society, pp. 59–68.
- Klein, M. & Noy, N. (2003), 'A component-based framework for ontology evolution', *IJCAI'03 Workshop: Ontologies and Distributed Systems, Acapulco Mexico*.
- Marsh, S. & Dibben, M. (2005), Trust, untrust, distrust and mistrust - an exploration of the dark(er) side, in 'iTrust'.
- McGuinness, D., Fikes, R., Rice, J. & Wilder, S. (2000), The Chimaera ontology environment, in 'Proceedings AAAI', Austin, Texas.
- McGuinness, D. & van Harmelen, F. (1994-2006), 'OWL Web Ontology Language Overview'.
- Noy, N. F., Mitra, P. & Jaiswal, A. R. (2004), OMEN: A probabilistic ontology mapping tool, in 'ISWC'04 Workshop on Meaning coordination and Negotiation', Hiroshima.
- Noy, N. F. & Musen, M. A. (2000), PROMPT: Algorithm and tool for automated ontology merging and alignment, in 'Proceedings AAAI', Austin, Texas.
- Noy, N. F. & Musen, M. A. (2003), 'The PROMPT suite: Interactive tools for ontology merging and mapping', *International Journal of Human-Computer Studies* **59**.
- Resnick, P., Kuwabara, K., Zeckhauser, R. & Friedman, E. (2000), 'Reputation systems.', *Commun. ACM* **43**(12), 45–48.
- Zheng, X., Chen, H., Wu, Z. & Zhang, Y. (2006), A computational trust model for semantic web based on bayesian decision theory., in X. Zhou, J. Li, H. T. Shen, M. Kitsuregawa & Y. Zhang, eds, 'APWeb', Vol. 3841 of *Lecture Notes in Computer Science*, Springer, pp. 745–750.
- Ziegler, C.-N. & Lausen, G. (2004), Spreading activation models for trust propagation, in 'Proceedings - 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE 2004'.

# Mephisto I

## Towards a Formal Theory

D. A. Lambert

C. Nowak

Command and Control Division  
 Defence Science and Technology Organisation  
 PO Box 1500, Edinburgh SA 5111, Australia  
 Email: {Dale.Lambert, Chris.Nowak}@dsto.defence.gov.au

### Abstract

Mephisto is a framework that will enable ontology-based high-level information fusion. Within the framework, the name Mephisto can be used to refer to a conceptualisation, an ontology, an agent, a society, a formal theory, and an implementation.

A Mephisto conceptualisation assumes that the world can be understood in terms of processes. A Mephisto ontology is a specification of the conceptualisation. A Mephisto agent employs the ontology. A Mephisto theory is a formal theory of processes. A Mephisto implementation implements the theory.

A Mephisto theory plays a crucial role within the framework.

### 1 Introduction

*Mephisto* (Nowak 2003, Nowak & Lambert 2005) is a framework that will enable ontology-based high-level information fusion. Within the framework, the name Mephisto can be used to refer to a *conceptualisation*, an *ontology*, an *agent*, a *society* of agents, a *formal theory* (or *theories*), and an *implementation*.

A Mephisto conceptualisation assumes that the world can be understood in terms of processes, and five levels of processes should be distinguished for different levels of abstraction, namely Metaphysical, Physical (Environmental), Functional, Intentional (Cognitive), and Social (see (Nowak & Lambert 2005) for justifications on selecting the five levels). A Mephisto ontology is a specification of a Mephisto conceptualisation in a given ontology language. A Mephisto agent is an agent that employs a Mephisto ontology. A Mephisto society is a society of Mephisto agents. A Mephisto theory is a formal theory of Metaphysical, Physical (Environmental), Functional, Intentional (Cognitive), and Social processes. A Mephisto implementation is an implementation of a Mephisto theory.

A Mephisto theory plays a crucial role within the framework—it clarifies the conceptualisation and facilitates the implementation of a Mephisto ontology based agent society.

There is a question of what exactly is meant by a *Mephisto theory*? Even a list of kinds of processes that are to be considered—namely metaphysical, physical, functional, intentional and social (cf. Table 1)—indicates that any Mephisto theory would include primitives for dealing with:

- metaphysical level individuation (selecting a metaphysical process);
- mereological aspects of processes (*fragment*, or *part-of*, relation);
- operations on processes (*meet*, *join* and *complement* functions);
- spatial and temporal processes (*space*, *time*, *spatial*, *temporal*);
- existence (*exists* relation);
- physical level individuation (selecting a physical process);
- topological aspects of processes (*connects* relation);
- orientation (*between* relation);
- distance (*distance* function);
- physical substance, phase change (*solid*, *liquid*, *gas* relation);
- substance properties (*temperature*, *pressure* functions);
- functional level individuation (selecting a functional process);
- functional primitives (*transforms*, *moves*, *senses*, *informs* relations);
- intentional level individuation (selecting an intentional process);
- propositional attitude theory (*believes*, *desires*, *intends* relations);
- social level individuation (selecting a social process);
- social entities (*societies* / *teams* / *groups*);
- social relations (*possesses*, *influences*, *contract*, *agreement*);
- social theories (*theory of conflict* & *agreement*).

Building formal theories for selected small subsets of the above primitives constitute areas of significant research activity. Theories of mereotopology (Varzi 1996), space, time and space-time are still being developed. Example theories such as RCC (Randell, Cui & Cohn 1992), Allen temporal algebra (Allen 1981) or  $\mathcal{ST}_i$  logics (space-time frameworks merging space formalisms with time formalisms (Gabbay, Kurucz, Wolter & Zakharyashev 2003)) are research fields in their own right.

Copyright © 2006, Commonwealth of Australia. This paper appeared at the Australasian Ontology Workshop (AOW 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 72. M. A. Orgun and T. Meyer, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

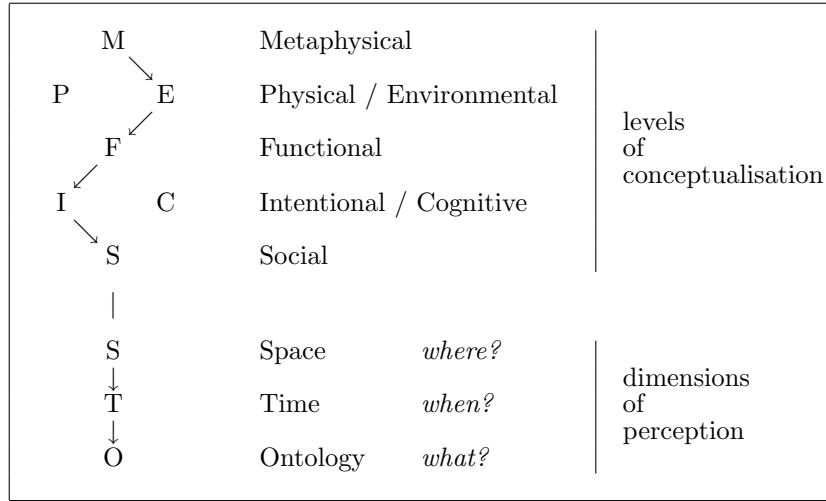


Table 1: Mephisto’s conceptualisation &amp; perception: MEFIS-STO.

Theories of *orientation* and *distance* are not parts of most space/time frameworks, and there are no social, intentional and functional level theories readily available.

In a long term, a core formal theory of processes of the M,P,F,I,S levels will be built. The core theory would include fundamental primitives common to multiple domains, and it would most probably need to be extended for different domains. This core theory—call it Mephisto Theory—would be subject to additions, modifications and selections (not only could new primitives be added, and other ones defined, but subsets of the existing set of primitives would give rise to useful subtheories of Mephisto Theory).

The name<sup>1</sup> *Mephisto* has been used in (Nowak 2003). The framework initially called *MPFIS* (metaphysical, physical, functional, intentional and social processes), was then changed to *MePFIS*<sup>2</sup>, or *Mephisto*.

The structure of the paper is as follows. Section 2 specifies *the domain* of interest; Section 3 discusses *conceptualisations* and *ontologies*; Section 4 considers *agents*; Section 5 describes steps in building *Mephisto Theory*; Section 6 concludes.

## 2 Domain

A selected military scenario provides a domain for the Mephisto ontology, agents and implementation; relevant primitives are listed in Table 2.

The scenario involves friendly and hostile military forces, military platforms (ships and aircrafts), military operations (ship convoys, air attack defences), command and control activities, and situation and threat assessment.

The Commander of the Joint Task Force (CJTF) receives information from *observers* and *radar nodes*. Information fusion activities occur at object, situation and threat assessment levels. Mephisto ontologies and agents provide a computational framework to facilitate the information fusion activities, see Section 4.

<sup>1</sup>For information on what *Mephisto* may refer to, please see the WIKIPEDIA entry at: <http://en.wikipedia.org/wiki/Mephisto>.

<sup>2</sup>The name *MePFIS*<sup>2</sup> was changed to *Mephisto* to break the association with the names used for processes at the five levels.

## 3 Mephisto Conceptualisations & Ontologies

*High-level information fusion* requires an adequate *conceptualisation*. A *process-based* view of the world leads to a *process-based* conceptualisation, such as the Mephisto conceptualisation (Nowak & Lambert 2005).

A Mephisto ontology has been built using OilEd (<http://oiled.man.ac.uk/>), and is processed by Racer, a description logic reasoner (<http://www.racer-systems.com/>). Racer logic is closely related to the description logic *SHIQ* and to the OWL-DL language (Horrocks & Patel-Schneider 2004); Racer allows to reason with *SHIQ* and concrete domains.

Racer ontologies contain individuals, concepts and (binary) relations; Figure 1 presents Mephisto ontology’s concept and relation structures.

## 4 Mephisto Agents and Societies

A *Mephisto agent* is an agent that employs a Mephisto ontology.

Several Mephisto agents have been implemented in ATTITUDE, Prolog and Java, and placed on CoABS Grid (<http://coabs.globalinfotek.com/>). Figure 2 shows the agents.

vCJTF, the *virtual Commander Joint Task Force* is an ATTITUDE agent that receives update information from *observers* and *radar nodes* (not considered here), and communicates with *Racer\_Reasoner* and *Prolog\_Reasoner* in order to perform *situation* and *threat assessment*.

*Racer\_Client* is a Grid agent that can connect to *Racer\_Server* and therefore provides Racer system’s reasoning capabilities to other agents.

*Racer\_Reasoner* is a Grid agent, implemented in ATTITUDE, that extends the capabilities of the Racer reasoning engine (allows n-ary relations and higher order relations; retrieves facts stored in the ontology).

*Prolog\_Reasoner* is a Grid agent, implemented in Prolog; it conforms to the Mephisto conceptualisation, and provides (partial) implementation of Mephisto Theory in Prolog. Further, *Prolog\_Reasoner*’s KBs are Prolog versions of the Racer ontology used for dynamic information.

domain (of processes) A	constant $c_i \in A$	function $f_i : A^k \longrightarrow A$	relation $r_i \subseteq A^k$
Metaphysical			<b>fragment</b>
		<b>join</b> <b>meet</b>	
		<b>complement</b>	
	<b>nothing</b> <b>everything</b>		
		<b>space</b> <b>time</b>	
			<b>spatial</b> <b>temporal</b> <b>exists</b>
Physical			<b>connects</b> <b>between</b>
		<b>distance</b>	
			<b>land</b> <b>air</b> <b>water</b>
Functional			<b>transforms</b>
			<b>moves</b>
			<b>senses</b>
			<b>informs</b> <b>strikes</b>
			<b>operational</b>
Intentional			<b>has attitude</b>
			<b>believes</b> <b>desires</b> <b>intends</b>
Social			<b>owns</b> <b>hostile</b> <b>capable</b> <b>threat</b>

Table 2: MPFIS constants, functions and relations.

## 5 Mephisto Theories

This section describes first steps towards building Mephisto Theory for the M,P,F,I,S levels, and connections between the levels.

### 5.1 Towards Metaphysical Theory

Metaphysical Theory calls any *fragment* of the spatio-temporal universe a *process*. The following list of definitions and axioms leads to a theorem<sup>3</sup> stating that processes form a Boolean algebra.

#### Definition 5.1 (processes)

Let  $(P, \leq, \equiv)$  be a set of processes, with a fragment and an identity relation. If  $p_1, p_2, p_3, p_4 \in P$  and  $p_1 \leq p_2$  and  $p_3 \equiv p_4$  then  $p_1$  is said to be a fragment of  $p_2$ , and  $p_3$  is said to be identical to  $p_4$ .

#### Axiom 5.1 (Identity, Fragmentation and Universe Axioms)

$$\begin{aligned} \forall x, y \in P [x \equiv y \Leftrightarrow \forall z \in P [z \leq x \Leftrightarrow z \leq y]]. \\ \forall x, y \in P [x \leq y \Leftrightarrow \forall z \in P [z \leq x \Rightarrow z \leq y]]. \\ \exists x \in P \forall y \in P [y \leq x]. \end{aligned}$$

#### Definition 5.2 (universe $\Omega$ )

$$z \equiv \Omega \text{ iff}_{\text{def}} \forall y \in P [y \leq \Omega].$$

#### Axiom 5.2 (Join and Meet Axioms)

$$\begin{aligned} \forall x, y \in P \exists z \in P [x \leq z \ \& \ y \leq z \ \& \ \forall u [(x \leq u \ \& \ y \leq u) \Rightarrow z \leq u]]. \\ \forall x, y \in P \exists z \in P [z \leq x \ \& \ z \leq y \ \& \ \forall u [(u \leq x \ \& \ u \leq y) \Rightarrow u \leq z]]. \end{aligned}$$

#### Definition 5.3 (join and meet)

$$\begin{aligned} z \equiv x + y \text{ iff}_{\text{def}} (x \leq z \ \& \ y \leq z \ \& \ \forall u [(x \leq u \ \& \ y \leq u) \Rightarrow z \leq u]). \\ z \equiv x \bullet y \text{ iff}_{\text{def}} (z \leq x \ \& \ z \leq y \ \& \ \forall u [(u \leq x \ \& \ u \leq y) \Rightarrow u \leq z]). \end{aligned}$$

#### Axiom 5.3 (Distribution Axiom)

$$\forall x, y, z \in P [x \bullet (y + z) \leq (x \bullet y) + (x \bullet z)].$$

#### Axiom 5.4 (Difference Axiom)

$$\forall x, y, z \in P [x \equiv z + (x \bullet y) \ \& \ \forall u [(u \leq z \ \& \ u \leq (x \bullet y)) \Rightarrow \forall v [v \equiv u + v]]].$$

#### Definition 5.4 (difference)

$$z \equiv x - y \text{ iff}_{\text{def}} (x \equiv z + (x \bullet y) \ \& \ \forall u [(u \leq z \ \& \ u \leq (x \bullet y)) \Rightarrow \forall v [v \equiv u + v]])$$

#### Definition 5.5 (complement)

$$z \equiv \bar{x} \text{ iff}_{\text{def}} z \equiv \Omega - x.$$

#### Definition 5.6 (nothing $\perp$ )

$$z \equiv \perp \text{ iff}_{\text{def}} z \equiv \bar{\Omega}.$$

#### Theorem 5.1 (process Boolean Algebra)

$(P, +, \bullet, -, \perp, \Omega)$  is a Boolean Algebra.

The Boolean algebra of processes is further extended by adding *existence* and *space* and *time* primitives:

$$(P, +, \bullet, -, \perp, \Omega, \text{exists}, \text{space}, \text{time}, \text{spatial}, \text{temporal}),$$

where:

$$\text{exists}(x) \text{ iff}_{\text{def}} \neg(x \equiv \perp),$$

$$\text{space, time: } P \longrightarrow P,$$

<sup>3</sup> Theorems (e.g., stating that:  $\equiv$  is an equivalence relation,  $\leq$  is an order relation, *universe*, *join*, *meet* and *difference* are unique) have not been included here.



Figure 1: Mephisto, Concepts, Relations.

$$a \equiv_s b \text{ iff}_{\text{def}} \mathbf{space}(a) \equiv \mathbf{space}(b),$$

$$a \equiv_t b \text{ iff}_{\text{def}} \mathbf{time}(a) \equiv \mathbf{time}(b),$$

$$\mathbf{spatial}, \mathbf{temporal} \subseteq P,$$

$$\mathbf{spatial}(a) \text{ iff}_{\text{def}} \mathbf{space}(a) \equiv a,$$

$$\mathbf{temporal}(a) \text{ iff}_{\text{def}} \mathbf{time}(a) \equiv a;$$

further, given:

$$P_s = \{p \in P \mid \mathbf{spatial}(p)\} \text{ and}$$

$$P_t = \{p \in P \mid \mathbf{temporal}(p)\},$$

the structures:

$$(P_s, +, \bullet, -, \perp, \Omega) \text{ and}$$

$$(P_t, +, \bullet, -, \perp, \Omega)$$

are Boolean algebras that are subalgebras of  $(P, +, \bullet, -, \perp, \Omega)$ .

## 5.2 Towards Physical Theory

Physical Theory is concerned with physical substances: relations **solid**, **liquid** and **gas** provide a way of classifying substances. In the case of the domain described in Section 2, relations **land**, **water** and **air** are employed; they allow to process information about location of assets (on land, water or air, respectively).

An important question is what space-time primitives should be included in the theory of this level.

It seems that although the mereological primitive *part-of* (or *fragment*) belongs to the *metaphysical* level, the topological primitive *connects* belongs to the *physical* level, for it allows *physical individuation* of objects obtained by *wholeness/unity/self-connectedness* (that can be defined in terms of the **connects** relation of RCC).

If this stance is taken, then a *mereotopological theory* splits across the metaphysical/physical border, and **fragment** (*part-of*) is a metaphysical level primitive, while **connects** is a physical level primitive; this stance is taken in Table 2. As a consequence, other space-time relations for *orientation* and *distance* are put at the physical level; hence, **between** (an *orientation* primitive) and **distance** at the physical level in Table 2.

The above assumes a way of formalising *space-time* (rather than formalising *space*, formalising *time*, and merging the two). In such an approach, the (*topological*) primitive **connects** is applied to *space-time* regions, and so is the (*orientation*) primitive **between**. This is an attractive approach, see (Stell 2000, Stell & West 2004, Muller 2002, Vakarelov, Düntsch & Bennett 2001); and this is the preferred approach.

Usually however, rather than building a theory of *space-time* (and then *analysing* it to obtain a theory of *space* and a theory of *time*), one has two separate theories, a theory of *space* and a theory of *time* (synthesising, or *merging* them to obtain a theory of *space-time*). If this is the approach taken, then e.g. RCC and Allen (Randell et al. 1992, Allen 1981) can be employed to provide *spatial* and *temporal* rea-

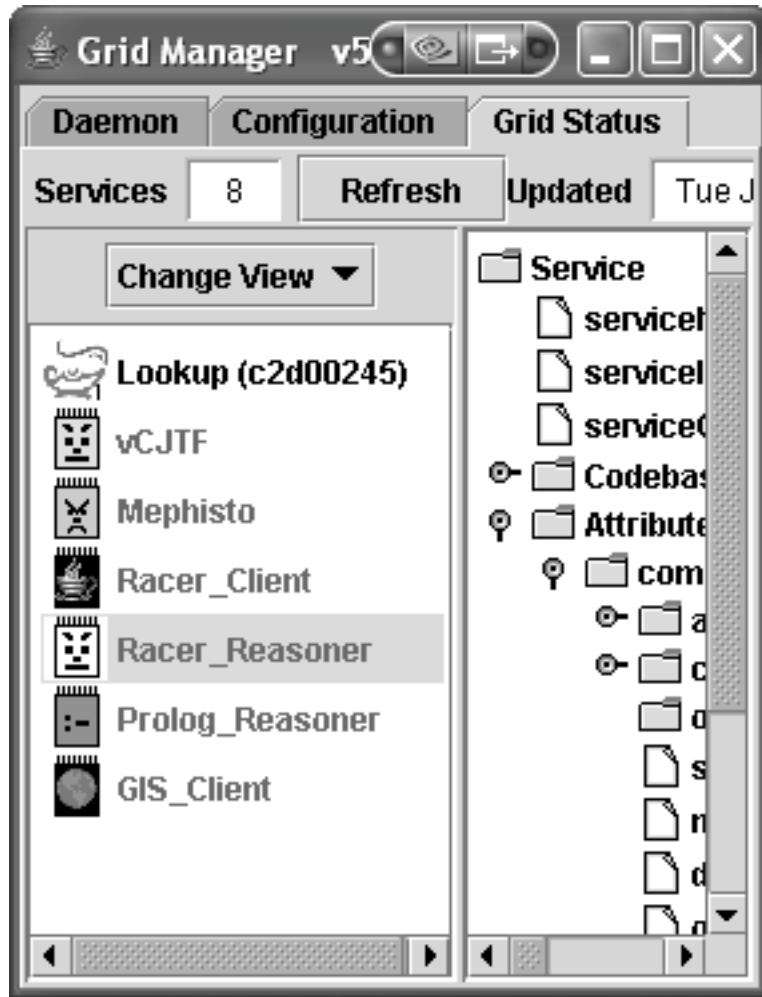


Figure 2: Agents on CoABS Grid.

soning, respectively. Currently, this is the approach taken in the Mephisto implementation, with the RCC and Allen<sup>4</sup> frameworks being implemented in Prolog and forming parts of `Prolog_Reasoner`; in future, *space-time* approach will be attempted.

### 5.3 Towards Functional Theory

Functional Theory should allow to reason about functional capabilities of assets, including military platforms such as aircrafts and ships.

It has been decided that a **transforms** relation is a primitive, a relation **moves** is defined in terms of **transforms**, and relations **senses**, **informs** and **strikes** are defined in terms of **moves**.

$$\text{transforms}(z, x \bullet t_x, y \bullet t_y) \Rightarrow \text{before}(t_x, t_y)$$

$$\begin{aligned} \text{moves}(z, x \bullet s_x, y \bullet s_y) \text{ iff}_{\text{def}} \\ \text{transforms}(z, x \bullet s_x, y \bullet s_y) \ \& \ s_x \neq s_y \end{aligned}$$

$$\begin{aligned} \text{senses}(\text{sensor} \bullet s_s \bullet t_s, \text{target}) \text{ iff}_{\text{def}} \\ \text{moves}(\text{target}, \text{pulse} \bullet s_{p1} \bullet t_{p1}, \text{pulse} \bullet s_{p2} \bullet t_{p2}) \\ \& \ s_s \equiv s_{p2} \ \& \ t_s \equiv t_{p2} \end{aligned}$$

$$\begin{aligned} \text{informs}(\text{transm} \bullet s_t \bullet t_t, \text{rcvr} \bullet s_r \bullet t_r, \text{info}) \text{ iff}_{\text{def}} \\ \text{moves}(\text{transm}, \text{info} \bullet s_{i1} \bullet t_{i1}, \text{info} \bullet s_{i2} \bullet t_{i2}) \\ \& \ s_r \equiv s_{i2} \ \& \ t_r \equiv t_{i2} \end{aligned}$$

<sup>4</sup>It should be noted that the temporal primitive *meets* incorporates both *connects*-like (topology) and *between*-like (orientation) primitives.

$$\begin{aligned} \text{strikes}(\text{strkr} \bullet s_s \bullet t_s, \text{trgt} \bullet s_t \bullet t_t, \text{weapn}) \text{ iff}_{\text{def}} \\ \text{moves}(\text{strkr}, \text{weapn} \bullet s_1 \bullet t_1, \text{weapn} \bullet s_2 \bullet t_2) \\ \& \ s_t \equiv s_2 \ \& \ t_t \equiv t_2 \end{aligned}$$

As indicated in Table 2, a relation **operational** also needs to be considered; this in turn leads to such relations as **neutralised**, **disrupted**, **damaged** and **destroyed**. The relation **strikes** can be *specialised* to such relations as **strikes-and-damages** and **strikes-and-destroys**.

### 5.4 Towards Intentional Theory

Intentional Theory needs to incorporate a *theory of propositional attitudes*, and in particular such attitudes as **believes**, **desires** and **intends**. ATTITUDE—a multi-agent programming language—is an implementation of an Intentional Theory (ATTITUDE is also used to implement most of the Mephisto agents). This level does not constitute a difficulty, although a formal Intentional Theory needs to be extracted from ATTITUDE.

### 5.5 Towards Social Theory

Social Theory is a challenge. It also is the most important one, for it embraces all the theories of the lower levels, and in complex domains (domains where social interactions come into play) it provides the highest level view of the domain.

The distinguishing feature of this level is that multiple agents are involved, agents forming groups, teams and societies. It is communication, negotiation,

collaboration, agreement—and other interactions—between agents that dominate the level, and in fact all the levels.

It is suggested here that a *theory of agreement and conflict* is a core of Social Theory. There are many social level notions that capture crucial aspects of social interactions. *Possession* and *ownership* seem important, and so do *trading contracts*, and *contracts* in general. Many social interactions can be considered *agreement interactions*. However, social agents sometimes fail to achieve agreements—they enter *conflicts*. Conflicts in turn seem to be characterised by *hostility* and *threat*.

The above suggests that the following list of (selected) social level relations are of interest: *possession*, *ownership*, *contract*, *trading*, *agreement*, *conflict*, *hostility* and *threat*. Some initial steps in formalisation of the social level are suggested below.

$$\begin{aligned} \text{owns}(x, y, z) &\text{ iff}_{\text{def}} \text{ possesses}(x, y) \ \& \ \text{ legal-contract}(z) \\ \text{trades}(u, v, w, x, t_1, t_2, y) &\text{ iff}_{\text{def}} \\ &\text{ owns}(u \bullet t_1, v \bullet t_1) \ \& \ \text{ owns}(w \bullet t_1, x \bullet t_1) \ \& \\ &\text{ owns}(u \bullet t_2, x \bullet t_2) \ \& \ \text{ owns}(w \bullet t_2, v \bullet t_2) \ \& \\ &\text{ legal-contract}(y) \\ \text{trades}(u, v, w, x, t_1, t_2, y) &\Rightarrow \text{ agrees}(u, w, y) \\ \neg \text{ agrees}(u, w, y) &\Rightarrow \\ &\Rightarrow \text{ conflict}(u, w, y) \Rightarrow \\ &\Rightarrow \text{ hostile}(u, w) \ \& \ \text{ threat}(u, f(y)) \ \& \\ &\ \& \ \text{ threat}(w, g(y)) \\ \text{threat}(x, y) &\text{ iff}_{\text{def}} \text{ capable}(x) \ \& \ \text{ hostile}(x, y) \end{aligned}$$

## 6 Conclusion

In this paper some aspects of the Mephisto framework have been presented. Significant portions of the Mephisto conceptualisation have been built, and the conceptualisation has been *specified* in the Racer/OilEd language, see Figure 1. A little society of Mephisto agents have been implemented in ATTITUDE, Prolog and Java, and placed on the CoABS Grid, see Figure 2. The Mephisto implementation is at this stage used to test and experiment with Mephisto Theories (although it does also provide an implementation of the *scenario*, or it's fragments). Mephisto Theories form the essence of the Mephisto effort: it is the theories that will make Mephisto a success, or otherwise.

Mephisto Theories differ from theories of e.g., space, time, or knowledge in an important way: it is not only that the theories need to be built and their meta-level properties (such as soundness, completeness, decidability, tractability) established; it is a significant task to decide what functions and relations are of interest, which of these are to be primitives, and how can other functions and relations be defined in terms of the primitives. The scope of this task is enormous; and this paper reports a miniscule first step.

A formalisation of Mephisto in Isabelle/HOL has been undertaken, but this is not reported here.

## References

- Allen, J. F. (1981), An interval-based representation of temporal knowledge, in 'Proceedings 7th IJCAI', pp. 221–226.
- Gabbay, D., Kurucz, A., Wolter, F. & Zakharyashev, M. (2003), *Many-Dimensional Modal Logics*, Elsevier.

- Gabelaia, D., Kontchakov, R., Kurucz, A., Wolter, F. & Zakharyashev, M. (2005), 'Combining spatial and temporal logics: Expressiveness vs. complexity', *JAIR* **23**, 167–243.
- Horrocks, I. & Patel-Schneider, P. (2004), A Proposal for an OWL Rules Language, in 'World Wide Web Conference'.
- Lambert, D. A. (2003), Grand Challenges of Information Fusion, in 'Proceedings of the Sixth International Conference on Information Fusion'.
- Lambert, D. A. (2006a), Formal Theories for Semantic Fusion, in 'Proceedings of the Ninth International Conference on Information Fusion'.
- Lambert, D. A. (2006b), A Unification of Sensor and Higher-Level Fusion, in 'Proceedings of the Ninth International Conference on Information Fusion'.
- Muller, P. (2002), 'Topological Spatio-Temporal Reasoning and Representation', *Computational Intelligence* **18**(3), 420–450.
- Nowak, C. (2003), On Ontologies for High-Level Information Fusion, in 'Sixth International Conference on Information Fusion'.
- Nowak, C. & Lambert, D. (2005), The Semantic Challenge for Situation Assessments, in 'Eighth International Conference on Information Fusion'.
- Randell, D. A., Cui, Z. & Cohn, A. G. (1992), A Spatial Logic based on Regions and Connection, in 'KR'92'.
- Renz, J. (1998), A canonical model of the region connection calculus, in 'KR'98: Principles of Knowledge Representation and Reasoning'.
- Stell, J. G. (2000), 'Boolean connection algebras: A new approach to RCC', *Artificial Intelligence* **122**(1–2).
- Stell, J. G. & West, M. (2004), A Four-Dimensionalist Mereotopology, in 'FOIS-2004', IOS Press.
- Vakarelov, D., Düntsch, I. & Bennett, B. (2001), A note on proximity spaces and connection based mereology, in 'FOIS'01', ACM.
- Varzi, A. C. (1996), 'Part-Whole Relations: The Prospects of Mereotopology', *Data and Knowledge Engineering* **20**.



# Towards Scalable Ontology Engineering Patterns: Lessons Learned from an Experiment based on W3C's Part-whole Guidelines

Laurent Lefort, Kerry Taylor and David Ratcliffe

CSIRO ICT Centre

GPO Box 664 Canberra ACT 2601, Australia

{laurent.lefort, kerry.taylor, david.ratcliffe}@csiro.au

## Abstract

This paper presents an empirical evaluation of description logic reasoners to support the selection of scalable ontology engineering patterns for TBox reasoning. Our main objective is to define the rationale behind the design decisions required for the generation of large ontologies with XSLT-based tools. We discuss here the outcomes of an experiment focusing on aircraft components and parts for which we have implemented the ontology design guidelines for part-whole relationships published by W3C's Semantic Web best practices working group. We have worked with the following reasoners, being the best state-of-the-art currently available: FaCT++, RACER, Pellet and CEL. We found considerable variation in reasoner performance and have attempted to characterise the factors that distinguish the reasoners to enable a best-practice design style to be successfully applied for the generation of very large ontologies.

**Keywords:** Description logics, reasoner, classification, performance.

## 1 Introduction

### 1.1 Motivation

Experimental evaluations of advanced description logic (DL) reasoners have been performed, but there is a lack of systematic evaluation of modern reasoner performance where large-scale ontology reasoning (TBox-reasoning) is emphasised in preference to large-scale instance reasoning. As a general rule, based on our own experience to date as well as the literature, numbers of concepts in the low thousands are likely to be enough to create difficulty for OWL-DL reasoners. One particular challenge is to understand which reasoners perform better for each style of ontology, and this is difficult when the ontologies used for the evaluation are built on a large range of ontology engineering patterns (OEP).

Our objective is to provide a more systematic evaluation to support design decisions that trade off the choice of the more appropriate pattern and the scalability constraints imposed by the available DL reasoners. Now we can use ontologies based on guidelines produced by the W3C's Semantic Web Practice and Deployment<sup>1</sup> group (Schreiber 2006, KnowledgeWeb 2005) to better judge the performance of DL reasoners.

### 1.2 Related work

From our review of the literature, we have identified three main categories of performance evaluation studies: reasoner, benchmark and production studies.

- **Reasoner studies** are papers published by the authors of DL reasoners and are more likely to focus on specific DL features and the selection of the testing samples is often biased towards the newly available or improved features.
- **Benchmark studies** are papers published by end users trying to understand the difference between DL reasoner products and constructing specific benchmarks to support their analysis. The Lehigh University Benchmark (LUBM) is a popular benchmark designed against a model of ontology expressiveness with domain-realistic queries for ABox reasoning.
- **Production studies** are papers published by end users trying to use reasoners over their own large ontologies, and may or may not include comparative analyses of several DL products. The goal is to build ontologies for a specific purpose and then to use them, so the size, the complexity and the overall quality of the ontology are generally higher than what is evaluated elsewhere. The issue with such ontologies is that reasoners may or may not cope with the full content.

---

Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at the Australasian Ontology Workshop (AOW 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 72. M. A. Orgun and T. Meyer, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

---

<sup>1</sup> SWBPD <http://www.w3.org/2001/sw/BestPractices/>

**Table 1** is a summary of the existing literature based on these criteria.

Study / Group of studies	Study type
Haarslev et al., 2004, 2005 (RACER)	Reasoner + Benchmark
Sirin et al., 2005, 2006 (Pellet)	Reasoner + Benchmark
Tsarkov & Horrocks 2005a, 2005b (FaCT++)	Reasoner
Motik et al., 2002, Motik 2006 (Kaon 2)	Reasoner + Benchmark
Baader et al., 2005, 2006 (CEL)	Reasoner
Guo et al., 2003, 2004 (LUBM)	Benchmark
Gardiner et al., 2006	Benchmark
Dameron et al., 2005	Production

**Table 1: Published studies**

Our approach combines the merits of the benchmark and the production types of studies. We benefit here from the reuse of large domain-specific inputs. With the help of XO, our ontology generation tool (Lefort & Taylor 2005), we have generated an aircraft ontology describing components and parts from the Service Difficulty Reports (SDR) published by the Federal Aviation Administration (FAA 2005). The resulting ontology has 4000 classes. To enrich the performance evaluation, we have split the ontology into modules of various sizes accorded to the functional hierarchy of the industry-specific ATA/JASC coding system (FAA 2002).

### 1.3 Ontology normalisation and patterns

Rector (2003, 2005) defines ontology normalization as the application of a limited number of criteria to eventually “let the reasoner do multiple classification”. This is done through the construction of complex primitives as “a conjunction of one class and a boolean combination of zero or more restrictions”.

This experience in the creation of TBox-based ontologies has been disseminated through the W3C best practice group as shown in **Table 2**.

Normalisation (Rector)	Ontology Engineering Pattern documents (W3C)
Upper skeleton axioms	N-ary Relations
Complex primitives w/ property restrictions	Simple part-whole relations
Complex primitives w/ cardinality restrictions	Qualified restrictions cardinality
Refining primitives	Specified Values

**Table 2: Relationships between W3C’s OEP and Rector’s Ontology Normalization**

To enable a more scalable ontology engineering practice, we need to study how the reasoners can cope with the recommended ontology engineering patterns defined by the W3C OEP task force. The pattern selected for this study is the one described in the “Simple part-whole relations in OWL ontologies” document edited by Rector & Welty (2005).

### 1.4 Patterns for part-whole relations

Rector & Welty (2005) first provide useful advice on how to handle the *isPartOf* relation with two properties: one transitive property for the general case completed with a sub-property to handle “direct” relations. It also recommends focus on the *isPartOf* relation rather than on the inverse *hasPart* one, and to avoid using the two together because of scalability issues with cross-referenced existential restrictions.

Rector & Welty’s 2005 pattern “N.4” is for the propagation of properties along the part-whole hierarchy. It aims at an ontology combining a part inventory with a fault finding system to describe the devices made in a factory with a focus on the part-whole relations between parts and sub-parts. Its goal is to enable the inference that a fault in a part is a fault in the whole at different levels of the part-whole hierarchy. In this paper, we apply this pattern to manage the *hasFunction* relation with respect to the functional hierarchy provided by the ATA coding system.

### 1.5 Outline of the paper

This paper is structured in four parts. Section 2 outlines the goal of the experiment and section 3 describes the method used to evaluate the reasoner performance. The results and our supportive analysis are included in Section 4. Section 5 discusses what is required to enable a more scalable ontology engineering practice, both in terms of guidelines and tools. Section 6 concludes and identifies the opportunities to extend this work.

## 2 Experimental goals

The objective of this experiment is to understand the impact of the ontology design patterns on reasoner performance and the differences between reasoners and the effectiveness of their optimization tactics. We have created several variants of the part-whole ontologies to study specific DL reasoners and to strengthen the evaluation outcomes. Because the CEL reasoner (Baader et al. 2006) cannot represent concept disjunction (or *unionOf* in OWL), we have generated an alternative ontology without this part of the pattern. We have also evaluated the reasoner performance on ontologies using simultaneously roles and their inverses, e.g. *isPartOf* and *hasPart*, because this practice is flagged as a cause of problems in the Rector-Welty paper.

## 2.1 The part-whole ontology engineering pattern

The example used by Rector and Welty describes the OWL pattern for an ontology about:

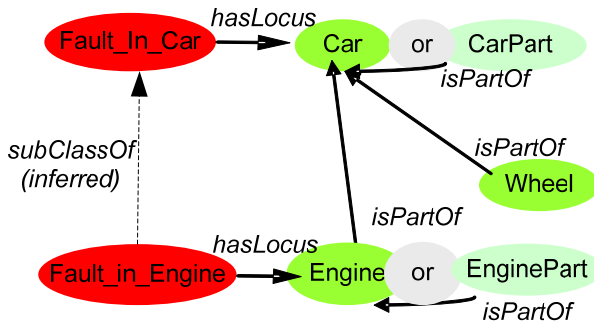
- A part inventory for the devices made in a factory with the relation between each part and its sub-parts.
- A fault finding system for a device in which we want to progressively narrow down the functional region of the fault.

This pattern is based on the DL expression which uses the abstract syntax for OWL defined by W3C (Patel-Schneider et al. 2004).

```
Class(FaultInCar complete intersectionOf
(Fault
restriction (isLocusOf someValuesFrom
(unionOf (Car
restriction (isPartOf someValuesFrom (Car)))))))
```

(From Rector & Welty (2005))

A graphical view is given in **Figure 1**. CarPart is used to represent the anonymous class defined by the existential restriction (isPartOf someValuesFrom (Car)).



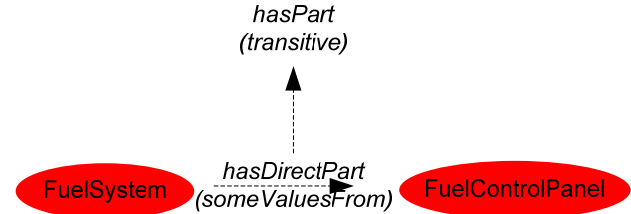
**Figure 1: Rector-Welty pattern N.4 adapted from Rector and Welty (2005)<sup>2</sup>**

This pattern can be repeated and thereby propagated at each level of the hierarchy, first for the Engine class as illustrated in **Figure 1** and then repeatedly. Let's suppose this pattern is applied again for sub-parts of Engine such as Crankcase. Once this is done, the reasoner will be able to infer subsumption relations between faults by going back up the part-whole hierarchy. A fault at the lowest level (a fault in a bolt in the crankcase) will be inferred as a fault in the crankcase and then as a fault in the engine and eventually as a fault in the car. The *isPartOf* relation

must be declared as transitive for the pattern to work repeatedly up the part-whole hierarchy.

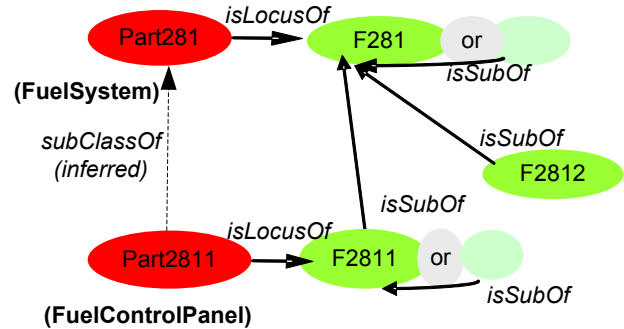
## 2.2 Application to aircraft ontologies

The Rector-Welty recommendations are directly transposable to the aircraft domain we are interested in. Figure 2 shows an example of relation between part and whole derived from the SDR inputs.



**Figure 2: Basic Part-Whole relation**

It can be applied to manage the relation between components/parts and function and exploit the functional hierarchy provided by the ATA coding system. In this case study, our approach is to apply the same pattern to propagate the functional hierarchy down the part-whole hierarchy. This is illustrated in Figure 3.



**Figure 3: Functional hierarchy of physical parts**

Here, we create supplementary classes for each function (e.g. Part281, Part2810) and then use the relation between the artefacts and the functions to infer where each named component (e.g. Fuel System) and part (e.g. Fuel Control panel) stands in the ATA-based hierarchy of classes.

## 2.3 Compatibility with DL languages

The Rector-Welty part-whole pattern N.4 uses a subset of the OWL Lite description logic language known as *SHF*, which is supported by all the reasoners belonging to the OWL-DL family such as FaCT++, RACER and Pellet. According to Zolin (2006), the theoretical complexity for SHF, is ExpTime-complete.

Our analysis of the literature suggests that reasoner performance problems occur with ontologies expressed in *ALC* using a large number of existential restrictions. In practice, these reasoners will differ mostly by the optimisations implemented in their tableau-based algorithms.

<sup>2</sup> (Figure 1., available from draft version 0.2 only <http://www.cs.man.ac.uk/~rektor/swbp/simple-part-whole/simple-part-whole-relations-v0-2.html> )

Theoretical complexity is not the same with *acyclic* and *cyclic* TBoxes<sup>3</sup>. A cyclic TBox is one that contains concept inclusion axioms that reference the same (or equivalent) classes on both side of the subsumption relation (known as a terminological cycle). In other words, a terminological cycle is one that defines a concept in terms of itself in some way. Generally speaking, complexity results for cyclic Tboxes are worse than for the same language in the presence of cycles (Nebel 1991, Baader 2003).

$\mathcal{EL}+$  is a description logic that does not belong to the OWL family. The reasoners implementing  $\mathcal{EL}+$ , such as CEL (Baader et al., 2005), can offer a different trade-off between tractability and expressivity. Inference procedures, such as checking for ontology consistency, concept satisfiability, subsumption and instance checking have known deterministic polynomial time complexity, even for cyclic TBoxes. On the other hand,  $\mathcal{EL}+$  users lose several OWL features, most notably value restrictions, i.e. *allValuesFrom*, which is necessary for OWL domain and range assertions. Nor can they use concept disjunction or *unionOf* in OWL.

This is why we cannot use CEL to reason over ontologies applying the Rector-Welty pattern, but we study CEL for a substantially different perspective on reasoning.

### 3 Evaluation method

#### 3.1 Generation of the test ontologies

XO (Lefort & Taylor 2005), is a tool to build transformations/conversions from XML to OWL. XO allows us to get the variability in ontology size and complexity required for this study. Because we import real world data from existing sources, this study has some elements which are specific to the aircraft maintenance domain. We process the Service Difficulty Reports to get information about parts, their physical and functional relations to other parts and their functional affiliation to the categories defined by ATA/JASC coding system (FAA 2002).

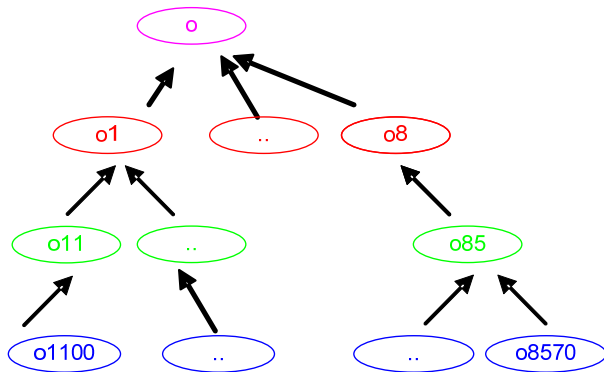


Figure 4: Multi-levels ontology test samples

**Figure 4** illustrates how we work at multiple levels of this hierarchy to generate ontologies of variable size.

- The *aircraft ontology* [o] covers all ATA codes;
- each *one-digit ontology* [o1..o8] comprises between 0 and 1000 four-digit ATA codes;
- each *two-digit ontology* [o11..o85]) comprises between 0 and 100 four-digit ATA codes;
- And each *four-digit ontology* [o1100..o8570] comprises between 0 and 10 ATA codes.

This working environment is flexible enough to allow us to study some variants of the Rector-Welty patterns: for example, to evaluate CEL, we have generated another whole set of ontologies for which we have removed the constructs from the Rector-Welty pattern which are not allowed in  $\mathcal{EL}+$ .

#### 3.2 Evaluation of reasoner performance

*Classification* can be defined as the computation of the subsumption hierarchy for classes and properties. In this experiment, we consider classification to be the crucial TBox reasoning task. Gardiner et al. (2006) describe a method to trigger the classification and to measure its time for any reasoner with a DIG interface. In this method, the classification time is obtained from the response time to a query on the satisfiability of the top concept, requested once the ontology has been loaded. We have used a comparable method for FaCT++, RACER and Pellet. For CEL, the classification time is measured internally and is provided as an output. We have also used a more direct measurement method for Pellet corresponding to the published benchmark results.

#### 3.3 Extraction of supporting metrics

Additional size and complexity indicators are required to help us to study the impact of the Ontology Engineering Patterns (or OEP) on the reasoner performance. For each sample, we have recorded the number of classes, properties, and triples and the number of cycles flagged by RACER.

We have also defined two specific indicators for this study to help us to analyse how we apply the Rector-Welty pattern described above. The first one is for the top part of the Rector-Welty pattern (1) and the second one is for the bottom part (2).

We define the *number of times we use the Rector-Welty pattern* as the number of times we use constructs like:

```
Class(Entity281 complete intersectionOf
(Entity
restriction (hasFunction someValuesFrom
(unionOf (Function 281
restriction (isSubOf someValuesFrom (Function281)))))))
```

(1) *Class counted as one use of the Rector-Welty pattern*

<sup>3</sup> The DL Complexity navigator web page from Zolin (2006) can give the theoretical complexity for acyclic and for “general” TBoxes for the relevant range of DLs.

We define the *number of classes involved in the Rector-Welty pattern* as the number of times we use constructs like:

```
Class(Entity2810 complete
  intersectionOf (Entity
    restriction hasFunction someValuesFrom Function2810))
```

(2) *Class involved in the Rector-Welty pattern*

We define the *number of isLocusOf existential restrictions* as the number of times we use constructs like the one present in (3). A given class may have one or more of such existential restrictions:

```
Class(FuelControlPanel partial intersectionOf
  (Entity
    restriction isLocusOf someValuesFrom Function2810))
```

(3) *Class with one isLocusOf existential restriction*

For the Rector-Welty based ontologies, we also use the number of *isPartOf* existential restrictions to measure the number of constructs of the form presented in **Figure 2**.

To monitor the presence of “cyclic axioms” (Haarslev et al. 2005), we also extract the number of cycles from RACER’s warning messages.

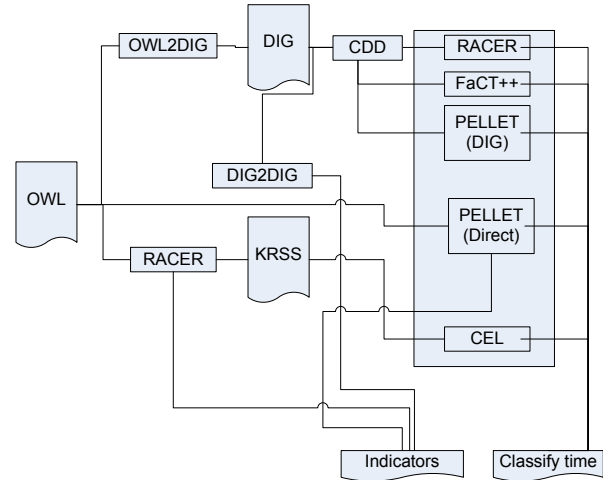
**Table 3** provides a summary of the indicators which are used for this study:

Indicators
Number of classes
Number of properties
Number of triples
Number of cycles from RACER (option -v)
Number of times we use the Rector-Welty pattern
Number of classes involved in the Rector-Welty pattern
Number of <i>isLocusOf</i> existential restrictions ( <b>relations between entities and functions</b> )
Number of <i>hasDirectPart</i> existential restrictions ( <b>relations between components and parts</b> )
Total number of existential restrictions

**Table 3: Complexity indicators**

### 3.4 Experimental setup

Our XO tool is available through an ANT-based working environment which can be used to create the ontologies, to evaluate the performance of reasoners and to provide the supporting complexity indicators of **Table 3**. **Figure 5** gives an overview of this working environment.



**Figure 5: Experiment setup**

In the figure, OWL2DIG is the tool developed by Zhang and Zhou available from SemWebCentral<sup>4</sup>. CDD, the tool used to interact with the DIG reasoners is derived from a subset of the Context-Driven Development Toolkit<sup>5</sup> developed by Wagelaar to provide a service equivalent to the approach described by Gardiner et al (2006). The authors’ own XSL transformation Dig2Dig is used to tidy the DIG file and to generate some of the size and complexity indicators specifically defined for this study. RACER is also used directly to convert ontologies from OWL to KRSS, the format required by the version of CEL used in this experiment.

We can use this setup to compare the results for Pellet in two execution modes, when it is used directly and when it is used as a DIG server. The results presented here correspond only to the direct execution mode.

**Table 4** lists the tool versions and the more critical configuration parameters we have used.

Tool	Version	Parameters
FaCT++	1.1.3	N/A
RACER	1.7.24	Stack size 45000000
Pellet	1.3	Max heap size (jvm) 800m
CEL	0.8	N/A

**Table 4: Tools versions and parameters**

Our classification time results are measured in CPU seconds on a PC workstation with a 3 GHz CPU and 2 Gb

<sup>4</sup> SemWebCentral: <http://projects.semwebcentral.org/>

<sup>5</sup> available from <http://ssel.vub.ac.be/viewcvv/>



of RAM. After 300 CPU seconds, we terminate the classification task and consider it failed.

We also used Protégé<sup>6</sup> and Swoop<sup>7</sup> to browse the ontologies and check the classification results.

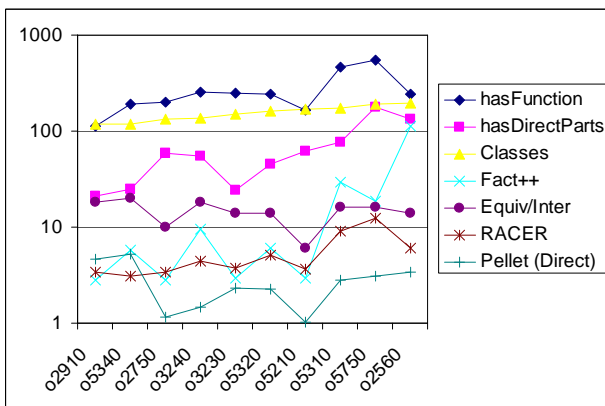
## 4 Results

We provide here three separate analyses. The first one will show that, on ontologies based on the Rector-Welty pattern, FaCT++ and RACER behave differently to Pellet with respect to the number of classes, the number of existential restrictions and the number of cycles. The second one illustrates the relatively superior tractability of CEL and its ability to scale for the full aircraft configuration. CEL is much faster than the other reasoners, but can only be used when it is possible to re-factor the ontologies to be  $\mathcal{EL}^+$  compatible. The third analysis focuses on the changes in reasoner performance.

### 4.1 Performance on Rector-Welty ontologies

A detailed comparative reasoner performance analysis over the full set of ontologies derived from the Service Difficulty Reports is not presented here in detail. Instead, we provide an illustrative sample of the results we have obtained, as shown below in **Figure 6** which illustrates this analysis for the top 10 four-digit ontologies. To get the top 10 four-digit ontologies, we have ordered the 253 four-digit ontologies (see 3.1 for the definition of one, two and four-digit ontologies) according to the three following indicators in decreasing order of significance: the number of classes and the number of existential restrictions for *isLocusOf* and *hasDirectPart*.

**Figure 6** uses a logarithmic scale to facilitate the comparison between the classification time (in seconds) and the numbers used to characterise the complexity of the test samples described in **Table 3**.



**Figure 6: Results for the top 10 four-digit Rector-Welty ontologies (117 to 196 classes)**

Our empirical finding is that to avoid reasoner timeouts, it is recommended to limit the generated ontology to use the Rector-Welty pattern less than 6 times, with less than 40

sub-classes involved in it, and to keep the overall size down to less than 150 classes and less than 300 to 400 existential restrictions tied to the same property. We believe that further cleaning and tuning of the generation process is possible to push these empirical limits upward so that we can work over larger subset of the aircraft domain at a time.

We believe that the performance of the three evaluated reasoners is tied to the number of classes involved in the pattern (Classes in **Figure 6**). We believe that the performance of Pellet varies more significantly with respect to this indicator and to the number of times the pattern is exploited; this value is constant for the test sample, so it is not shown in **Figure 6**. RACER and FaCT++ performance is also dependent on this indicator because all these classes are also flagged as cycles, which degrade the performance of these reasoners. In our test samples, FaCT++ and RACER are also penalised by the presence of large numbers of existential restrictions tied to small numbers of properties (*hasFunction* and *hasDirectParts* in **Figure 6**).

We have also noticed that FaCT++ performance is significantly degraded for ontologies containing cycles corresponding to modelling mistakes inherited from the original source: o2560 (on the far right in **Figure 6**) presents such a case with a classifying time over 100s.

### 4.2 Performance on re-factored ontologies

The re-factored ontologies correspond to a different generation method which suppresses the representation pattern N.4 from Rector & Welty and introduces a greater number of properties to manage relations between components and parts.

Rector & Welty (2005) note that “the inability of existing classifiers to cope with ontologies mixing *isPartOf* and *hasPart* is a significant limitation”. The re-factored ontologies are designed to check if this statement is valid for CEL or not: for each type of relations defined in the ontology, we have added the statements required to represent the inverse relation, but we have not used the *owl:inverseOf* construct for this case because CEL cannot handle it. With this approach, the measured number of classes involved in cyclic axioms signalled by RACER is almost equal to the total number of classes.

In this configuration, CEL is roughly 10 times better than Pellet, and Pellet is roughly 10 times better than RACER. One of the main finding of this experiment is that Pellet is less sensitive to the presence of cyclic TBox inclusion axioms than RACER and FaCT++. This also confirms our previous analysis on the sensitivity of RACER and FaCT++ to cycles. The full results are shown in **Figure 7** for the full set of 51 two-digit ontologies.

<sup>6</sup> Protégé <http://protege.stanford.edu>

<sup>7</sup> Swoop <http://www.mindswap.org/2004/SWOOP/>

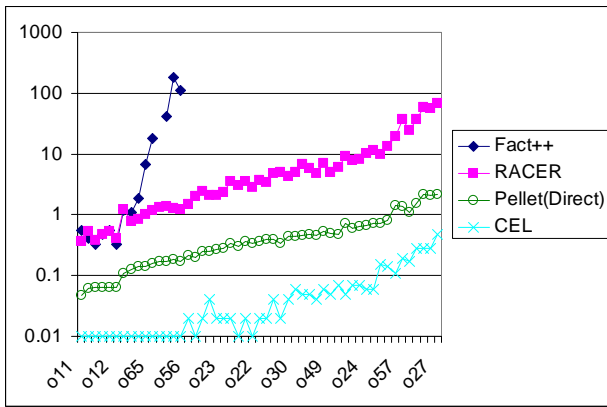


Figure 7: Results for the 51 two-digit re-factored ontologies (10 to 410 classes)

### 4.3 Performance on ontologies without cycles

To complete this analysis, we have modified the ontology generation instructions to create another set of *mono-directional* re-factored ontologies, simply by removing the statements corresponding to one of the inverse relation pairs (*isPartOf* and *hasPart*, *isFunctionOf* and *hasFunction*, etc.). This allows us to investigate the sensitivity of CEL to cyclic axioms and to check how the performance of FaCT++ and Pellet varies for ontologies without cycles.

For CEL, we can use the totality of the ontology. Table 5 provides the classification time in seconds, the number of classes and the number of cycles for the mono-directional and bi-directional re-factored ontologies comprising all ATA codes. The remaining cycle for the mono-directional re-factored ontologies (60 classes in total signalled by RACER) are inherited from modelling errors in the source data which are not suppressed by the generation process.

Ontology	CEL	Number of classes	Number of cycles
o w/o/ cycles	40.640	3957	60
o w/ cycles	50.530	3957	3687

Table 5: CEL results for the mono-directional and bi-directional re-factored ontologies

The classification time for the ontology “with cycles” (i.e. the ones including both the *is[...]* and the *has[...]* properties) is bigger than the result for the one “without cycles”. This increase is easily explained by the doubling in the number of existential restrictions in the second ontology and suggests that cycles do not pose a problem for performance.

For FaCT++, we focus on the four one-digit ontologies (o1, o4, o6, and o8) for which no cycles are signalled by RACER. Figure 8 shows the classification times for the Rector-Welty ontologies (FaCT++(1) and Pellet(1)) and for the mono-directional re-factored ontologies (FaCT++(4) and Pellet(4)).

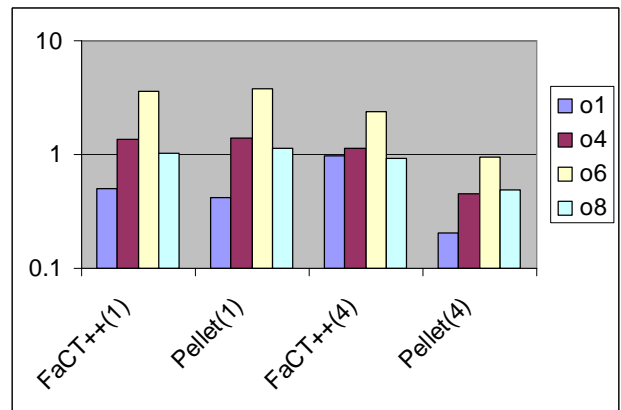


Figure 8: Results for 4 one-digit R-W ontologies and mono-dir. re-factored ontologies (43 to 327 classes)

Figure 8 shows that the performance figures for FaCT++ and Pellet are a lot closer to what has been observed previously. For the Rector-Welty ontologies, the minor difference between FaCT++(1) and Pellet(1) is much smaller than the gap visible in Figure 6. The slightly larger difference between FaCT++(4) and Pellet(4) contrasts with the large gap visible in Figure 7.

### 4.4 Other lessons learned

We summarise here the lessons learned during this experiment with respect to how to get the most out of the DL reasoners.

*Avoid duplicate superclass declarations.*

With our method, it is not always easy to prevent the occurrence of duplicates, especially those linked to existential restrictions which we do not handle well. The capability to remove all types of duplicates, now available in Protégé 3.2, should be added to our generation environment. To give an idea of the size of the problem, more than 7 hours have been necessary for Protégé to remove the duplicate definitions out of the largest ontology of this experiment, with a resulting ontology one tenth of the size. Further analysis is required to understand how the presence of duplicates may influence reasoner performance in their various execution modes. This gain in simplicity is now available for the users of Protégé thanks to the *duplicate superclasses post processor* developed by H. Knublauch<sup>8</sup>.

*Beware of oversized content and of DIG server behaviour.*

Both execution modes for Pellet have been used for evaluation to compare the direct execution of the reasoning core to the one triggered through the DIG interface. Our results show a significant gap between the two modes and memory errors that are more likely to occur through the DIG interface. FaCT++ also has trouble with oversized input because it does not manage interrupted sessions adequately. RACER is probably the

<sup>8</sup> DuplicateSuperclassesPostProcessor.java is accessible through the Protégé source code repository hosted at <http://smi-protege.stanford.edu/> Accessed 25 Aug. 2006.

most robust and stable DIG implementation we have used. The DIG capability for CEL has not yet been evaluated.

## 5 Discussion

### 5.1 Scalable ontology engineering guidelines

It is challenging to develop a best-practice design style which can be successfully applied for the generation of very large ontologies. To reach this objective, we need more precise guidelines and modularisation approaches which acknowledge the practical limitations of reasoners.

There is a lack of reference to *scalable modelling style* in the currently available guidelines from W3C listed in **Table 2**. **Table 6** proposes four categories to better scope the presently available advice with respect to the orientation of the users towards scalable TBox reasoning or ABox reasoning and to their reasoner preferences. Further refinement of these categories is likely to be needed once the work on new tractable fragments to extend OWL from the OWL community (2006) is finalised.

Scalable modelling style	TBox / ABox	Reasoner
SNOMED-like ontologies	TBox	CEL (FaCT++)
GALEN-like ontologies	TBox	FaCT++, (RACER), (Pellet)
DOLCE and Wine-like ontologies	TBox / ABox	RACER, Pellet, (KAON2)
LUBM-like ontologies	ABox	(RACER), (Pellet), KAON2

**Table 6: Benchmark categories and reasoners**

End users are likely to be annoyed by classification times greater than 5 minutes for non repetitive tasks and 30 seconds or less for repetitive tasks. This is why the present OEP guidelines also need to be augmented with more empirical advice on the number of times a specific pattern can be used and the number of classes and existential restrictions which can be planned so that the generated ontologies do not lead to excessive reasoning delays. In the case of the Rector-Welty pattern, an important requirement for potential users is to know how many times it can be applied recursively. Further work is required because in the results presented here, the Rector-Welty pattern is acted upon over the ATA/JASC code hierarchy and not over the part-whole hierarchy inherited from the input.

### 5.2 The Rector-Welty guideline and CEL

We have discussed above that CEL cannot handle the *unionOf* construct present in the Rector-Welty pattern. For this case, an alternative approach should be defined.

CEL does support a particular non-OWL feature called role inclusion, a form of role composition that supports declarations of transitive roles (also in OWL) and right identity (not in OWL). Right identity is likely to be important for applications of aircraft configuration ontologies: indeed it obviates the Rector-Welty pattern for explicit transmission of faults up the *isPartOf* hierarchy. We illustrate by example, following the syntax and style of the medical example of Horrocks and Sattler (2003).

We assert the right-identity axiom,

$$hasLocus \circ isPartOf \leq hasLocus$$

to mean that something that is located in a part P is also located in the places that P is a part of. This means that an expression declaring a fault of the crankcase of the aircraft engine:

$$Fault \cap \exists hasLocus \bullet (Crankcase \cap \exists isPartOf \bullet Engine)$$

is inferred to be a fault of the engine itself. That is, the following can be inferred to hold, from the previous two declarations:

$$Fault \cap \exists hasLocus \bullet Engine$$

Furthermore, assuming the *isPartOf* role is declared to be transitive and the appropriate part and component taxonomy is defined, a fault located in the aircraft as a whole is also inferred. Presently, we can apply the transitive *isPartOf* structure in the ontologies we have generated (see section 6), but not the right-identity feature because it is not (yet) handled by tools based on the present versions of OWL and on DIG.

### 5.3 Scalability and modularisation

The development of modularisation approaches to managing the scalability of ontologies is in a very early stage; and is typically applied from the point of view of the need for independent development of modules by knowledge engineers (Rector & Pan 2005).

There is certainly an opportunity to consider what can be done in a semi-automated environment such as that which is offered by XO, where a combination of techniques related to modularisation and restrictions to the language may be achievable.

### 5.4 Handling cyclic axioms through DIG

The working environment we have defined allows us to retrieve the cyclic axiom warnings reported by RACER. We have also manually monitored a similar type of warning as reported by FaCT++. We would be interested to get this type of information more seamlessly. A possible approach would be to extend the DIG interface with specific services providing the number of cycles and the identity of the classes involved with as much detail as



available. This would help users of tools such as Protégé to better understand how the reasoner performance is affected by the choice of the ontology engineering pattern and also to help them to fix possible modelling errors.

## 6 Conclusion

Reasoner performance over large ontologies with large TBox components is highly variable and dependent on the choice of the ontology engineering patterns. Starting from the published advice on part-whole ontologies has been critical for this experiment, guiding us towards a much needed simplification of the generated ontologies.

The main contribution of this paper is our experimental analysis on the representation pattern recommended for the propagation of properties along the part-whole relation. Our experience shows that this pattern can be used on sub-modules of a size of less than around 150 to 200 classes.

We have also confirmed that avoiding or removing inadvertent cyclic axioms is critical for the performance of some reasoners, especially for FaCT++. This reinforces the W3C advice to avoid combining *isPartOf* relations and their inverses for part-whole ontologies.

CEL is less sensitive to terminological cycles, and we are confident it can be used to handle an ontology corresponding to the whole aircraft configuration with 4000 classes or more such as the one we have created for this experiment.

We plan to continue this work in two directions. Our first priority is to expand the results of this study for the other ontology engineering patterns authored by the OEP task force listed in **Table 2**. Our second priority is to investigate the scalability and modularity challenges presented by the large medical ontology, SNOMED. Further work is required to validate and evaluate the adaptation proposed to the present W3C part-whole recommendation to match CEL capabilities.

Finally, this study illustrates the effectiveness of modularisation approaches combined with the selection of the appropriate modelling style to support the generation of large ontologies out of existing resources. More support from the existing tools is required to better align the published guidelines to the specificities of each reasoner and to help the user to eliminate the present causes of problems such as cycles or duplicate superclass definitions which are more likely to occur in this context.

## Acknowledgements

The authors gratefully acknowledge the support of Boeing in this work, particularly of Robb Graham, Steve Uczekaj and Craig Battles.

Many thanks also to John Colton of CSIRO.

## 7 References

- Baader, F. (2003): *Terminological Cycles in a Description Logic with Existential Restrictions*. LTCS-Report 02-02, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2002.
- Baader, F., Brandt, S., and C. Lutz, C. (2005): Pushing the EL Envelope, In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, Edinburgh, UK, Morgan-Kaufmann.
- Baader, F., Lutz, C. and Suntisrivaraporn, B. (2006): CEL-A Polynomial-time Reasoner for Life Science Ontologies (System Description). In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, Seattle, WA, USA, Lecture Notes in Artificial Intelligence 4130. Springer, pp. 287–291.
- Dameron, O., Rubin, D., and Musen, M. (2005): Challenges in converting frame-based ontology into OWL: the Foundational Model of Anatomy case-study. In *Proceedings of the American Medical Informatics Association Annual Symposium (AMIA05)*, Washington DC, pp. 181-185.
- FAA (2002): *Joint Aircraft System/Component Code* Federal Aviation Administration, Flight Data Services [http://av-info.faa.gov/isdr/documents/JASC\\_Code.pdf](http://av-info.faa.gov/isdr/documents/JASC_Code.pdf) Accessed 25 Aug 2006.
- FAA (2005): *FAA Service Difficulty Reporting Service* <http://av-info.faa.gov/isdr/> Accessed 25 Aug. 2006.
- Gardiner, T., Horrocks, I., and Tsarkov, D. (2006): Automated benchmarking of description logic reasoners. In *Proceedings of the 2006 Description Logic Workshop (DL 2006)*. Windermere, Lake District, UK, Parsia, B., Sattler, U. and Toman, D. Eds. CEUR Workshop Proceedings 189, CEUR-WS.org
- Guo, Y.; Heflin, J; and Pan, Z. (2003): Benchmarking DAML+OIL Repositories In *The Semantic Web - Proceedings of the Second International Semantic Web Conference (ISWC 2003)*, Sanibel Island, FL, USA, Springer, pp, 613-627.
- Guo, Y., Pan, Z., and J. Heflin, J., (2004): An Evaluation of Knowledge Base Systems for Large OWL Datasets. In *Proceedings of Third International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan, LNCS 3298, Springer, 2004, pp. 274-288.
- Haarslev, V., Möller, R. and Wessel, M. (2004): Querying the Semantic Web with Racer + nRQL In *Proceedings of the Workshop on Description Logics 2004 (ADL 2004)*, Ulm, Germany, 2004
- Haarslev, V., Möller, R. and Wessel, M. (2005): Description Logic Inference Technology: Lessons Learned in the Trenches in *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, Edinburgh, Scotland, UK, July 26-28, 2005,
- Horrocks, I. and Sattler, U. (2003): Decidability of SHIQ with complex role inclusion axioms. *Artificial Intelligence*, Volume 160, Issues 1-2:79-104,
- KnowledgeWeb (2005): *Success Stories and Best Practices, KnowledgeWeb Deliverable 1.4.2* Lancieri,

- L., Maynard, D. and Gandon, F. Eds. KWEB/2004/D1.4.2/v3 KnowledgeWeb consortium
- Lefort L. and Taylor, K. (2005): Large scale colour ontology generation with XO In *Proceedings of the Australasian Ontology Workshop (AOW 2005)*, Conferences in Research and Practice in Information Technology (CRPIT), Vol. 58. Meyer T. and, Orgun, M. Eds. Sydney, Australia.
- Motik, B., Volz, R. and Maedche, A. (2002): Optimizing query answering in description logics using disjunctive deductive databases. In *Proceedings of the 10th International Workshop on Knowledge Representation Meets Databases (KRDB-2003)*, Hamburg, Germany, pp 39–50.
- Motik, B. (2006): *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, University of Karlsruhe, Germany.
- Nebel, B. (1991): Terminological cycles: Semantics and computational properties. In *Principles of Semantic Networks*, Sowa, J., F. Ed., Morgan Kaufmann, Los Altos, pp 331-361.
- OWL community (2006): *Tractable Fragments of the OWL 1.1 Extension to the W3C OWL Web Ontology Language*, Editor's Draft of 14 June 2006, Cuenca Grau, B. Ed., University of Manchester <http://owl1-1.cs.manchester.ac.uk/Tractable.html> Accessed 25 Aug. 2006.
- Patel-Schneider, P., Hayes, P. and Horrocks, I. (2004): OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation 10 February 2004 <http://www.w3.org/TR/owl-semantics/> Accessed 25 Aug. 2006
- Rector, A. (2003): Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In *Proceedings of the 2nd international Conference on Knowledge Capture (K-CAP '03)* Sanibel Island, FL, USA, ACM Press, New York, NY, pp 121-128.
- Rector, A. (2005): *Ontology Normalisation, Pre- and Post- Coordination* Advanced Ontology Tutorial <http://www.cs.man.ac.uk/~rektor/tutorials/feb/Presentations/Normalisation.ppt> Accessed 25 Aug. 2006.
- Rector, A. and Pan, J. (2005): Chapter 12 Engineering Robust Modules In *Report on Modularization of Ontologies*, Spaccapietro S. Ed. KnowledgeWeb Deliverable D2.1.3.1, KWEB/2004/D2.1.3.1/v1.1 KnowledgeWeb consortium
- Rector, A. and Welty, C. (2005): *Simple part-whole relations in OWL Ontologies*, W3C Editor's Draft 11 Aug 2005, Version 1.5, <http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/index.html> Accessed 25 Aug. 2006.
- Schreiber G. (2006): *Ontology patterns (Lecture notes)*, Knowledge modeling course, Dutch Research School for Information and Knowledge Systems, Vught, Netherlands <http://hcs.science.uva.nl/SIKS/> Accessed 25 Aug. 2006.
- Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A. and Katz, Y. (2005): Pellet: A Practical OWL-DL reasoner. *UMIACS Technical Report 2005-68*. Submitted for Publication to Journal of Web Semantics
- Sirin, E., Cuenca Grau, B. and Parsia, B. (2006): From wine to water: Optimizing description logic reasoning for nominals. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Tenth International Conference (KR2006)*, Lake district, UK. Doherty, P. Mylopoulos, J. and Welty, C. Eds, AAAI Press, pp 90-99.
- Tsarkov, D. and Horrocks, I. (2005a): Ordering heuristics for description logic reasoning. In *Proceedings of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pp. 609-614.
- Tsarkov, D. and Horrocks, I. (2005b): Optimised classification for taxonomic knowledge bases. In *Proceedings of the 2005 Description Logic Workshop (DL 2005)*, Edinburgh, Scotland, UK, Ian Horrocks, I. Sattler, U. and Wolter, F. Eds, CEUR Workshop Proceedings 147, CEUR-WS.org
- Zolin, E. (2006): Complexity of reasoning in description logics <http://www.cs.man.ac.uk/~ezolin/logic/complexity.htm> Accessed 25 Aug 2006

# Approaches for Semantic Interoperability between Domain Ontologies

**B. Orgun, M. Dras, A. Nayak**

Intelligent Systems Group, Department of Computing  
Macquarie University - Sydney NSW 2109 Australia

{borgun, madras, anayak}@comp.mq.edu.au

**G. James**

CSIRO ICT Centre

Building E6B, Macquarie University Campus  
Herring Road, North Ryde NSW 2113, Australia

Geoff.James@csiro.au

## Abstract

Domain ontologies and knowledge-based systems have become very important in the agent and semantic web communities. As their use has increased, providing means of resolving semantic differences has also become very important. In this paper we survey the approaches that have been proposed for providing interoperability among domain ontologies. We also discuss some key issues that still need to be addressed if we were to move from semi to fully automated approaches to provide consensus among heterogeneous ontologies.

*Keywords:* Heterogeneous ontologies, semantic interoperability, agent-based systems.

## 1 Introduction

Ontologies have become increasingly popular in a broad range of applications and have moved away from academic knowledge representation projects to the commercial world. Early commercial ontologies were primarily used as navigation aids as in the case of Yahoo or Lycos and were simple taxonomies of class names. We now see more complicated ontologies that are being used in diverse applications ranging from smart search engines [1] to intelligent agent reasoning. The increased effort in ontology development has also brought about a greater need for standardisation in order to provide highly reusable, extensible structures that are viable over long periods of time. Parallel to the need for standardising, is the need for providing interoperability among multiple, distributed ontologies. With the growing trend in ontology based multi-agent systems this need has become imperative.

Providing interoperability among heterogeneous ontologies can be broadly categorised into merging, aligning and integrating, and for the purpose of the study

we have used these terms interchangeably. Merging ontologies provides a single coherent ontology that includes information across all the sources. Aligning ontologies is the preferred approach when sources must be separately kept consistent and coherent with one another. Consensus ontologies are developed through ontological alignment where the source ontologies are physically separate but are hybrid forms of each other. Integrating ontologies involves building new ontologies by assembling, extending, specialising or adapting other existing ontologies.

This paper discusses the various approaches to providing access to multiple heterogeneous ontologies. The paper is organised as follows: in section 2 we describe commonly used terms and explain their relevance in understanding the paper. Section 3 covers approaches for merging, aligning and integrating ontologies. In section 4 we discuss other work related to providing interoperability across heterogeneous data sources, followed by a discussion in section 5 and we conclude in section 6.

## 2 Commonly used terms and their significance

Domain ontologies represent abstract models of how people think about concepts related to a particular area. In this paper, we assume that ontology typically consists of concepts or terms related to the domain arranged in a hierarchical fashion. The concepts have attributes associated with them defining their properties as well as their relationships with other concepts. Concepts have instances that capture their values. Whenever terms different from those mentioned above are used in this paper, we provide details as applicable.

One of the key problems in the development of consensus ontologies has been automating syntactic and semantic interoperability. Syntactic translation is easier to automate and in [3] Dou et al. claim that semantic translation cannot be fully automated. Syntactic interoperability depends on determining equivalence of two terms and only guarantees that two concepts each have a structure with apparently similar attributes and values but represented using different syntax. Syntactic processing involves providing syntactic interoperability across heterogeneous ontologies. Some approaches have defined their own notions of syntactic interoperability

which is different from ours and we will address these individually as we discuss the various approaches.

Semantic interoperability on the other hand, provides means to address the heterogeneity gap between ontologies by identifying related concepts. Two data types are semantically equivalent if they have the same defining attributes and operations and the same set of constraints on those attributes and operations. Traditionally semantic mapping was undertaken by human domain experts and only recently have approaches been developed to automate this process. The resolution of semantic interoperability is achieved through semantic processing. We are more interested in the current approaches to provide semantic interoperability.

### 3 Approaches for ontological interoperability

The key areas fuelling much of the ontological interoperability research have been the semantic web effort and multi-agent development. The Semantic Web relies heavily on the formal ontologies that structure underlying data for the purpose of comprehensive and transportable machine understanding [5]. Information mapping across different ontologies requires knowledge of their semantic mappings. The use of ontologies in multi-agent systems has also seen an increase in the last few years. Some important research topics here are: defining the semantics of agent communication primitives, dealing with different vocabularies, and specifying and verifying interaction protocols. Recently agent-based approaches have also been used to serve as *mediators* between multiple ontologies, supporting cross system agent communication. In multi-agent systems, languages such as ACL [6] and KQML [7] provide the standard for agent communication. Since the ontologies used in the communication are not standard, ontology negotiation is crucial to enable cooperation among agents that are based on different ontologies.

In this section we present a (fairly comprehensive) survey of the various approaches based on their underlying technology i.e. agent or non-agent, the degree of automation and the use of intermediaries such as meta-ontologies or lexicons. Information regarding ontology language, whether the approach used tree or graph structures, had instances related to concepts, provided performance evaluation, and provided conflict resolution, has been discussed when such information was accessible. Deviations from our understanding of ontology, syntactic and semantic interoperability as discussed in section 2 are also identified as are any known limitations.

#### 3.1 Non Agent-based semi-automatic approaches

In this section we discuss some non agent-based, semi-automatic approaches that have been used to merge, map or align different ontologies. The methods used range from machine learning to simple heuristics in determining syntactic and semantic equivalence of concepts.

#### 3.1.1 ITTalks

ITTalks [8] [9] is a web-based system for automatic and intelligent notification of information technology talks. The ITTalks project was a case study in the application of the DARPA Agent Markup Language (DAML) [10] / DAML plus Ontology Inference Layer (DAML + OIL) and agent interaction. DAML was developed to provide a semantic mark-up language that provides sufficient rules for ontology development and support intelligent agents and other applications. Although ITTalks supports agent interaction, the mapping resolution process is not agent-based. Prasad et al. [11] were responsible for developing the semi-automated ontology mapping sub system and we describe their effort below.

The experiments for ITTalks were conducted using the ACM Topic ontology and a small ITTopic ontology that organised classes of IT related talks differently from the ACM classification. The ontologies were treated as tree structures with nodes representing concepts. Sets of *exemplars* (or URL's pointing to locations of text based abstracts of documents that belong to this concept) were attached to each concept in the two ontologies. The Rainbow text classifier [12] was used to build models for all ontologies. The model has statistical information about the exemplars of each concept and is represented as a similarity matrix. Each concept in the first ontology is mapped to one or more concepts of the second ontology by comparing their exemplars and using the Rainbow text classifier. The classifier returns raw similarity scores that are used to produce a set of possible mappings by the automated mapper. The user then specifies a set of *landmark mappings* between concepts from the two ontologies and then can select from either the heuristic or the Bayesian approaches for classification undertaken by the automated mapper. Experts evaluated the mapping results, and found mappings generated by the Bayesian approach to be better than the simple heuristic. No conflict resolution mechanism was provided.

It is claimed that the use of greater number of exemplars and exemplars that contained full-length papers rather than just abstracts would help improve accuracy. It was noted that a classifier other than Rainbow would produce different results. This approach deals with semantic interoperability only. Since both the ontologies being mapped are in DAML or DAML + OIL format the need for syntactic interoperability does not arise. The human evaluation of the landmark mappings had potential for possible misclassification. The assumption of mutual exclusivity made for the Bayesian approach might not hold true for all leaf nodes and could result in error. Attributes associated with concepts were not utilised in the mapping process and were left for future work.

#### 3.1.2 Ontoprise - OntoEdit

Developed for enabling easy ontology development for the semantic web community, the Ontoprise [13] suite of products also consists of *OntoEdit* and *Text-To-Onto*. Maedche et al. [5] described the underlying ontology learning framework for OntoEdit. While the work focuses primarily on facilitating the construction of ontologies, it

also incorporates the various phases of importing (and reusing for merger), extracting, refining and evaluating data from various sources including free texts, web crawlers, legacy databases and existing ontologies.

The import and reuse phase draws upon various components: a generic management component, a resource processing component, an algorithm library and a graphical user interface for ontology engineering. The management component allows the user to select the sources of data to be processed and processing methods from the resource-processing component and relevant algorithms. The resource-processing module transforms the source data into the format required by the algorithm that would process it further. In this phase the syntactic processing of source data occurs over varied data sources and source ontology languages. The import sources are identified and their general content evaluated by domain experts prior to import.

The merging of the imported conceptual structures is a bottom-up approach that is also influenced by the source data. The details of their approach for import and reuse are not clearly outlined except to say that it depends largely on the input data. Following the data acquisition and merger, the ontology extraction phase is implemented iteratively in the ontology learning environment *Text-To-Onto* and exploit various type of web resources. The techniques used in this phase include *lexical entry and concept extraction*, *hierarchical concept clustering*, *dictionary parsing* and *data-mining based association rules*. At the end of this phase the user is provided suggestions for changes that may or may not be incorporated into the ontology. The next phase is the pruning phase, where the pruning mechanism is based on relative counts of frequency of terms. The proposed approach is semi-automatic and the techniques used in the import, reuse and extraction phases greatly vary with input data. The phases from merger to pruning and refinement are responsible for resolving semantic interoperability between the various data sources using various techniques, but the application of these techniques is decided by the expert user in an ad hoc manner. The ontology extracted in *OntoEdit* can be exported in DAML-ONT, OIL as well as in F-Logic [14] based extension of RDFS [15]. Constraint checking is provided through other plugins to *OntoEdit*. The ontology within *OntoEdit* is represented as a graph structure corresponding closely to RDFS, with concepts arranged in a *hierarchy* (a taxonomic structure that supports multiple inheritances). The relations between concepts can also be arranged in a heterarchy. Both concepts and relations have instances associated with them. Details on any evaluation criteria used are not provided.

### 3.1.3 OntoMerge (Ontology translation by Merging ontologies)

OntoMerge is an online tool that provides a semi-automated nexus [16] for combining notational differences between ontologies with overlapping subject areas. The OntoMerge project is part of the DAML [10] program. Source and target ontologies represented in DAML [17] or DAML + OIL [3] are automatically

converted into a uniform internal representation, Web - Planning Domain Definition Language (Web-PDDL) [18]. After conversion into Web-PDDL the syntactic differences related to organization of the concepts are resolved and thus its use as the intermediary format helps separate the syntactic and semantic operations. It is important to note that the syntactic operations provide syntactic interoperability as mentioned in section 2 by providing a standard syntax for representing the ontology's structure. Web-PDDL uses a Lisp [19] like syntax.

The Web-PDDL based ontology has a structure that is different from the once described in section 2. A Web-PDDL ontology contains *vocabulary terms* derived from instances of concepts that existed in the DAML ontology, *term types* or concept names, *axioms* that define the relationships between instances of concepts, *predicates* or attributes associated with concepts and set of *facts* derived from instances of concept attributes that existed in the DAML ontology. Human experts construct an intermediary merged ontology containing the union of the concepts and the axioms that form part of the source and target ontologies. Experts add *bridging axioms* in the merged ontology to relate terms in two ontologies.

The key features of OntoMerge are the use of an intermediary language to resolve notational syntactic differences, the use of intermediary merged ontology built by domain experts; and the automated reasoning engine for translation. Details on conflict resolution and performance evaluation are not provided.

### 3.1.4 Prompt & Anchor Prompt

Prompt [20] previously known as SMART [21] is a product of the Stanford Medical Informatics (SMI) Lab and is available as a plug in for Protégé-2000 [2] the ontology editor by the same lab. The knowledge model is frame based and Open Knowledge Base Connectivity (OKBC) [22] compatible. Since it is used in conjunction with Protégé-2000, the input and output ontologies can be in a wide variety of ontology languages. The conversion facility in Protégé-2000 can be used to convert source ontologies to a consistent language format thus providing the syntactic conversion as mentioned in section 2. The ontology is represented as a hierarchy of concepts.

The Prompt algorithm provides semi-automatic merging and alignment of ontologies. Initial comparisons based on syntactic and semantic matching of the content and structure of the source ontologies are used to automatically create a list of matches with which the user is prompted. The syntactic matching being referred to here is based on two concepts or attributes having identical names and is different from our notion of syntactic interoperability as mentioned in section 2. Semantic matching utilises linguistic techniques details of which are not provided.

Ontology merging operations include merging concepts, merging attributes, merging bindings between concepts and attributes, and deep and shallow copying of concepts. Instances of concepts are not utilised in mapping resolution. The types of conflicts that are identified

include name conflicts (more than one concept or attribute with the same name), dangling references (a concept or attribute refers to another concept or attribute that does not exist), redundancy in concept hierarchy (more than one path from a class to a parent other than the root) and attribute value restrictions that violate concept inheritance. The decision to merge (create a new ontology from multiple source ontologies) or align (add the suggested changes to each one of the constituent ontologies while keeping them as separate ontologies) is made by the end user. In the tests performed as part of Prompt's evaluation [20], human experts followed 75% of the conflict resolution and 74% of the merging suggestions and Prompt had 30% more correct suggestions than Chimaera [23].

Anchor-Prompt [24] is also a product of the Stanford Medical Informatics (SMI) Lab and is OKBC [22] compatible. It augments the earlier Prompt [20] algorithm which matches terms from two different ontologies based on names of classes and slots, subclasses, superclasses, domains and ranges of slot values and the actions performed by the user. The main difference between Prompt and Anchor Prompt is that in the latter *anchors* (or related concepts) are used to establish a link between common terms in the source ontologies. The user can input the set of anchors or these terms can be automatically identified through *lexical matching* (based on the use of a lexicon or morphemes to identify similar words). Anchor-Prompt is also a plug-in to Protégé-2000 and can import and export ontologies in a wide variety of ontology languages. Similar to Prompt [20] the conversion facility in Protégé-2000 can also be used to convert source ontologies to a consistent language format thus providing the syntactic conversion as in Prompt.

In the evaluation tests on merging ontologies developed independently by different group of researchers, 75% of the results produced by Anchor-Prompt were deemed correct by experts. The Anchor-Prompt algorithm produces good results only if ontology developers link the concepts in a similar fashion even though different names are assigned to them. Information related to conflict resolution is not provided.

### 3.1.5 Chimaera

Chimaera [23] is an interactive web based merging and diagnostic tool based on the Ontolingua ontology editor. It was developed by the Stanford University Knowledge Systems Laboratory (KSL) and follows the OKBC [22] standard. Chimaera takes knowledge base source files as its input prior to merging them into a new or existing knowledge base. The source files can be in a wide variety of different source languages thus providing syntactic interoperability (see section 2). In Chimaera ontology is represented by concepts and attributes arranged in a hierarchy. Child concepts of a concept can be disjoint from one or more sub concepts or exhaustively cover all the sub concepts of the concept in question. Instances related to concepts and attributes are utilised in the merging process. In contrast to Prompt, the Chimaera environment supports the creation and editing of disjoint partition information, allows bringing together of

ontologies built using different formalisms like Knowledge Interchange Format [25] and OKBC. Ontologies can also be created and edited in OKBC compliant Ontosaurus [26] and Ontoweb [27] before being loaded into Chimaera. Possible merger candidates are based on the outcomes of two tasks: (1) the automatic identification of semantically identical terms from separate ontologies that can be coalesced to provide a new name for the newly merged concept and (2) the automatic identification of subsumption, disjunction and instance relationships.

Chimaera only addresses the merging of child concepts, parent concepts and attributes of concepts. The merging and evaluation consists of a *name resolution list* generation and *taxonomy resolution list* generation. The name resolution list generation suggests candidates from the two ontologies that can be merged based on concept or slot names. The taxonomic resolution suggests taxonomy areas that are candidates for reorganisation. Performance evaluation found Chimaera to be 3.46 times faster than Ontolingua for merging substantial taxonomies, and 14 times faster in name resolution.

### 3.1.6 GLUE

The GLUE [30] project utilises machine learning to find semantic mappings across heterogeneous ontologies in the semantic web. GLUE uses multiple learning techniques thus exploiting different types of information from the taxonomic structure or data instances in the ontologies. The underlying notion of semantic similarity in GLUE is based on the joint probability distribution of the concepts involved using *Jaccard's coefficient* [31] and depends on the semantic content of the concepts and not their syntactic specification. GLUE also incorporates common-sense knowledge, domain constraints and general heuristics in the mapping process using a technique borrowed from the field of computer-vision called *relaxation labelling*. Relaxation labelling is a constraint optimisation technique and is used to exploit different types of information either in the data instances or the taxonomic structure of the ontologies represented as tree structures. In GLUE, ontology consists of concepts, attributes, instances of concepts and relations modelled into a taxonomic tree. The concept instances are also used in the mapping process. Details on the ontology languages as well as how this approach caters for syntactic interoperability are not provided.

Their experiments resulted in 60 – 97 % accuracy in matching concepts from real-world domains. Limitations include processing sophisticated mappings (i.e., non 1-1 mappings), exploiting constraints related to concept attributes and their relationships.

## 3.2 Non Agent meta-ontology based semi-automatic approaches

The use of a meta-ontology for providing ontological interoperability has its analogue in the use of inter-lingua for machine translation in natural language based systems. This section covers some semi-automatic approaches that use this technique.



### 3.2.1 Ontolingua

The Ontolingua [33] ontology development environment provides a suite of ontology authoring tools and a library of modular reusable ontologies. The tools in Ontolingua are oriented towards the authoring of ontologies by assembling and extending ontologies obtained from the library that contains a number of knowledge representation ontologies, common-sense ontologies, upper-level ontologies, generic ontologies that could be reusable across domains, domain dependent ontologies, etc. As is common in many efforts to build global ontologies, such as Cyc [34], SUMO [35], Penman upper model [36] and SHOE [37], Ontolingua ontologies form a lattice with the most general purpose ontologies at the top of the hierarchy and the more specific ones at the bottom. The Ontolingua server provides tools for integration or translation of two ontologies using the upper level general-purpose ontology serving as an inter-lingua to mediate the translation. The main drawback of this approach is that there is no all-encompassing global ontology available to date. Achieving consensus among ontology experts to provide translators between their ontology and this global ontology and maintaining consistency among all the ontologies also make this approach impractical. Ontolingua is OKBC [22] compliant and supports querying in Java, C, Lisp and Knowledge Interchange Format or KIF [25].

### 3.2.2 CYC

The use of ontologies to enable heterogeneous agent interaction draws upon systematic, multi-domain modelling, with higher-level models providing the inter-relations. Cyc [34], a large commonsense knowledge base, is one of the first attempts at providing a near-universal model. Over the last 15 years several ontologies including SENSUS [26], MeSH [38], Snomed [39], UMLS [40], large portions of WordNet [41], have been mapped or integrated with Cyc [42]. Domain experts using Cyc's interactive-clarification-dialog based tools achieved the mapping between different source ontologies and Cyc. Majority of the work involved mapping terms and the process varied with the ontology being merged into Cyc. Other issues dealt with included resolving simple and complex structural differences as well as resolving fundamentally different representations. The work was semi-automatic and required domain expert level of knowledge for validation.

### 3.3 Non-agent based automatic approaches using lexical tools

The use of lexical tools for providing ontological interoperability in non-agent based systems has been used frequently in natural language processing. This section covers an automatic approach that uses this technique.

#### 3.3.1 ODEMerge and WebODE

ODEMerge [43] [44] is a web-based client-server tool to merge ontologies developed by the Ontology Group at Technical University of Madrid. The approach for merging ontologies [45] involves (a) syntactic

transformation of formats of the ontologies to be merged, (b) evaluation of the ontologies using rules, (c) merging of the ontologies, (d) evaluation of the result and (e) transformation of the format of the resulting ontology to be adapted to the application where it will be used. WebODE helps in steps (a), (b), (d) and (e) of the merging methodology, and ODEMerge carries out the merge of taxonomies of concepts in step (c) and caters for attributes and relation level merging, and it incorporates many of the rules identified in the methodology. The inputs to ODEMerge include the source ontologies to be merged, a *table of synonyms*, which contains the synonymy relationships of the terms of the two ontologies and a *table of hyperonyms*, which contains the hyperonymy relationships of the terms of the two ontologies. The output of ODEMerge is a newly merged ontology. The tool merges the source ontologies using the synonymy and hyperonymy tables. ODEMerge can be used to merge ontologies in different ontology implementation languages such as XML, RDF [15] or CARIN [46], and allows exporting into XML, RDF [15], DAML+OIL, CARIN [46], F-Logic [14], Prolog, Jess [47], Java and HTML. Since explicit synonym and hyperonym tables have to be provided, their contents greatly influence and thus bias the end results.

### 3.4 Agent-based automatic approaches using a lexical tool

Multi-agent systems require the ability to communicate with other multi-agent systems that were built using different ontologies. In this section we cover fully automatic agent-based approaches that use a lexicon to facilitate the mapping of concepts between the ontologies.

#### 3.4.1 Ontology Negotiation Protocol (ONP)

Bailin et al. [48] [49] proposed an ontology negotiation protocol that allows agents to discover ontology conflicts and through an incremental process of *interpretation*, *clarification*, *relevance evaluation* and *ontology evolution*, establish a common basis for communicating with each other. Their approach aims at automating the agent dialogue that leads to resolving ontology conflicts by providing agents with a common language in which to converse. While the proposed new message types were implemented as KQML [7] primitives, the authors claim that they can be easily adapted to the FIPA – ACL [6]. The ontology negotiation protocol is state-based with a predefined set of available operations. Messages exchanged between agents are sequences of keywords describing the documents that are queried or found. In the interpretation phase, message received is analysed to see if its constituent keywords exist in the local ontology, the WordNet [41] lexical database is also used at this stage to find synonyms of the keywords. The clarification phase involves querying the source ontology to retrieve further information about any keywords that were not resolved during the interpretation phase. The results of a query are evaluated against the original query during relevance analysis. A relevance measure for query results is accumulated using the criteria specified as *evidence of relevance* that includes tests for connectedness and

specialisation. ONP culminates in either one or both agents modifying their ontologies thus evolving their local ontologies. In the ONP API hierarchical information contained in the ontology is in the form of term trees. Terms are analogous to ontological concepts as specified in section 2. Agents should be able to convert their internal ontological representation into these term trees. ONP allows heterogeneous agents developed within different frameworks to communicate. However, in the ONP approach resolving inconsistencies that may arise from clarification of concepts by different agents have been not dealt with effectively and are currently treated by accommodating alternative interpretations of a term, i.e. treated as synonyms of a term rather than as separate concepts. No syntactic operations were undertaken and details on performance evaluation are also not provided.

### 3.4.2 Consensus Ontologies for Web Services

This approach proposed by Williams et al. [4] is based in the B2B electronic commerce domain of web services. When an agent is searching for a web service, its ontology of the web service may not match the ontologies of other agents listing their web services. Agents need to form local consensus ontologies in order to find matches between their needed services and those being provided by another agent. This approach involves agents autonomously merging ontologies to form relatively small local ontologies between the agents they wish to communicate and query. The experimentation was done on 26 ontologies created for experiments in [50]. All these ontologies were developed in DAML [10]. Similarity identification of concepts between ontologies was based on determining syntactic equivalence using *edit distance* [51] and semantic equivalence using WordNet [41]. It is important to note here that the syntactic equivalence mentioned here is not the same as semantic interoperability mentioned in section 2 and deals primarily with determining equivalence based on the names of concepts, attributes and relations.

### 3.5 Agent-based automatic approaches using a meta-ontology

In this section we discuss the use of a meta-ontology for providing ontological interoperability in an agent-based system. The approach discussed is fully automatic.

#### 3.5.1 Rosetta

Rosetta [53] is a middleware message translation system that supports communication between heterogeneous agents using ontology merging techniques. Agents register their goal representations and planning capabilities modelled as ontologies with Rosetta, which creates a model for each agent. Rosetta acts as a matchmaker between agents, matching requests with required capabilities. Agents in Rosetta might use different ontology languages to describe goals, operators and plans and thus require translations between them in order to communicate. Rosetta provides translations between heterogeneous ontologies through the use of a common overarching ontology called Planet [54]. The Rosetta approach facilitates scalable agent architecture as

any agent can talk to any other agent as long as its ontology can be mapped into the Planet ontology thus requiring only a single interface. The main drawback is that once the rewrite rules are written the translation is automatic, but the generation of the re-write rules for syntactic translation between agent ontologies and Planet require input from a domain expert.

### 3.6 Other Agent-based automatic approaches

Agent-based approaches that do not rely on intermediary meta-ontologies or lexicons and are fully automatic are covered in this section.

#### 3.6.1 Language Games

The solution to the interoperability problem according to Wiesman et al. [56] [57] lies in solving the structural and semantic heterogeneity using language games [58] to learn ontology mappings. Structural heterogeneity in this context refers to representational conflict such as the family name of a person could be “Van Der Herik” in an ontology and “Herik, Van Der” in another. This type of representational conflict should not be confused with the notion of syntactic interoperability as mentioned in section 2 which deals mainly with representation differences in different ontology languages. In language games, two agents try to communicate about a concept, gradually building and modifying their private ontologies and lexicon based on success or failure of the games. In order to use language games for ontology mapping, an assumption is made that agents wishing to communicate about a concept share some instances of this concept, this determines the *joint attention* of the agents. In order to establish joint attention for a concept, one agent produces an *utterance*, which is a unique representation of the concept and an instance of the concept. The other agent tries to match to a certain degree a concept and its instance from the agent’s own ontology. For this the agent measures the proportion of words (each concept in an utterance contains the collection of attribute value pairs or words associated with the particular instance of the concept in question) the two instances have in common. The instance with the highest proportion of corresponding words, forms, together with the communicated instance, the *joint attention*. Subsequently a mapping is established between the two concepts. The experimentation was limited to resolving database mappings.

#### 3.6.2 Description compatibility

Description compatibility is another approach for agents to find compatible services across communities subscribing to differentiated ontologies (where terms have formal definitions as concepts related to other concepts, and local concepts inherit from concepts that are shared, and that most or all primitives are shared) [59] [60]. Ontological Concept definitions are converted to *description graphs* [61] implemented in Very Basic Description Logic (VBDL), an extension of the Description Logic group of languages. Description compatibility measures of the *syntactic correspondence* (syntactic equivalence) between definitions of pairs of



terms are used to search for satisfactory mappings or to bypass them altogether. Artificial ontologies are generated based on the potential target services. Given two terms, *semantic overlap* (semantic equivalence) of their meanings is calculated using the intersection of their denoted sets and is used for evaluating the accuracy of the syntax (in this instance syntax refers to the structure) based compatibility measures. The background of this research is agent brokerage services where agents ask brokers for service recommendations. The broker recommends a service description that map most strongly to the request. The service description is a modelled as a concept in the ontology. Eight measures of description compatibility between pairs of concept definitions were proposed. The proposed approach has not been implemented as a real-world application and testing has been limited to simulated study without actual differentiated ontologies. The proposed eight measures provide varying degree of semantic overlap and require a degree of discrimination in their application for specific purpose tasks. Out of the eight measures, *matching structure* (PMHR) is claimed to be the most accurate and usable in realistic systems.

#### 4 Additional approaches

In addition to the methodologies discussed in the previous section, many other approaches address specific issues with merging ontologies, knowledge bases or even databases. These descriptions are not as comprehensive; however, they provide valuable insight into related techniques for developing consensus ontologies.

SHOE [37] is a web based knowledge representation language that supports multiple versions of ontologies. SHOE based ontologies are made publicly available by locating them on web pages. Ontology reuse is accomplished by extending general ontologies. Addition or removal of categories, relations and / or axioms results in revision of an ontology. Revisions may also extend a new ontology. The primary purpose of SHOE is to provide a common global ontology. There are three basic methods to achieve ontology integration across different versions of SHOE as proposed by Heflin et al. [63].

The Semint (SEMantic INTEgrator) system developed by Li et al. [64] [65] uses machine learning to address semantic heterogeneity in database schemas. Since providing interoperability in databases is similar to providing interoperability in ontologies, Semint is a good candidate for automating semantic integration in ontologies. Neural networks were used to identify similarities between attributes (classes of data items, this includes field names, data types, and any applicable constraints) from two database schemas. Semint is used to identify heterogeneity in databases and build mappings in the form of equivalence relationship maps in the pre-processor stage thus eliminating the need for mappings at query time. The approach requires little human intervention except for result evaluation at the end. Semint is suitable only for specific frameworks of localised problem domains as the means of determining similarity is learned on a per-database basis.

Loom [69] knowledge representation system is based on description logics and has been used in natural language processing [70], intelligent user interfaces, explainable expert systems, diagnostic expert systems, and interfacing to multiple expert systems [71]. The centre of the Loom system is its inference engine called the “classifier”. The job of the Loom classifier is to compute subsumption (superset / subset) relationships between concepts and to compute instance-of relationships between concepts and instances which can be used for identifying candidates for mappings between heterogeneous ontologies. The unique property of the Loom classifier is its ability to concurrently update its computations when modifications are made to either concepts or instances of concepts. Loom also provides multiple programming interfaces thus facilitating the integration of multiple inference tools. Albeit Loom is a single ontology classifier, the techniques used can also be used in cross-classifying ontologies for interoperability.

#### 5 Discussion

Although there are considerable differences between the methodologies described above, a number of points clearly emerge. Some approaches cater more specifically to ontology merging (OntoMerge, Prompt, Anchor Prompt, and Chimaera), some others to ontology mapping (ITTalks, OntoEdit, Anchor Prompt, GLUE, Rosetta) and some others to aligning ontologies (Prompt).

Majority of the non-agent-based approaches (ITTalks, OntoEdit, OntoMerge, Prompt, Anchor Prompt, and GLUE) are semi-automatic (except for ODEMerge) as opposed to agent-based approaches (ONP, work done by Williams et al.) which provide automatic integration of ontologies. In general syntactic translation has been automated to a greater degree than semantic translation. Most approaches to semantic mapping that consider structural relationships between concepts, base their analysis on studying concepts that are directly related to the concept in question e.g., in the ITTalks approach attributes associated with concepts were not utilised in the mapping process. Prompt and Chimaera consider subclass and superclass relationships and attributes directly attached to a concepts. Prompt also considers concepts being referred to by the attributes associated with the concepts in question. Anchor Prompt claims to analyse non-local context by considering even classes that are not directly related to the concepts being analysed but the details are not provided. Automatic agent-based approaches that use some lexicon (Williams et al., ONP) exploit simple one-to-one semantic relationships between concepts as compared to semi-automatic approaches.

Most semi-automatic approaches rely heavily on input from domain experts, but this can introduce bias as well as potential for error e.g., in the case of ITTalks an identified weakness of the approach was the potential for possible misclassification due to the human evaluation of the landmark mappings, which also applies to the Anchor Prompt system. In OntoEdit various techniques were utilised for extraction, but the application of those techniques was undertaken in an ad hoc manner with the expert user making all the relevant decisions. In

OntoMerge experts add bridging axioms manually to bridge terms in two related ontologies as well as to prepare the newly merged ontology for further mergers, choosing different bridging axioms or adding more bridging axioms would result in different outcomes.

Few approaches for example Chimaera allow merger of ontologies built using different formalisms like Knowledge Interchange Format [25] and OKBC [22]. The ONP approach allows *strange* agents built using different environment to communicate.

The quality and amount of descriptive text associated with the concepts in the ontology also affected the mapping results like in the case of ITTalks. Other factors that influenced the results were: the assumption of mutual exclusivity made for the Bayesian approach, which might not hold true for all leaf nodes (and could result in error) or using a classifier other than Rainbow would produce different results. In OntoEdit the techniques used in the import, reuse and extraction phases greatly vary with input data. The Anchor-Prompt algorithm produces good results only if ontologies had similar structures. The tools used in pre-processing data also influence the mapping results.

The use of global, overarching upper level ontology to serve as an inter-lingua (Cyc, SHOE, Ontolingua, SUMO, Penman upper model, Rosetta) is not practical because it is not possible to build one that covers all possible present and future ontologies. The agent-based approaches that use the WordNet lexicon for determining semantic equivalence among terms (ONP, work done by Williams et al.) are also prone to error due to the inherent inconsistencies within WordNet [52].

Few approaches (Prompt, Anchor Prompt, Chimaera, GLUE, and Description Compatibility) give experimental results on their quality and utility.

Currently there exist a variety of heuristics and other techniques that can be utilised for semantic interoperability as described in section 3, but there is still plenty of scope for refinement and for providing fully automated frameworks.

## 6 Conclusion

Providing semantic interoperability among heterogeneous ontologies is still primarily a semi-automated process. If ontologies for multi-agent systems and the semantic web are to realise their full potential, it is important to fully automate the semantic translation among ontologies. The issues outlined above must be addressed in order to establish a generic, domain independent, fully automated approach for interoperability across heterogeneous ontologies.

## 7 References

- [1] Sullivan R., "Google Florida Update," in [Online document], cited: March, 2004. Available:<http://www.searchengineposition.com/info/articles/GoogleFloridaUpdate.asp>.
- [2] Noy N., F., Fergerson, R., W., Musen, M., A., "The Knowledge model of Protege-2000: combining interoperability and flexibility," presented at 2th International Conference on Knowledge Engineering and Knowledge Management, Juan-les-Pins, France, 2000, pp. 17 - 32.
- [3] Dou D., McDermott, D., Qi, P., "Ontology Translation on the Semantic Web," presented at International Conference on Ontologies, Databases and Applications of Semantics, 2003, pp. 952 - 969.
- [4] Williams A., Padmanabhan, A., Blake, M., B., "Local consensus ontologies for B2B oriented service composition," presented at Second Joint International Conference on Autonomous Agents and Multi Agent Systems, Melbourne, 2003, pp. 647 - 654.
- [5] Maedche A., Staab, S., "Ontology learning for the semantic web," *IEEE Intelligent Systems*, vol. 16, 2001, pp. 72 - 79.
- [6] FIPA, "FIPA 97 Part 2 Version 2.0: Agent Communication Language Specification," in [Online document], cited: December, 2004. Available:<http://www.fipa.org/specs/fipa00003/>.
- [7] Finin T., Labrou, Y., Mayfield, J., "KQML as an agent communication language," in *Software Agents*, Jeff Bradshaw, Ed. Cambridge: AAAI / MIT Press, 1997, pp. 291 - 316.
- [8] Cost R., S., Finin, T., Joshi, A., Peng, A., Nicholas, C., Soboroff, I., Chen, H., Kagal, L., Perich, F., Zou, Y., Tolia, S., "ITTalks: A Case Study in the Semantic Web and DAML," presented at First International Semantic Web Workshop - Infrastructure and Applications for the Semantic Web, Stanford, 2001, pp. 477 - 494.
- [9] Cost R., S., Finin, T., Joshi, A., Peng, A., Nicholas, C., Soboroff, I., Chen, H., Kagal, L., Perich, F., Zou, Y., Tolia, S., "ITTalks: A Case Study in the Semantic Web and DAML+OIL," *IEEE Intelligent Systems*, 2002, pp. 40 - 47.
- [10] DARPA, "DAML: Darpa Agent Markup Language," in [Online document], cited: March, 2004. Available:<http://www.daml.org/>.
- [11] Prasad S., Peng, Y., Finin, T., "Using Explicit Information To Map Between Two Ontologies," presented at Workshop on Ontologies in Agent Systems as part of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, 2002, pp. 52 - 56.
- [12] McCallum A., "Rainbow," in [Online document], cited: January, 2004. Available:<http://www-2.cs.cmu.edu/~mccallum/bow/rainbow/>.
- [13] University Karlsruhe, "Ontoprise: Semantics for the web," in [Online document], cited: January, 2004. Available:<http://www.ontoprise.com>.
- [14] Kifer M., Lausen, G., "F-logic: a higher-order language for reasoning about objects, inheritance, and scheme," presented at ACM SIGMOD international conference on Management of data, Portland, Oregon, United States, 1989, pp. 134 - 146.
- [15] W3C, "Resource Description Framework (RDF)," in [Online document], cited: January, 2002. Available:<http://www.w3.org/RDF/>.
- [16] McDermott D., V., Dou, D., Qi, P., "OntoMerge," in [Online document], cited: December, 2003. Available:<http://cs-www.cs.yale.edu/homes/dvm/daml/ontology-translation.html>.
- [17] Dou D., McDermott, D., Qi, P., "Ontology translation by ontology merging and automated reasoning," presented at

- EKA2002 Workshop on Ontologies for Multi-Agent Systems, Sigüenza, Spain, 2002, pp. 3 - 18.
- [18] McDermott D., Dou, D., Qi, P., "PDDL - An Automatic Translator Between PDDL and DAML," in [Online document], cited: January, 2004. Available: [http://www.cs.yale.edu/homes/dvm/daml/pddl\\_damtranslator1.html](http://www.cs.yale.edu/homes/dvm/daml/pddl_damtranslator1.html).
- [19] McCarthy J., "RECURSIVE FUNCTIONS OF SYMBOLIC EXPRESSIONS AND THEIR COMPUTATION BY MACHINE (Part I)," *Communications of the ACM*, vol. 3, 1960, pp. 184 - 195.
- [20] Noy N., F., Musen, M., A., "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," presented at Seventeenth National Conference on Artificial Intelligence (AAAI-2000), Austin, TX, 2000, pp. 450 - 455.
- [21] Noy N., F., Musen, M., A., "SMART: Automated support for ontology merging and alignment," presented at Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW '99), Banff, Alberta, Canada, 1999, pp. 1 - 20.
- [22] Chaudhri V., K., Farquhar, A., Fikes, R., Karp, P., D., Rice J., P., "OKBC: A Programmatic Foundation for Knowledge Base Interoperability," presented at Fifteenth National Conference on Artificial Intelligence, Madison, Wisconsin, 1998, pp. 600 - 607.
- [23] McGuinness D., L., Fikes, R., Rice, J., and Wilder, S., "An Environment for Merging and Testing Large Ontologies," presented at Seventh International Conference on Principles of Knowledge Representation and Reasoning, Breckenridge, Colorado, USA, 2000, pp. 483 - 493.
- [24] Noy N., F., Musen, M., A., "Anchor-PROMPT: Using Non-Local Context for Semantic Matching," presented at Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence, Seattle, WA, 2001.
- [25] Genserech M., Fikes, R., "Knowledge Interchange Format," Computer Science Department, Stanford University, Stanford, USA, Technical Report, Logic-92-1 1992.
- [26] Swartout B., Patil, R., Knight, K., Russ, T., "Toward distributed use of large-scale ontologies," presented at Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Alberta, Canada, 1996, pp. 32.1 - 32.19.
- [27] Domingue J., "Tadzebao and Webonto: Discussing, Browsing and editing ontologies on the web," presented at Eleventh Knowledge Acquisition Workshop, KAW98, Banff, 1998.
- [28] Smith D., C., "Plisp User's Manual," Apple Computer, August 1990.
- [29] MacGregor R., M., "A Description Classifier for the Predicate Calculus," presented at Twelfth National Conference on Artificial Intelligence, Seattle, Washington, 1994, pp. 213 - 220.
- [30] Doan A., Madhavan, J., Domingos, P., Halevy, A., "Learning to map between ontologies on the semantic web," presented at Eleventh international conference on World Wide Web, Hawaii, 2002, pp. 662 - 673.
- [31] Rijsbergen J., V., *Information Retrieval*, Second ed. London: Butterworths, 1979.
- [32] Domings P., Pazzani, M., J., "On the optimality of simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, 1997, pp. 103 - 130.
- [33] Gruber T. R., "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, vol. Vol.5, 1993, pp. 199-220.
- [34] Lenat D., Guha, R., *Building Large Knowledge-based Systems: Representation and Interface in the Cyc Project*. Reading, MA: Addison-Wesley, 1990.
- [35] Niles I., Pease, A., "Origins of the Standard Upper Merged Ontology: A Proposal for the IEEE Standard Upper Ontology," presented at Workshop on the IEEE Standard Upper Ontology at IJCAI-2001, Seattle, Washington, 2001, pp. 37 - 42.
- [36] Bateman J., Kasper, R., Moore, J., Whitney, R., "A general organization of knowledge for natural language processing: The Penman Upper Model," Information Sciences Institute, Marina del Rey, California, Technical Report 1989.
- [37] Heflin J., Hendler, J., Luke, S., "SHOE: A Knowledge Representation Language for Internet Applications," Institute for Advanced Computer Studies, University of Maryland, Technical, CS-TR-4078 1999.
- [38] (NLM) National Library of Medicine, *Medical Subject Headings*. Bethesda, MD, 1994.
- [39] (CAP) College of American Pathologists, *SNOMED*, Second edition ed. Skokie, IL.: College of American Pathologists., 1982.
- [40] Lindberg D., Humphreys, B. and McCray, A., "The Unified Medical Language System (UMLS)," in *1993 Yearbook of Medical Informatics*, J. van Bommel, Ed. Amsterdam: International Medical Informatics Association, 1993, pp. 41-53.
- [41] Fellbaum C., *WordNet: An Electronic Lexical Database*. Cambridge, MA, USA: MIT Press, 1998.
- [42] Reed S., Lenat, D., B., "Mapping ontologies into CYC," presented at AAAI 2002 Conference Workshop on Ontologies For The Semantic Web, Edmonton, Canada, 2002.
- [43] Ramos J., A., "Mezcla automática de ontologías y catálogos electrónicos," Facultad de Informática de la Universidad Politécnica de Madrid, Madrid, Spain 2001.
- [44] Consortium OntoWeb, "A survey on ontology tools," OntoWeb Consortium, IST-2000-29243 2002.
- [45] de Diego R., "Método de mezcla de catálogos electrónicos," Facultad de Informática de la Universidad Politécnica de Madrid, Madrid, Spain 2001.
- [46] Goasdoue F., Lattes, V., Rousset, M.-C., "The use of carin language and algorithms for information integration: The picse project," *International Journal of Cooperative Information Systems*, vol. 9, 1999, pp. 383 - 401.
- [47] Friedman-Hill E., "JESS," in [Online document], cited: December, 2003. Available: <http://herzberg.ca.sandia.gov/jess/>.
- [48] Bailin S., Truszkowski, W., "Ontology negotiation between agents supporting intelligent information management," presented at Workshop on Ontologies in Agent-based Systems (OAS 2001), ACM Agents 2001 Conference, Montreal, 2001, pp. 13 - 20.
- [49] Bailin S., Truszkowski, W., "Ontology negotiation between Intelligent Information Agents," *Knowledge Engineering Review*, vol. 17, 2002, pp. 7 - 19.

- [50] Stephens L., Huhns, M., N., "Consensus Ontologies: Reconciling the Semantics of Web Pages and Agents," *IEEE Internet Computing*, vol. 5, 2001, pp. 92 - 95.
- [51] Levenshtein V., I., "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics - Doklady*, vol. 10, 1966, pp. 707 - 710.
- [52] Hanks P., "WordNet: What is to be done?," in [Online document], cited: December, 2003. Available: <http://ckl.mff.cuni.cz/events/lexsem/hanks-panel.pdf>.
- [53] Blythe J., Chalupsky, H., Gil, Y., MacGregor, R., "Rosetta: Ontology-Based Agent Communication with Rosetta," in [Online document], cited: January, 2004. Available: <http://www.isi.edu/expect/projects/agents/rosetta.html>.
- [54] Gil Y., Blythe, J., "PLANET: A Shareable and Reusable Ontology for Representing Plans," presented at AAAI 2000 workshop on Representational Issues for Real-world Planning Systems, 2000, pp. 28 - 33.
- [55] Cohen P., Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., Burke, M., "The DARPA High-Performance Knowledge Bases Project," *Artificial Intelligence Magazine*, vol. 19, pp. 25 - 49.
- [56] Wiesman F., Roos, N., Vogt, P., "Automatic ontology mapping for agent communication," Maastricht : MERIT, Maastricht Economic Research Institute on Innovation and Technology, MERIT - Infonomics Research Memorandum Series, 2001 - 23 2001.
- [57] Wiesman F., Roos, N., Vogt, P., "Automatic ontology mapping for agent communication," presented at First international joint conference on Autonomous agents and multiagent systems, 2002, pp. 563 - 564.
- [58] Steels L., "Emergent Adaptive Lexicons," presented at Fourth International Conference on Simulating Adaptive Behaviour, Cambridge, MA, USA, 1996, pp. 562 - 567.
- [59] Weinstein P., Birmingham, W., "Agent communication with differentiated ontologies: eight new measures of description compatibility," Department of Electrical Engineering and Computer Science, University of Michigan, CSE-TR\_383\_99 1999.
- [60] Weinstein P., Birmingham, W., "MatchingRequests for Agent Services with Differentiated Vocabulary," presented at Fourteenth National Conference on Artificial Intelligence, Providence, Rhode Island, 1997, pp. 850.
- [61] Cohen W., W., Hirsh, H., "The Learn-ability of Description Logics with Equality Constraints," *Machine Learning*, vol. 17, 1994, pp. 169 - 199.
- [62] Baader F., Horrocks, I., Sattler, U., "Description Logics as Ontology Languages for the Semantic Web," in *Festschrift in honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence*, D. Hutter, Stephan, W., Ed.: Springer-Verlag, 2003.
- [63] Heflin J., Hendler, J., Luke, S., "Coping with Changing Ontologies in a Distributed Environment," presented at AAAI-99 Workshop on Ontology Management, Orlando, Florida, 1999, pp. 74 - 79.
- [64] Li W.-S., "Semint: a system prototype for semantic integration in heterogeneous databases," presented at ACM SIGMOD international conference on Management of data, 1995, pp. 484.
- [65] Li W.-S., "Knowledge gathering and matching in heterogeneous databases," presented at Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, 1995, pp. 116 - 121.
- [66] Kohonen T., "Adaptive, Associative, and Self-organising functions in Neural Computing," *Applied Optics*, vol. 26, 1987, pp. 4910 - 4918.
- [67] Li W.-S., "Semantic integration in heterogeneous databases using neural networks," presented at 20th International Conference on Very Large Data Bases, Santiago, Chile, 1994, pp. 1 - 12.
- [68] Gaines B. R., "SISYPHUS," in [Online document], cited: December, 2003. Available: <http://ksi.cpsc.ucalgary.ca/KAW/Sisyphus/>.
- [69] MacGregor R., "The Evolving TEchnology of Classification-based Knowledge Representation Systems," in *Principles of Semantic Networks*, Sowa, Ed. Los Altos, CA: Morgan Kauffmann, 1991, pp. 385 - 400.
- [70] Kasper R., T., "Unification and classification: an experiment in information-based parsing," presented at International Work shop on Parsing Technologies, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1989, pp. 1 - 7.
- [71] MacGregor R., "Retrospective on Loom," in [Online document], cited: January, 2004. Available: [http://www.isi.edu/isd/LOOM/papers/macgregor/Loom\\_Retrospective.html](http://www.isi.edu/isd/LOOM/papers/macgregor/Loom_Retrospective.html).

# Aspects of Automatic Ontology Extension: Adapting and Regeneralizing Dynamic Updates

Ekaterina Ovchinnikova<sup>1</sup>

Kai-Uwe Kühnberger<sup>2</sup>

<sup>1</sup> Seminar für Sprachwissenschaft  
University of Tübingen  
72074 Tübingen, Wilhelmstr. 19, Germany  
Email: e.ovchinnikova@gmail.com

<sup>2</sup> Institute of Cognitive Science  
University of Osnabrück  
49076 Osnabrück, Albrechtstr. 12, Germany  
Email: kkuehnbe@uos.de

## Abstract

Ontologies are widely used in text technology and artificial intelligence. The need to develop large ontologies for real-life applications provokes researchers to automatize ontology extension procedures. Automatic updates without the control of a human expert can generate potential conflicts between original and new knowledge. As a consequence the resulting ontology can contain inconsistencies. On the other hand, even if the information extracted from the external sources automatically is consistent with the original ontology it can be generalized unsystematically and conceptually wrong. This in turn can lead to mistakes in applications of the extended ontology. We propose an algorithm that models the process of the adaptation of an ontology to new information and regeneralizes the resulting ontology in a more intuitive way inserting additional knowledge where this is possible.

## 1 Introduction

There is an increasing interest in applying and using ontological knowledge in artificial intelligence. Examples for applications of ontologies in AI are expert systems, dialogue systems, robotics, reasoning systems, web services, and text technological tools. In general, knowledge-based systems are prototypical examples for using and applying ontological knowledge. The interested reader is referred to [www.cs.utexas.edu/users/mfkb/related.html](http://www.cs.utexas.edu/users/mfkb/related.html), where a long list of different knowledge-based systems and ontology projects can be found.

An important motivation for research in ontology design is the fact that inference processes can be made more efficient. For example, an ontology with a subsumption relation based on a many-sorted logic allows to restrict inferences to those rules that are in accordance to the sortal constraints. A classical (and famous) application in the field of theorem proving is the steamroller problem (Walter 1985) where the number of clauses that are necessary to solve the problem can be significantly reduced by introducing hierarchical constraints on these sorts. Besides such technical aspects, there is a further very general reason for the endeavor to develop models for ontological

systems: ontologies are still one of the few possibilities to explore the hard problem, whether machines can assign meanings to symbols, i.e. whether machines can develop an important aspect of human-level intelligence.

The most important new development motivating many researchers on focusing on ontologies is the omnipresence of the world wide web together with its numerous applications and its economic importance. It is often claimed that ontological (i.e. semantic) knowledge about domains of interest is one of the most important steps in order to develop new and intelligent web applications (Berners-Lee, Hendler & Lassila 2001). Examples for such services are intelligent search tools for large archives of multi-modal information, multi-modal resources for artificial agents and personal assistants (that are permanently connected with the internet), or intelligent document management tools for libraries and companies. But also e-commerce applications, the development of portals, or geospatial applications could benefit from ontological knowledge.

Since the manual development of large ontologies has been proven to be a very tedious, time-consuming and expensive task, automatic procedures for semantic annotations of relevant resources (texts and web content) and the possibility to automatically adapt and extend such ontologies would be desirable. Therefore one can find many current investigations that are devoted towards a development of automatic ontology learning methods (Gómez-Pérez & Manzano-Macho 2003).

During the last decades several formalisms have been proposed to represent ontological knowledge. In recent years the world wide web and its connection to various economically important applications has been provided the environment for dynamic developments in representation language standards. Probably the most important one of existing markup languages for ontology design is the Web Ontology Language *OWL* (OWL 2004) in its three different versions *OWL Lite*, *OWL DL*, and *OWL Full* (W3C 2004). The mentioned *OWL* versions are hierarchically ordered, such that *OWL Full* includes *OWL DL*, and *OWL DL* includes *OWL Lite*. Consequently they differ in their expressive strengths with respect to possible concept formations.

All versions of *OWL* are based on the logical formalism called Description Logic *DL* (Baader et al. 2003). Description logics were originally designed for the representation of terminological knowledge and reasoning processes. They can be characterized as subsystems of first-order predicate logic using at most two variables. Two points should be mentioned:

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Australasian Ontology Workshop (AOW 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 72. M. A. Orgun and T. Meyer, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.



- In comparison to full first-order logic, description logics are – due to their restrictions concerning quantification – rather weak logics with respect to their expressive strength. Nevertheless they are considered as appropriate representation formalisms for ontological knowledge.
- DL can be used to characterize the different OWL versions. For example, *OWL DL* can be logically characterized as a syntactic variant of the description logic  $\mathcal{SHOIN}(\mathbf{D})$  (Motik, Sattler & Studer 2004). As a consequence of the clear logical foundation of the OWL versions using description logics, important formal properties of the different OWL versions can be specified, for example, their decidability properties: whereas *OWL Full* is undecidable (due to the lack of restrictions to transitive properties), *OWL DL* and *OWL Lite* are decidable.

Although most of the tools extracting or extending ontologies automatically output the knowledge in the OWL-format, they usually use only a small subset of the underlying description logic. Core ontologies generated in practice consist of a set of concepts, the subsumption relation defined on concepts (inducing a taxonomy) and general relations (such as part-of) defined on concepts. At present complex ontologies making use of the whole expressive power and advances of the various versions of description logics can be achieved only manually or semi-automatically. Disadvantages of manual or semi-automatic applications of knowledge representation formalisms are costs (expensive and time-consuming).

However, several approaches appeared recently tending not only to learn taxonomic and general relations but also to state which concepts in the knowledge base are equivalent or disjoint (Haase 2005). In the present paper, we concentrate on these approaches. We will consider only terminological knowledge (called TBox in DL) leaving the information about assertions in the knowledge base (called ABox in DL) for the further investigation.<sup>1</sup>

Approaches of automatic ontology learning and automatic ontology extension – in particular if they are based on rather expressive logics – are often faced with the so-called generalization problem.<sup>2</sup> The appropriate level of granularity of an underlying ontology is usually hard to achieve. Two major problems can be distinguished:

- Inappropriate generalizations of concepts can lead, in the worst case, to inconsistencies if ontologies are automatically extended. Assume a concept  $C$  in the ontology was overgeneralized, then a new axiom that must be added to the ontology – due to new available information – can represent an exception towards  $C$  and can conflict with its definition. In this case  $C$  is too coarse. Resolving inconsistencies in logic based systems is well-known to be a hard problem.
- The undergeneralization of concepts in an ontology does not provoke inconsistencies, but it can lead to a loss of information by an ontology application (Ceusters et al. 2003). In this case, the underlying concept must be generalized because in its original form it is too fine-grained.

In this paper, we provide an overview of the mentioned generalization problems in ontologies automatically learned from external sources. Sections 3 and 4

<sup>1</sup>The formal definition of terminological knowledge coded in a TBox is stated in Section 2.

<sup>2</sup>For a motivation compare (Haase et al. 2005).

discuss the overgeneralization problem, whereas Section 5 deals with the undergeneralization problem. We give algorithmic solutions for both problems, in particular we specify how to resolve occurring contradictions and get additional knowledge where this is possible, and how regeneralizations of an ontology can be achieved if some underlying concepts is too fine-grained.

The paper has the following structure: In Section 2, we roughly summarize some important definitions of the syntax and semantics of description logics and we introduce the notion of least common subsumer. Section 3 starts with a rough summary of classical existing approaches to model inconsistent information and presents some intuitive ideas how occurring inconsistencies in ontology extension processes can be resolved. In Section 4, we present the algorithm *AdaptOnto* that allows the extension of ontologies with inconsistent information by an adaptation process. Section 5 addresses the generalization problem in consistent ontologies and proposes an algorithmic solution of this problem by the algorithm *Regen* as well as the prototype implementation. Last but not least, Section 6 adds some remarks concerning semantic issues and Section 7 concludes the paper.

## 2 Description Logic

### 2.1 Basic Definitions

In this section, we define the DL-logic underlying the ontological knowledge representation considered in this paper.<sup>3</sup> For a detailed presentation and an overview of various description logics, including their syntax and semantics, the reader is referred to (Baader et al. 2003).

Given a set of concept names  $N_C$  and a set of role names  $N_R$  a *TBox* (terminological box) is a finite set of axioms of the form  $A_1 \equiv A_2$  (equalities) or  $A \sqsubseteq C$  (inclusions) where  $A$  stands for a concept name and  $C$  (called *concept description*) is defined as follows ( $R$  denotes a role name):

$$C \rightarrow A \mid \neg A \mid \forall R.A$$

The symbol  $\doteq$  denotes the syntactical equality of concept descriptions. The concepts occurring on the left side of an axiom are called *axiomatized* ( $ax$ ). In an axiom with a concept  $A$  on the left side the concept on its right side is called *definition* of  $A$ .

Concept descriptions are interpreted in a classical model-theoretic sense (for details compare (Baader et al. 2003)). An *interpretation*  $\mathcal{I}$  is a pair  $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  where  $\Delta^{\mathcal{I}}$  is a non-empty domain of individuals and the interpretation function  $\cdot^{\mathcal{I}}$  maps concept names to subsets of  $\Delta^{\mathcal{I}}$  and role names to subsets of  $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ . Concept descriptions are interpreted as follows:

$$\begin{aligned} (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (\forall R.A)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \rightarrow y \in A^{\mathcal{I}}\} \end{aligned}$$

An interpretation  $\mathcal{I}$  is a *model* of a TBox  $\mathcal{T}$  if for every inclusion  $A \sqsubseteq C$  in  $\mathcal{T}$  it holds  $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$  and for every equality  $A_1 \equiv A_2$  we have  $A_1^{\mathcal{I}} = A_2^{\mathcal{I}}$ . A concept description  $D$  *subsumes*  $C$  in  $\mathcal{T}$  (formally represented by  $\mathcal{T} \models C \sqsubseteq D$ ) if for every model  $\mathcal{I}$  of  $\mathcal{T}$ :  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ . A concept  $C$  is called *satisfiable towards*  $\mathcal{T}$  if there is a model  $\mathcal{I}$  of  $\mathcal{T}$  such that  $C^{\mathcal{I}}$  is nonempty. Otherwise  $C$  is called *unsatisfiable* and it holds  $\mathcal{T} \models C \sqsubseteq \perp$ .

<sup>3</sup>In the following definitions we closely follow (Haase et al. 2005) who present an approach using one of the most powerful DL-version in ontology learning procedure.

Algorithms for checking satisfiability of concept descriptions have been implemented in several reasoning systems.<sup>4</sup> For example, the FaCT system implements subsumption for a very expressive DL  $\mathcal{SHIQ}$  (Horrocks 1998). Most of these systems are based on Tableau calculi. The idea is to use facts about the world (coded in the ABox) in order to construct a model for these facts (relative to the given TBox). The construction is usually performed by so-called *expansion rules* decomposing underlying concepts until no further application of a rule is possible or a contradiction is reached. It should be noted that these reasoners usually use the well-known correspondence between the subsumption  $C \sqsubseteq D$  and the unsatisfiability of  $C \sqcap \neg D$ .

## 2.2 Least Common Subsumers

Recent research in description logics is strongly concerned with *non-standard inferences* (Baader & Küsters 2006) tending to support bottom-up constructions of knowledge bases. A knowledge engineer introduces typical examples of a new concept and the system tries to find commonalities between them and generalizes it in a definition.

An important task for generalizing a new concept is the computation of the *least common subsumer* for a set of concepts (first mentioned in (Cohen et al. 1993)). Intuitively, the least common subsumer (lcs) for two concept descriptions  $C_1$  and  $C_2$  is a concept description that collects all common features of  $C_1$  and  $C_2$  and is most specific towards subsumption.

**Definition 1** A concept description  $E$  of DL  $\mathcal{L}$  is a *least common subsumer (lcs)* of the concept descriptions  $C_1, \dots, C_n$  in  $\mathcal{L}$  ( $\text{lcs}_{\mathcal{L}}(C_1, \dots, C_n)$  for short) if and only if the following two conditions are satisfied:

1.  $\forall i \in \{1, \dots, n\} : C_i \sqsubseteq E$  and
2.  $\forall E' \in \mathcal{L} : \text{if } \forall i \in \{1, \dots, n\} : C_i \sqsubseteq E' \text{ then } E \sqsubseteq E'$

There are algorithms for computing lcs for different DL logics (Baader & Küsters 2006). All of these algorithms work with logics allowing at least the existence of a top element. Because of the specificity of the logic considered in this paper (no conjunction, no top and bottom elements, multiple definitions for one concept) we define the set of the least common subsumers *lcss* for the set of concept descriptions  $C_1, \dots, C_n$  towards a TBox  $\mathcal{T}$  slightly differently from the usual way.

First of all, let us recursively define the function *subsumers* computing the set of all possible subsumers (formulated in the DL under consideration) for a concept  $C$  towards a TBox  $\mathcal{T}$ :

$$\begin{aligned} \text{subsumers}_{\mathcal{T}}(C) = \{ & D \mid D \sqsupseteq C \vee C \sqsubseteq D \in \mathcal{T} \vee \\ & \exists D' : C \sqsubseteq D' \in \mathcal{T} \wedge D \in \text{subsumers}_{\mathcal{T}}(D') \} \cup \\ & \{ \neg A \mid \exists A' : C \sqsupseteq \neg A' \wedge A \in \text{subsumers}_{\mathcal{T}}(A') \} \cup \\ & \{ \forall R.A \mid \exists A' : C \sqsupseteq \forall R.A' \wedge A' \in \text{subsumers}_{\mathcal{T}}(A') \} \end{aligned}$$

The definition can be summarized as follows: For every concept description  $C$  the set  $\text{subsumers}_{\mathcal{T}}(C)$  contains the following elements:

- (a)  $C$  itself;
- (b) Concept descriptions occurring on the right side of the axioms in  $\mathcal{T}$  that axiomatize  $C$ ;
- (c) Concept descriptions that subsume the concept descriptions from (b) (the subsumption relation is transitive).

If  $C$  is a negated atomic concept ( $C \doteq \neg A'$ ), then the set  $\text{subsumers}_{\mathcal{T}}(C)$  collects the negated subconcepts of  $A'$ . Since general axioms (with concept descriptions on the left side) are disallowed in our logic, atomic concepts can subsume only atomic concepts and no negations or value restrictions. Every value restriction  $\forall R.A'$  is subsumed only by the relational restrictions  $\forall R.A$  where  $A$  subsumes  $A'$ .

According to this constructive definition the set *subsumers* is exhaustive in our DL as stated explicitly in Fact 1:

**Fact 1** For every TBox  $\mathcal{T}$ , for all concept descriptions  $C, C'$ :

$$C' \in \text{subsumers}_{\mathcal{T}}(C) \Leftrightarrow \mathcal{T} \models C \sqsubseteq C'$$

Now we define the function *lcss* computing the set of least common subsumers for a set of concept descriptions  $C_1, \dots, C_n$  towards a TBox  $\mathcal{T}$  according to Definition 1.

**Definition 2** A set  $L$  is the set of least common subsumers of the concept descriptions  $C_1, \dots, C_n$  towards the TBox  $\mathcal{T}$  ( $\text{lcss}_{\mathcal{T}}(C_1, \dots, C_n)$ ) if and only if it is specified as follows:

1.  $CS = \bigcap_{i \in \{1, \dots, n\}} \text{subsumers}_{\mathcal{T}}(C_i)$  and
2.  $L = \{C \in CS \mid \forall C' \in CS : \mathcal{T} \models C' \sqsubseteq C \rightarrow C' \doteq C\}$

In the following sections, we show how the notion of the least common subsumer can be used for regeneralizing a TBox.

## 3 Ontology Extension: Inconsistencies

### 3.1 Classical Approaches for Inconsistent Information

Inconsistencies occurring in reasoning processes do have a long history in artificial intelligence. Due to the fact that many approaches in AI are based on one or the other form of classical logic and inconsistencies, for example, triggered by new information added to a knowledge base, cannot be easily treated in classical logic, researchers proposed many approaches to solve this problem. The difficulties in modeling inconsistencies in classical logic are strongly connected to the monotonicity property of logic. This property can be described as follows: for all sets  $\Delta$  of first-order formulas and all first-order formulas  $\phi$  and  $\psi$  it holds:

$$\text{if } \Delta \vdash \phi \text{ then } \Delta \cup \psi \vdash \phi$$

In an application, based on classical logic, an update of the set of premises  $\Delta$  with  $\psi$  such that  $\psi \leftrightarrow \neg\phi$  the consequence  $\phi$  is still provable, contrary to the intuition that we have evidence for  $\neg\phi$ . In order to avoid this type of conclusion, so-called non-monotonic reasoning techniques were developed and extensively discussed in the literature (Bibel et al. 1993). We mention three prominent approaches that were proposed to model non-monotonicity.

- Default logic (Reiter 1980): A default theory  $\langle W, \Delta \rangle$  distinguishes two types of rules.  $W$  represents a world description, i.e. strict background knowledge, whereas  $\Delta$  denotes a set of defaults, representing revisable information. Intuitively (and very simplified) this means: if we have no evidence that a formula  $\neg\theta$  is true, then assume that  $\theta$  holds.

<sup>4</sup>Some of the DL reasoners are listed at <http://www.cs.man.ac.uk/~sattler/reasoners.html>.

- Answer set programming (Baral 2003): A natural idea of modeling inconsistencies is to introduce a ranking order on rules that can be applied in reasoning systems. Intuitively more specific rules are higher ranked than very general rules, i.e. general rules can be overwritten (revised) by specific information.
- Circumscription (Lifschitz 1994): Based on the idea of logical minimization, the circumscription of a predicate  $P$  relative to a world description  $W$  means that there is no other predicate  $P'$  such that  $W$  still holds and the extension of  $P'$  is strictly smaller than the extension of  $P$ . In other words,  $P$  is minimal with respect to  $W$ .

The listed approaches represent only a few examples of theories that were proposed to model inconsistencies and non-monotonicity. Nevertheless no generally accepted solution for non-monotonic reasoning seems to be available.

### 3.2 Inconsistencies and Ontologies

We want to consider inconsistencies in ontologies more closely. An ontology based on description logic can contain contradictions only if its underlying logic allows negation. Ontologies share this property with every logical system (like, for example, first-order logic). For the approaches concerned with core ontologies no contradictions in the ontological knowledge base is possible. But for approaches using more powerful logics, the problem of inconsistency becomes very important (Haase et al. 2005). In order to make the notion of inconsistency of a TBox precise we give the following definition.

**Definition 3** A TBox  $\mathcal{T}$  is inconsistent if there exist a concept  $C \in ax(\mathcal{T})$  that is unsatisfiable.

A number of approaches have been proposed treating inconsistencies by extending the underlying description logic with additional syntactical means. Some examples are extensions by default sets (Heymans & Vermeir 2002), by planning systems (Baader et al. 2005), by belief-revision processes (Flouris et al. 2005), or by epistemic operators (Katz & Parsi 2005). Unfortunately, these approaches are beyond ordinary description logics, i.e. they cannot be coded in standard versions of description logic. Therefore the application of classical DL reasoners is impossible due to the fact that standard DL inferences cannot be performed.

There are several theoretical approaches treating occurring inconsistencies for quite expressive DL-logics, but – contrary to the cases above – do not go beyond description logic. The following list summarizes some of these approaches:

- (Ghilardi et al. 2006) suggests a characterization of non-conservative extensions of an ontology: If a concept description is satisfiable prior to an extension, but becomes unsatisfiable after the extension, then a witness concept description demonstrating this fact will be suggested to the ontology engineer. Occurring inconsistencies caused by an ontology extension can be considered as a special case of a non-conservative extension. This approach does not provide a general solution of the inconsistency problem, but can help the human expert to discover occurring inconsistencies in certain cases.
- In (Fanizzi et al. 2005), the authors propose an ontology refinement procedure based on positive

and negative assertions for concepts. If a concept  $C$  becomes unsatisfiable after an ontology extension, then the axiom defining  $C$  is replaced by a new axiom constructed on the basis of the positive assertions for this concept. Thus, the information previously defined in the TBox for the concept  $C$  gets lost.

- (Ovchinnikova & Kühnberger 2006a) introduce a procedure automatically changing the original ontology if it conflicts with new information. The changes in the conflicting axioms are performed in order to achieve a resulting ontology that is consistent. Additionally these changes can be interpreted as an adaptation process, amalgamating previous knowledge to new data.

The listed approaches (and many others dealing with non-monotonic and non-conservative extensions) are concerned with quite expressive DL-logics and tend to support a semi-automatic development of ontologies. The situation with automatic ontology learning seems to be different:

- First, there is no ontology engineer who supervises the procedure.
- Second, the changes in the ontology are supposed to be relevant and possibly minimal.
- Third, the axioms extracted automatically from the external sources can be inconsistent.
- Finally, the underlying logic tends to be rather weak with respect to its expressive power.

In the next subsection, we consider some types of examples for which an automatic adaptation process can be implemented.

### 3.3 Resolving Occurring Inconsistencies in Automatic Ontology Learning

An interesting approach towards an automatic ontology learning procedure was proposed in (Haase et al. 2005). If the ontology under consideration is provably inconsistent, then one or more axioms must be deleted from this ontology. The axioms to be deleted are chosen according to a confidence rating. This rating is computed on the basis of the term distribution in texts used for learning.

Intuitively plausible is the consequence that in some cases the removal of a whole axiom can lead to a loss of relevant information. We consider an example to make this point clear.

TBox:  $\{\text{Bird} \sqsubseteq \text{Flying}, \text{Flying} \sqsubseteq \text{Moving}, \text{Canary} \sqsubseteq \text{Bird}, \text{Penguin} \sqsubseteq \text{Bird}\}$   
 New axiom:  $\text{Penguin} \sqsubseteq \neg \text{Flying}$

By removing the information *birds fly* we will obtain the proper generalization, but lose the knowledge that all birds considered before the ontology extension can fly (such as **Canary** - the subconcept of **Bird**) and that all birds can move. We propose the following solution of how to adapt the ontology to the new information *penguin cannot fly*.

Adapted TBox:  
 $\{\text{Bird} \sqsubseteq \text{Moving}, \text{Flying} \sqsubseteq \text{Moving}, \text{FlyingBird} \sqsubseteq \text{Bird}, \text{FlyingBird} \sqsubseteq \text{Flying}, \text{Canary} \sqsubseteq \text{FlyingBird}, \text{Penguin} \sqsubseteq \text{Bird}, \text{Penguin} \sqsubseteq \neg \text{Flying}\}$

The proposed solution is simple: We want to keep in the definition of the concept **Bird** subsuming **Penguin** a maximum of information that does not



conflict with the definition of **Penguin**. The conflicting information is moved to the definition of the new concept **FlyingBird**, which is declared to subsume all former subconcepts of **Bird** (such as **Canary** for example).

Presupposing that the new axioms extracted from external sources in order to be added to the ontology contain true information we conclude that the inconsistency is provoked by overgeneralized concepts. In the example above, the statement *all birds fly* is too general for the described update, namely the introduction of a counterexample.

The example below represents a case where two overgeneralized definitions of the same concept conflict with each other:

TBox:  $\{\text{Tomato} \sqsubseteq \forall \text{hasColor.Red},$   
 $\text{Red} \sqsubseteq \text{Color}, \text{Red} \sqsubseteq \neg \text{Yellow},$   
 $\text{Yellow} \sqsubseteq \text{Color}, \text{Yellow} \sqsubseteq \neg \text{Red}\}$   
 New axiom:  $\text{Tomato} \sqsubseteq \forall \text{hasColor.Yellow}$   
 Adapted TBox:  
 $\{\text{Tomato} \sqsubseteq \forall \text{hasColor.Color}, \text{Red} \sqsubseteq \text{Color},$   
 $\text{Red} \sqsubseteq \neg \text{Yellow}, \text{Yellow} \sqsubseteq \text{Color}, \text{Yellow} \sqsubseteq \neg \text{Red}\}$

In the example above, both definitions of **Tomato** ( $\forall \text{hasColor.Red}$  and  $\forall \text{hasColor.Yellow}$ ) are too specific. **Red** and **Yellow** being disjoint concepts produce a conflict. It seems to be an intuitive solution to replace these concepts by their least common subsumer **Color**. Furthermore it is plausible to claim that all tomatoes have color without specifying this color precisely.

Unfortunately, not all types of inconsistency can be resolved automatically. For axioms of the form  $A \sqsubseteq D$  and  $A \sqsubseteq \neg D$  no other alternative can be found to guarantee consistency, except to removing one of the problematic axioms. Without appealing to external knowledge (such as the confidence rating of the axioms) one cannot decide which axiom need to be removed.

We want to generalize the examples discussed so far. If a concept  $X$  is defined in the TBox  $\mathcal{T}$  by the axioms  $X \sqsubseteq A$  and  $X \sqsubseteq B$  such that  $A$  conflicts with  $B$  in  $\mathcal{T}$ , then the following options can be distinguished:

1.  $A$  and  $B$  are disjoint concept descriptions having common subsumers (the tomato example given above):  
 The solution in this case is to replace the axioms  $X \sqsubseteq A, X \sqsubseteq B$  with the corresponding definitions taken from the set  $lcst_{\mathcal{T}}(A, B)$ .
2.  $A \in ax(\mathcal{T})$  and some definition  $D$  of  $A$  conflicts with  $B$  (the penguin example given above):  
 This case can be considered as the overgeneralization of  $A$ , because the concept  $X$  being subconcept of  $A$  represents an exception towards the definition  $D$ . The definition  $D$  must be revised as follows: a)  $D$  is replaced with its most specific superconcepts that do not conflict with  $B$ ; if there is no such concept then  $A \sqsubseteq D$  is just deleted; b) a new concept  $A'$  is added to the TBox as a subconcept of  $A$  and  $D$ ;  $A$  is replaced with  $A'$  in the definition of all its subconcepts except in the definition of  $X$ .
3.  $A, B \in ax(\mathcal{T})$ , a definition  $D_A$  of  $A$  conflicts with  $B$ , and a definition  $D_B$  of  $B$  conflicts with  $A$ :  
 In this case there is no automatic logical solution. Any of the definitions ( $D_A$  or  $D_B$ ) can be changed in the way described in the previous option (2) in order to achieve a consistent ontology. A confidence rating as suggested in (Haase et al. 2005) can be used for a selection of the axioms to be changed. For example, a function

$r_{conf} : Concepts \rightarrow \mathbb{R}$  can be used to assign a confidence rating to every concept. In this paper, we do not discuss this rating in more detail.

#### 4. Otherwise:

One of the conflicting axioms ( $X \sqsubseteq A$  or  $X \sqsubseteq B$ ) must be removed from the TBox. A confidence rating as suggested in (Haase et al. 2005) can be used for the selection of the relevant axiom to be removed.

In order to formalize the procedure described above we need to introduce the notion of a *conflict* in the following definition.

**Definition 4** For every TBox  $\mathcal{T}$ , two concept descriptions  $C_1$  and  $C_2$  conflict with each other in  $\mathcal{T}$  if and only if for every model  $\mathcal{M}$  of  $\mathcal{T}$ :

$$C_1^{\mathcal{M}} \cap C_2^{\mathcal{M}} = \emptyset$$

If the underlying logic allows conjunction (what is not true for our logic), then the definition above is equivalent to the following statement: two concept descriptions  $C_1$  and  $C_2$  conflict with each other in TBox  $\mathcal{T}$  if and only if

$$\mathcal{T} \models C_1 \sqcap C_2 \sqsubseteq \perp$$

The ontology adaptation algorithm described in the next section is a modification of the ideas that have been developed for the  $\mathcal{AL}\mathcal{E}$ -DL in (Ovchinnikova & Kühnberger 2006a), in order to model the less expressive logic defined in (Haase et al. 2005). This relatively weak logic seems to be appropriate for an implementation of an automatic ontology extension procedure.

## 4 Ontology Adaptation Algorithm

In this section, we describe the algorithm adapting an ontology to a new axiom. Before applying the adaptation algorithm *AdaptOnto* to a TBox, all equalities must be replaced by inclusions:

$$A_1 \equiv A_2 \longrightarrow A_1 \sqsubseteq A_2, A_2 \sqsubseteq A_1$$

The proposed algorithm *AdaptOnto* adapts a TBox  $\mathcal{T}$  to a new axiom  $X \sqsubseteq Y$ . The algorithm revises the definitions of the subconcepts of  $X$  because the introduction of a new definition of a concept  $X$  can have an influence on the semantics of its subconcepts.

If a concept  $A$  is a subconcept of  $X$ , then every two definitions  $D_1$  and  $D_2$  of  $A$  are checked whether conflicts occur. If  $D_1$  conflicts with  $D_2$  and the set of common subsumers for  $D_1$  and  $D_2$  is non-empty, then  $D_1$  and  $D_2$  in the definitions of  $A$  will be replaced with their least common subsumers.

A definition of  $A$  ( $D_1$  or  $D_2$ ) is overgeneralized (and denoted by  $D_o$ ) if it is axiomatized in  $\mathcal{T}$  and some of its definitions conflicts with the alternative definition of  $A^5$ . The definition of  $D_o$  will be changed. The alternative concept taken from the set  $\{D_1, D_2\}$  is denoted by  $D_c$  and is called contradicting definition.

The set  $C_c$  collects superconcepts of  $D_o$  that conflict with  $D_c$ . On the other hand, the set  $C_n$  denotes non-contradicting concepts. As explained in Sec. 3, the conflicting definitions of  $D_o$  (from the set  $C_c$ ) are removed from  $\mathcal{T}$  and assigned to the new concept  $A_N$  if they are minimal towards subsumption.  $A_N$  is declared to be a subconcept of  $D_o$ . Non-contradicting

<sup>5</sup>In the case of two overgeneralized concepts the axioms to be changed are chosen according to the confidence rating.

**Input:** a TBox  $\mathcal{T}$ , an axiom  $X \sqsubseteq Y$   
**Output:** an adapted TBox  $\mathcal{T}'$

```

 $\mathcal{T}' := \mathcal{T} \cup \{X \sqsubseteq Y\}$ 
FOR  $A \in \{A' \in ax(\mathcal{T}') \mid \mathcal{T}' \models A' \sqsubseteq X\}$ 
  FOR  $\{D_1, D_2\} : A \sqsubseteq D_1 \in \mathcal{T}' \wedge A \sqsubseteq D_2 \in \mathcal{T}' \wedge D_1 \neq D_2$ 
    IF  $D_1$  conflicts with  $D_2$  in  $\mathcal{T}'$  THEN
      IF  $lcss_{\mathcal{T}'}(D_1, D_2) \neq \emptyset$  THEN
         $\mathcal{T}' := \mathcal{T}' \setminus \{A \sqsubseteq D_1, A \sqsubseteq D_2\} \cup \{A \sqsubseteq C' \mid C' \in lcss_{\mathcal{T}'}(D_1, D_2)\}$ 
      ELSE
        IF  $\exists i, j \in \{1, 2\} : \exists D'_i : D_i \sqsubseteq D'_i \in \mathcal{T}' \wedge D'_i$  conflicts with  $D_j$  in  $\mathcal{T}'$  THEN
          IF  $\exists D'_j : D_{j \neq i} \sqsubseteq D'_j \in \mathcal{T}' \wedge D'_j$  conflicts with  $D_i$  in  $\mathcal{T}'$  THEN
             $D_o := D_{k \in \{1, 2\}} \text{ such that } r_{conf}(D_k) < r_{conf}(D_{m \in \{1, 2\}, m \neq k})$ 
             $D_c \in \{D_1, D_2\} \setminus \{D_o\}$ 
            ELSE  $D_o := D_i, D_c := D_j$ 
             $C_c := \{C \in subsumers_{\mathcal{T}'}(D_o) \mid C \text{ conflicts with } D_c \text{ in } \mathcal{T}'\}$ 
             $C_n := \{C \in subsumers_{\mathcal{T}'}(D_o) \mid C \text{ does not conflict with } D_c \text{ in } \mathcal{T}'\}$ 
             $\mathcal{T}' := \mathcal{T}' \setminus (\{D_o \sqsubseteq C \mid C \in C_c\} \cup \{Z \sqsubseteq D_o \mid Z \neq A\}) \cup$ 
               $\{D_o \sqsubseteq C \mid C \in C_n \wedge \forall C' \in C_n : \mathcal{T}' \models C' \sqsubseteq C \rightarrow C' \dot{=} C\} \cup$ 
               $\{A_N \sqsubseteq C \mid C \in C_c \wedge \forall C' \in C_c : \mathcal{T}' \models C' \sqsubseteq C \rightarrow C' \dot{=} C\} \cup$ 
               $\{A_N \sqsubseteq D_o\} \cup \{W \sqsubseteq A_N \mid W \neq A \wedge W \sqsubseteq D_o \in \mathcal{T}'\}$ 
          ELSE  $D_r := D_{k \in \{1, 2\}} \text{ such that } r_{conf}(D_k) < r_{conf}(D_{m \in \{1, 2\}, m \neq k})$ 
             $\mathcal{T}' := \mathcal{T}' \setminus \{A \sqsubseteq D_r\}$ 
        END FOR
      END FOR
    END FOR
  END FOR

```

Figure 1: The Algorithm *AdaptOnto* for adapting a TBox  $\mathcal{T}$  to a new axiom given by a concept description. The output is a new TBox  $\mathcal{T}'$  resolving overgeneralized definitions.

concepts from  $C_n$  that are minimal towards subsumption are added to the TBox as definitions of  $D_o$ . Previous subconcepts of  $D_o$  are declared to be subsumed by the new concept  $A_N$  that captures the original semantics of  $D_o$ .

If the overgeneralization can not be defined then the definition of  $A$  ( $D_1$  or  $D_2$ ) that has the least confidence rating is removed from the TBox.

## 5 Generalization Problem in Consistent Ontologies

### 5.1 The Problem

Even a consistent ontology can contain generalization errors. Automatic ontology learning procedures often rely on random facts that are extracted from external sources and are not observed by a human expert. Therefore the proper generalization of an automatically extracted ontology is rather accidental than intended.

We saw in the previous sections – particularly in Section 3 – that definitions of overgeneralized concepts can be detected and appropriately revised by the appearance of exceptions. On the other hand, undergeneralized concepts can sometimes be revised without additional information. Let us consider a simple example. Suppose that our ontology contains the facts:

- *Dogs are animals that breathe and drink water.*
- *Cats are animals that breathe and drink milk.*
- *Horses are animals that breathe and eat hay.*
- ... and so on for other animals.

Probably we would like to conclude from such a collection of facts that all animals can breathe and reformulate the ontology in the following way:

- *All animals breathe.*
- *Dogs are animals that drink water.*
- *Cats are animals that drink milk.*
- *Horses are animals that eat hay.*
- ... and so on.

Undergeneralization does not lead to inconsistencies of an ontology. But it is also not only a matter of design. Suppose that by a further extension of an ontology or by an instantiation it will be derived that a mole is an animal, but nothing else is known about it. We would like to infer that a mole can also breathe like other animals do. A proper generalization will help us to do so.

In practical applications, the undergeneralization problem is well-known and usually treated either semi-automatically or by analyzing external linguistic data. For example, in (Ceusters et al. 2003) it is shown how cross-lingual information can be used to detect undergeneralization in large ontologies.

In (Ovchinnikova & Kühnberger 2006b), the authors introduce an induction procedure designed for the regeneration of the axioms in an  $\mathcal{ALCN}$  description logic. In the following sections we adapt this procedure to the simple DL logic under consideration in order to make it applicable in automatic ontology learning.

### 5.2 Induction

The idea of an induction procedure proposed in (Ovchinnikova & Kühnberger 2006b) is simple. If all subconcepts of a concept  $C$  are also subconcepts of some other concept  $G$ , then  $C$  is likely to be a subconcept of  $G$ . Such generalizations are very similar to what is called upward inheritance in feature logic or programming: if all subtypes of a type  $C$  share the same feature, this feature should be inherited by  $C$ .

For practical applications it is useful to introduce certain heuristics and generalize a concept only if it has more than  $t$  many subconcepts with the same feature, where  $t$  is considered as an empirical parameter. In other words, statistical information can be used in order to decide whether a concept should be generalized or not.

After the execution of inductions it can happen that mistakes occur provided that more information is available. For example, if only bird species occur in a certain context and we apply induction it could

happen that we end up with an ontology where all animals can fly. All inductions can be checked and adapted during the next steps of the ontology extension by applying the procedure described in the previous sections.

In Subsection 5.3 and Subsection 5.4, we will consider the algorithmic details of induction: first, we discuss the algorithm *Regen* in Subsection 5.3. Second, we add some remarks concerning the prototype implementation of this algorithm in Subsection 5.4. The following definition specifies the induction procedure:

**Definition 5** For every TBox  $\mathcal{T}$ , for every concept name  $A$  the induction function  $Ind : TBox \times A \rightarrow TBox$  is defined as follows:

$$Leaves(A) := \{concept\ name\ L \mid \mathcal{T} \models L \sqsubseteq A \wedge \forall\ concept\ name\ B : L \sqsubseteq B \vee \mathcal{T} \not\models L \sqsubseteq B\}$$

$$LCS = lcss_{\mathcal{T}}(A_1, \dots, A_n) \text{ where } \{A_1, \dots, A_n\} = Leaves(A)$$

$$Ind(\mathcal{T}, A) = \mathcal{T} \cup \{A \sqsubseteq C \mid C \in LCS \wedge \forall C' \in LCS : \mathcal{T} \models C' \sqsubseteq C \rightarrow C' \sqsubseteq C\}$$

In Definition 5, the induction function  $Ind$  is defined for a TBox  $\mathcal{T}$  and a concept name  $A$ . The set  $Leaves$  collects all the subconcepts of  $A$  that have no further subconcepts. The set  $LCS$  represents the least common subsumers of the leaves of  $A$ . The induction function returns the TBox  $\mathcal{T}'$  extending  $\mathcal{T}$  with axioms that are minimal towards subsumption concepts taken from  $LCS$ .

In order to make the induction procedure more transparent, we give a simple example<sup>6</sup> of the application of the induction function to the TBox below and the concept **Person**:

TBox:

$$\{Woman \sqsubseteq Person \sqcap \forall hasSpouse.Man, \\ Man \sqsubseteq Person \sqcap \forall hasSpouse.Woman\}$$

Regeneralized TBox:

$$\{Person \sqsubseteq \forall hasSpouse.Person, \\ Woman \sqsubseteq Person \sqcap \forall hasSpouse.Man, \\ Man \sqsubseteq Person \sqcap \forall hasSpouse.Woman\}$$

In the example above, new information is added to the definition of the concept **Person**. From the definitions of its leaves **Man** and **Woman** it can be induced that every spouse of a person is also a person.

In the next section, we introduce the algorithm regeneralizing a TBox formalized in the DL logic under consideration.

### 5.3 Ontology Regeneralization

This section presents the regeneralization algorithm *Regen* inducing new definitions for concepts in the TBox  $\mathcal{T}$  on basis of the definitions of their subconcepts.

The algorithm proposed in Figure 2 tries to regeneralize every concept  $A$  in  $\mathcal{T}$ . The set of leaves  $Leaves(A)$  is computed for  $A$ . If the cardinality of this set exceeds the empirical parameter  $t$  (for  $t \in \mathbb{N}$ ), then the set of the least common subsumers  $LCS$  is computed relative to the concepts in  $Leaves(A)$ . If the set  $LCS$  is non-empty, then the concept  $A$  will be regeneralized as follows:

- Concept descriptions from the set  $LCS$  will be declared to subsume  $A$ ;

<sup>6</sup>In the following examples we use the symbol  $\sqcap$  for abbreviation:  $\{C \sqsubseteq D_1 \sqcap D_2\}$  stands for  $\{C \sqsubseteq D_1, C \sqsubseteq D_2\}$ .

**Input:** a TBox  $\mathcal{T}$ , a parameter  $t$   
**Output:** a regeneralized TBox  $\mathcal{T}'$

$\mathcal{T}' := \mathcal{T}$

**FOR** concept name  $A$

$$Leaves(A) := \{concept\ name\ L \mid \mathcal{T} \models L \sqsubseteq A \wedge \forall\ concept\ name\ B : L \sqsubseteq B \vee \mathcal{T} \not\models L \sqsubseteq B\}$$

**IF**  $|Leaves(A)| \geq t$  **THEN**

$$LCS = lcss_{\mathcal{T}}(A_1, \dots, A_n) \text{ where } \{A_1, \dots, A_n\} = Leaves(A)$$

**FOR**  $C \in LCS$

**IF**  $\mathcal{T}' \not\models A \sqsubseteq C$  **AND**

$$\forall C' \in LCS : \mathcal{T}' \models C' \sqsubseteq C \rightarrow C' \sqsubseteq C$$

**THEN**  $\mathcal{T}' := \mathcal{T}' \cup \{A \sqsubseteq C\}$

**FOR**  $A' : \mathcal{T}' \models A' \sqsubseteq A$

$$\mathcal{T}' := \mathcal{T}' \setminus \{A' \sqsubseteq D \mid \mathcal{T}' \models C \sqsubseteq D\}$$

**END FOR**

**END FOR**

**END FOR**

Figure 2: Algorithm *Regen* for the regeneralization of a TBox  $\mathcal{T}$ . *Regen* resolves undergeneralized concept definitions.

- The definitions of the subconcepts of  $A$  that subsume concepts from  $LCS$  will be removed (since they become redundant).

The algorithm *Regen* computes the construction specified in Definition 5 by computing the generalization of a given TBox. The following example illustrates the application of the *Regen* algorithm:

TBox:

$$\{Cat \sqsubseteq Animal \sqcap Breathing \sqcap \forall drink.Milk, \\ Dog \sqsubseteq Animal \sqcap Breathing \sqcap \forall drink.Water, \\ Man \sqsubseteq Human \sqcap Breathing, \\ Milk \sqsubseteq Liquid, Water \sqsubseteq Liquid\}$$

Regeneralized TBox:

$$\{Animal \sqsubseteq Breathing \sqcap \forall drink.Liquid, \\ Cat \sqsubseteq Animal \sqcap \forall drink.Milk, \\ Dog \sqsubseteq Animal \sqcap \forall drink.Water, \\ Man \sqsubseteq Human \sqcap Breathing, \\ Milk \sqsubseteq Liquid, Water \sqsubseteq Liquid\}$$

In the example above, new definition is generated for the concept **Animal**. The redundant information about breathing is removed from the definitions of **Cat** and **Dog**.

### 5.4 Prototype Implementation

The prototype implementation of the regeneralization procedure has been tested successfully on an example ontology automatically extracted with the tools developed in the framework of the ASADO project ([www.cogsci.uni-osnabrueck.de/~ASADO](http://www.cogsci.uni-osnabrueck.de/~ASADO)). The basis of the ASADO project were scanned documents (a majority of them taken from the aviation industry). In the project, standard tools mainly taken from computational linguistics research were applied to make these documents electronically available. Examples of such tools were an OCR-software, a tagger, or a state-of-the-art statistical parser. Based on the resulting electronically enriched documents an ontology for these documents was automatically extracted. For cross-evaluation purposes other document corpora were also used.

The ASADO ontology contains only the taxonomy, general relations and as logical connective conjunction. Here is an example of a concept definition in the RDF format:

```
<rdfs:Class rdfs:about="o:telephone-account">
  <rdfs:subClassOf rdfs:resource="o:account"/>
</rdfs:Class>
```

According to this definition the concept *telephone account* is a subconcept of the concept *account* having the property to be "telephone". In description logics this information can be formalized as follows:

$$\text{telephone-account} \sqsubseteq \text{attribute\_telephone} \sqcap \text{account}$$

The considered ASADO ontology contains approximately 3000 concepts. Among them we have found 20 undergeneralized concepts. It is clearly a matter of discussion if all regeneralizations proposed by the system are relevant for the thematic area under consideration. But the discovered undergeneralization cases can give the ontology engineer important hints of how to refine an ontology extracted automatically. Furthermore the undergeneralization cases can serve as an additional evaluation criteria of the ontology learning procedure.

Let us consider a quite simple and obvious example of an undergeneralized concept. The ASADO ontology contains several concepts using the concept *hong* in the definitions: *epson-hong-kong*, *epson-hong-kong-limited*, *epson-hong-kong-ltd*, *hong-kong-phone*, *hong-kong-user*. It is obvious that *hong* never occurs in the ontology without the concept *kong*. The system suggests to unite the concepts *hong* and *kong* in one concept.

In the near future, we plan to test the proposed algorithm on ontologies formalized in more expressive description logics. Nevertheless, the described prototype implementation already supports the claim that the presented induction procedure is relevant for ontology engineering.

## 6 Semantic Issues

Although we do not focus in detail on semantic issues in this paper, some remarks concerning the semantics of a regeneralized TBox are added in this subsection. We postpone a thorough discussion of this important issue to another paper.

Let us consider the regeneralization of just one concept *A* axiomatized in a given TBox. By slightly simplifying things, we get the following situation: If *A* is undergeneralized, then its definition will be extended by the *Regen* algorithm. In the case of an overgeneralization some concept descriptions will be removed from the definition of *A* by *AdaptOnto*. It is easy to show that in both cases only the semantics of *A* itself changes, whereas the semantics of its subconcepts remains unchanged. Obviously, the two processes change the semantics of *A* in two different directions. Whereas the induction procedure narrows the semantics of *A* by adding further constraints, the adaptation procedure *AdaptOnto* extends it by removing removing constraints on *A*. The following Fact is a direct consequence of the considerations so far.

**Fact 2** *For every TBox  $\mathcal{T}$ , for every interpretation  $\mathcal{I}$ , and for every axiom  $Ax$  the following two claims hold:*

- (i) *If  $\mathcal{I}$  models  $\mathcal{T}$ , then it holds  $\mathcal{I}$  models the adapted TBox  $\text{AdaptOnto}(\mathcal{T}, Ax)$ .*
- (ii) *If  $\mathcal{I}$  models the regeneralized TBox  $\text{Regen}(\mathcal{T})$ , then it holds  $\mathcal{I}$  models  $\mathcal{T}$ .*

Assuming that the function *subsumers*(*C*) introduced above computes all possible concept descriptions subsuming *C* in the DL under consideration (Fact 1), we claim that the induction procedure computes "the best" regeneralization for *A* relative to a chosen heuristics. In other words nothing more can be induced about *A* in the induction process relative to the chosen heuristic.<sup>7</sup>

**Fact 3** *For every TBox  $\mathcal{T}$ , for every concept name  $A$ , and for every concept description  $C$  the following equivalence holds:*

$$(\mathcal{T} \models A' \sqsubseteq A \rightarrow \mathcal{T} \models A' \sqsubseteq C) \Leftrightarrow \text{Ind}(\mathcal{T}, A) \models A \sqsubseteq C$$

In the adaptation procedure, the choice of the overgeneralized concept is based on a heuristics. Therefore it is impossible to prove strictly logically that the *AdaptOnto* algorithm guarantees the minimality of changes<sup>8</sup> of an inconsistent ontology. But once the overgeneralized concept has been chosen, then it is quite obvious that *AdaptOnto* removes a minimum of information from its definition. This claim follows directly from the definition of the *subsumers* function (Fact 1). For an overgeneralized concept *A* the sets  $C_c$  and  $C_n$  of conflicting and non-contradicting concept descriptions subsuming *A* are computed on the basis of the *subsumers* function. Therefore these sets are exhaustive in the description logic under consideration.

## 7 Conclusion and Future Work

In this paper, we presented an approach for dynamically resolving conflicts appearing by automatic ontology learning. We have adopted ideas firstly presented in (Ovchinnikova & Kühnberger 2006a) for the subset of description logics corresponding to the logic practically used in systems for ontology learning (Haase et al. 2005). The main contributions of this paper are the specification of an algorithm for ontology adaptation for the mentioned weak logic practically used in systems and the specification of an algorithm generalizing ontologies (based on the same logic). Whereas ontology adaptation is strongly connected to non-monotonicity and the problem of handling inconsistencies, generalizations are strongly related to inductive reasoning.

The two algorithms *AdaptOnto* and *Regen* can be embedded into an overall architecture amalgamating ontologies automatically if new information about the original TBox is available. Figure 3 represents diagrammatically how these two algorithms can be embedded into an ontology framework. The following list summarizes some important steps in this integration process.

- Starting with a given ontology *O* new information (represented by axioms) updates the logical description of *O*. The result is a new (updated) ontology  $O^+$ , i.e. the underlying TBox is updated by these new axioms.
- In a second step, a standard reasoning system checks the consistency of  $O^+$ .

<sup>7</sup>Recall that the heuristics is based on the chosen natural number *t* of leaves of *A* with the same feature.

<sup>8</sup>Due to a long history of non-monotonic reasoning in AI where minimality conditions play an important role, the formal definition of the notion of minimality in this context requires an additional investigation. In this section, we use this term informally just to give an idea of how to evaluate the proposed procedures from the semantical point of view.

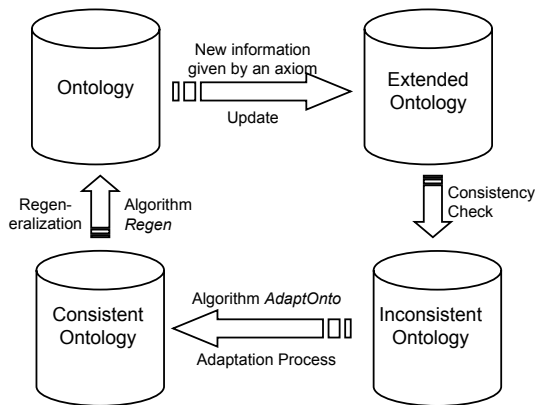


Figure 3: The diagrammatic representation of integrating the algorithms described in this paper into an ontology system. The resulting circle is permanently updating ontological knowledge with new information in a consistent way and keeps the ontology as compact as possible.

- If an inconsistency occurs, the presented algorithm *AdaptOnto* generates a new consistent ontology  $O_{con}^+$ . If no inconsistency occurs no adaptation is necessary.
- Finally the algorithm *Regen* rewrites the ontology  $O_{con}^+$  into an ontology representing knowledge in a more compact way by rewriting under-generalized concepts.
- The circle starts again by an update given of new axioms.

In the near future, we plan to develop a prototype implementation of the proposed architecture by combining the presented algorithms and test them on existing ontologies. It is of particular interest to see to what extent statistical information about the distribution and co-occurrence of concepts in texts can help to improve the adaptation procedure for making it more adequate to human intuition. Similarly, generalizations defined on ontologies are also dependent on statistically relevant information as can be seen in Figure 2 where generalization is only possible if a concept has more than  $t$  subsumers with the same feature.

Last but not least, an important theoretical issue concerns the complexity of the proposed algorithms. In the future, we plan to show characterization results specifying the complexities classes of the algorithms *AdaptOnto* and *Regen*, in order to prove theoretically the practical relevance and the tractability of the proposed architecture.

### Acknowledgment

This research was partially supported by the grant MO 386/3-4, a subproject of the collaborative research unit FOR 437 sponsored by the German Research Foundation (DFG).

### References

Baader, F., Lutz, C., Milićić, M., Sattler, U. & Wolter, F. (2005), Integrating Description Logics and Action Formalisms: First Results. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI'05)*, AAAI Press (2005).

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P. (eds.) (2003), *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F. & Küsters, R. (2006), Non-Standard Inferences in Description Logics: The Story So Far. *International Mathematical Series*, volume 4, *Mathematical Problems from Applied Logic. New Logics for the XXIst Century*.

Baader, F. & Sattler, U. (2001), An overview of tableau algorithms for description logics, *Studia Logica*, 69:5–40.

Baral, C. (2006), *Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets*. Cambridge University Press.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001), The Semantic Web – A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May 17, 2001, available on the world wide web (10th of March, 2006): [http://www.sciam.com/print\\_version.cfm?articleID=0004814410D21C7084A9809EC588EF21](http://www.sciam.com/print_version.cfm?articleID=0004814410D21C7084A9809EC588EF21).

Bibel, W., Hölldobler, S. & Schaub, T. (1993), *Wissensrepräsentation und Inferenz. Eine grundlegende Einführung*, Vieweg, Braunschweig.

Ceusters, W., Desimpel, I., Smith, B. & Schulz, S. (2003), Using Cross-Lingual Information to Cope with Underspecification in Formal Ontologies, In *Studies in Health Technology and Informatics*, 391–396.

Cohen, W., Borgida, A. & Hirsh, H. (1993), Computing Least Common Subsumers in Description Logics, In *Proc. of the 10th Nat.Conf. on Artificial Intelligence (AAAI-92)*. AAAI Press, 754–761.

Fanizzi, N., Ferilli, S., Iannone, L., Palmisano, I. & Semeraro, G. (2005), Downward Refinement in the ALN Description Logic. In: Masumi Ishikawa, Shuji Hashimoto, Marcin Paprzycki, Emilia Barakova, Kaori Yoshida, Mario Köppen, David W. Corne and Ajith Abraham (Eds.), *Hybrid Intelligent Systems (HIS'04)*, 68–73.

Flouris, G., Plexousakis, D. & Antoniou, G. (2005), Updating Description Logics using the AGM Theory. In *Proc. of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning*.

Ghilardi, S., Lutz, C. & Wolter, F. (2006), Did I damage my ontology: A Case for Conservative Extensions of Description Logics. In *Proc. of Principles of Knowledge Representation and Reasoning 2006 (KR06)* (to appear).

Gómez-Pérez, A. & Manzano-Macho, D. (2003), A survey of ontology learning methods and techniques, <http://ontoweb.aifb.uni-karlsruhe.de/Members/ruben/Deliverable>

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. & Sure, Y. (2005), A Framework for Handling Inconsistency in Changing Ontologies In *Proc. of the Fourth International Semantic Web Conference (ISWC2005)*, v. 3729, pp. 353–367. Springer (2005).

Heymans, S. & Vermeir, D. (2002), A Defeasible Ontology Language, In Robert Meersman and Zahir Tari et al., editors, *Confederated International Conferences: CoopIS, DOA and ODBASE 2002*.

- Horrocks, I. (1998), Using an expressive description logic: FaCT or fiction? In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, 636–645. Morgan Kaufmann, San Francisco, California.
- Katz, Y. & Parsia, B. (2005), OWL: Experiences and Directions, Galway Ireland, online available at: <http://www.mindswap.org/2005/OWLWorkshop/sub7.pdf>.
- Lifschitz, V. (1994), Circumscription. In: *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 3, 297–352, Oxford University Press.
- Motik, B., Sattler, U. & Studer, R. (2004), Query Answering for OWL-DL with Rules. In *Proc. of ISWC 2004*, LNCS 3298, 549–563, Springer (2004).
- Ovchinnikova, E. & Kühnberger, K. (2006a), Adaptive  $\mathcal{ALC}$ -TBox for Extending Terminological Knowledge, to appear in: A. Sattar and B. H. Kang (eds.): AI 2006, Proceedings of the 19th ACS Australian Joint Conference on Artificial Intelligence, LNAI 4304 (Lecture Notes in Artificial Intelligence), Springer, pp. 1111–1115.
- Ovchinnikova, E. & Kühnberger, K. (2006b), The Undergeneralization Problem in Ontology Design. In preparation.
- Reiter, R. (1980), A logic for default reasoning. *Artificial Intelligence* 13:81–132.
- OWL Web Ontology Language (2004), Overview. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/owl-features/>.
- Walter, C. (1985), A Mechanical Solution of Schubert's Steamroller by Many-Sorted Resolution. *Artificial Intelligence* 26:217–224.

# Metonymic and Holonymic Roles and Emergent Properties in the SNOMED CT Ontology

Jon Patrick

Centre for Health Informatics Research & Development

School of Information Technologies

University of Sydney

Sydney Australia

jonpat@it.usyd.edu.au

## Abstract

This paper discusses the manner in which SNOMED CT (SCT) has confused the metonymic role of some class labels as holonyms and has inappropriately assigned property inheritance down a holonymic chain due to its transitivity. The notion of emergent properties is introduced as the only form of property that can exist on a holonym and its use in a hypernymic inheritance hierarchy is discussed. The consequences of this modelling approach for SCT are discussed and the use of metonymic substitution for holonyms at the point of care is presented as a source of confusion for causing the modelling of inheritance in holonymic hierarchies for clinical care. The mathematical modelling of the metonymic substitution is discussed but left as future work.<sup>1</sup>

*Keywords:* ontology, metonymy, holonymy, emergent properties

## 1. Introduction

SNOMED CT (SCT) is a very large scale ontology used for the description of certain classes of medical and health community knowledge. We are interested in its value for the functional purposes of computation at the point of care and the issues in maintaining and delivering it for those purposes. SCT is maintained by the College of American Pathologists (CAP) and not necessarily fit for purpose for all health and medical domains. Although very wide ranging with over 360,000 concepts and 1.2 million relations it lacks some fundamental facets of a full ontology. It is not an ontology in the strictest sense of the word as it contains a great deal of knowledge that is compositions of fundamental elements within the system. That is, it is possible to express the same collection of ideas with multiple strategies either as a single concept (if sanctioned) or as a group of individual concepts, for example, fracture of neck of femur (concept 591300) vs {fracture (concept 72704001) | neck of femur (concept 29627003)}. Also, classes do not have attributes but rather they are expressed through relations between “characteristics” and “concepts”. SCT although maintained using the Protégé logic engine is released as

a set of 3 files prepared as comma separated values (CSV) and so fails to come with a logic engine to preserve its logical structure and constraints. Lastly the modelling has been performed over more than 40 years and in the late 1990s it was merged with another large ontology the Reed codes developed in the UK. This has blended together two different modelling paradigms so that SCT has a significant admixture of concepts and relations which have lead to redundancy, rival explanatory theories, and confused modelling.

Ontologies to be useful for computation for practical tasks need to be precise in two ways; firstly, they need to match as closely as possible our understanding of the natural world we wish to deal in. Ontologies need to be accurate and closely follow human understanding otherwise they create confusion in their design and their use. This means the ontology needs to be constructed with very close attention to the meanings of the terms used in it. Hence the names used in the ontology need to represent closely the understanding we have of the real world.

Secondly the variety of linguistic usage of those terms needs to be explored to uncover the diversity of semantic roles of the terms and ensure that only those roles that are useful are included in the ontological modelling, and to remove ambiguity in the use of those roles. Precision of definition is particularly important in establishing the relationships between elements and identifying the fundamental atomic elements and how they combine together systematically to make more complex semantic concepts.

Ontologies also consist of abstractions with each level of the ontology being more abstract as one moves up the hierarchy. This structure allows us to talk about the world at the different levels of abstraction. In using an ontology for computation there are two basic forms of abstraction available, aggregation and generalisation. Generalisation hierarchies are used throughout SCT as the basic mechanisms for relating content. Aggregation hierarchies on the other hand have not been used properly but rather transposed so that they appear like generalisation hierarchies. We investigate this replacement for a proper aggregation hierarchy and argue that it comes from a misunderstanding of both the linguistic use of terminology at the point of clinical care and the logic arguments developed for its justification. In particular in SCT the holonymic (or hypernym) role of an aggregating concept is used as a source of inheritance which is clearly incorrect. Our explanation for this SCT modelling strategy is that the role of such a holonym has undergone the process of metonymic substitution, which is substitution of the authentic word

<sup>1</sup> Copyright (c) 2006, Australian Computer Society, Inc. This paper appeared at *The Australasian Ontology Workshop (AOW 2006) Hobart, Tasmania, Australia*. Conferences in Research and Practice in Information Technology, Vol. 72. Mehmet Orgun & Thomas Meyer, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

for one that serves as a metaphor for the original. The assignment of attributes and relations of the meronymic (sub-part) members of the holonym (super-part) to be one of the holonym itself can at best be called metonymic inheritance. Importantly, if it is allowed to operate at all, it must operate from the bottom up, that is the attributes move from the part to the whole, that is, in reverse to what we normally think of as the direction of inheritance, from the top down.

## 2. The Meaning of Aggregation and Generalisation.

The discussion of Aggregation and Generalisation is drawn from the database literature so as to emphasize their computational aspects. Smith & Smith make the basic statement “aggregation and generalization are independent activities” (Smith & Smith, 1977, p116). Although they are two fundamental means of organizing the knowledge about the relationships between concepts or entities, they specify knowledge that is independent of each other. Generalisation is a hierarchy of classes and subclasses in a subtype relationship, usually referenced as IS\_A. Aggregation is a hierarchy of components sometimes called a PART\_OF or PART-WHOLE relationship. Generalisation and Aggregation are *relationship types* And hence cover many instances of different hierarchies of these *types* found in natural language. The similarities and differences between these two types of relationships are fundamental to understanding how ontologies are organised and the limitations to their processing.

Generalisation is a hierarchy of classes and subclasses where each subclass has a transitive relationship with its parents and children, that is, properties are carried from subclass to subclass down the hierarchy, but also the properties introduced into a subclass that are separate to the inherited properties are *not inherited* back up the hierarchy.

Aggregation is the idea that individual types of parts are brought together into a hierarchy to make a whole, such as in the assembly of a motor car. Hence, a sparkplug is part of a motor, is part of a car. Importantly there is a transitive relationship between the parts, that is, the sparkplug is part of the car. However, there are no inherited properties, so red wheels do not make the car a red car, nor does a red car make the wheels red. Furthermore, the relation is *irreflexive*, that is, an entity cannot have a part-of relationship with itself, so a car is not part of a car.

Algebraically the transitive property can be expressed as:

If  $aRb$ ,  $bRc$ , then  $aRc$ , where  $R$  is a defined relationship such as IS\_A\_SUBCLASS\_OF, or IS\_PART\_OF.

In this example the relation IS\_A\_SUBCLASS\_OF is defined to be of the relationship type Generalisation, while the relation IS\_PART\_OF is relationship type Aggregation. So whilst both relations are transitive and we can apply that algebraic function to their hierarchies, the meaning content of the relationship names are different and the relationship types are different.

## 3. Linguistic Terminology

In linguistics the Generalisation relationship is known as a *hypernym-hyponym* relationship. Aggregation is known as the *holonym-meronym* relationship. A further linguistic concept, *metonym*, is needed to complete the explanation of the problems with the SCT structures. A *metonym* is a word used in place of the correct word because it is associated with it in some way, so the word “dish” can be used in place of “roast beef” when talking about the evening meal. This is a very common linguistic strategy and in one of its extreme forms it is called a *euphemism*.

## 4. Processing with Ontologies

In SCT there are 19 generalisation hierarchies such as Organism, Body Structure, Clinical Finding, etc. The concept “part-of” is defined in the Attribute generalisation hierarchy. It has a means of representing part-whole concepts by using SEP structures based on the ideas created in Schulz, Romaker & Hahn (1998) which in turn claims its heritage in the GALEN project Rector, Bechofer, Goble, Horrocks, Nowlan & Solomon (2000) and Rogers, J. & Rector (2000). These papers present the problem of part-whole reasoning as a problem of differentiating it from reasoning for subsumption. They point out that some systems, MESH for example, treat reasoning for generalisation relations and part-whole relations the same way producing inconsistencies such as *blood* being a hypernym for both *foetal blood* (a type of blood) and *blood plasma* (a part of blood). They point out that a different logic is needed to compute over a part-whole hierarchy which consists of the features *transitivity* and *part-whole specialisation*.

## 5. Transitivity

A key issue for Schulz et al. is the notion of whether the meronymic relationship (IS\_PART\_OF) is transitive as in the relationship IS\_SUBCLASS\_OF (or IS\_A). The example provided is

appendix IS\_PART\_OF colon  
colon IS\_PART\_OF intestine

as this is manifestly true for common sense knowledge then it must be true that

appendix IS\_PART\_OF intestine

hence transitivity is demonstrated.

However Schulz et al. do not deal with the issue of inheritance of properties in either a modelling or a linguistic sense for a Generalisation hierarchy. Importantly later in their paper they assume that given the transitivity of both relationship types (that is generalisation and aggregation) that any property is also inherited between types. This is manifestly untrue. Take the example of a motor car and its wheels. A blue motor car does not necessarily have blue wheels and a car with a set of racing wheels is not necessarily a racing car. Hence the characteristics of each part are not necessarily inherited either up or down a holonymic hierarchy, in fact that is the point of differentiation between the two types of relationships, hyponyms have inheritance of characteristics and meronyms don't have inheritance. Furthermore red upholstery in a red car are



not instances of the same red characteristic, they are merely coincidental facts about the car. While it is possible to assign algebraic symbols for all these phenomena and then to perform computational manipulation that doesn't mean that any computation is sensible. Moderation of what is sensible to compute must lie with the meaning intention of the symbolic representation.

The misconception of the sharing of inheritance due to the equivalence of transitivity between the two relationship types needs to be investigated both from a linguistic perspective and from an historical perspective to understand how their interpretation arose.

## 6. The Usefulness of the “Part-Whole Specialisation”

Schulz et al. take holonymic *specialisation* from Horrocks et al. (1996) and call it *part-whole specialisation*. This phrase creates a warning sign of future difficulties. Specialisation is the inverse of generalisation, and generalisation-specialisation are distinctly different to part-whole relationship, so the term *part-whole specialisation* appears to be a contradiction in terms.

Schulz et al. produce a holonymic example with “shaft of femur” is PART\_OF “femur” (Fig 1), in a parallel structure they produce the pair “fracture of shaft of femur” IS\_A “fracture of femur” as a specialisation. They then assert that not only are the structures related by the relationship “FRACTURE\_OF” (relations R1 and R3) but the hyponym “fracture of shaft of femur” is related to the holonym “femur” by the same relation, R2, that is “fracture of shaft of femur” is a FRACTURE\_OF “femur”.

There are a number of difficulties with the configuration in Fig 1 and the line of argument presented in Schulz et al. The logical deduction follows these lines:

**Premise:** the “femur” and “shaft of femur” have a holonym-meronym relationship,

**Deduction:**

**If** the hyponym to meronym have a relationship, in this case FRACTURE\_OF,

**then** the hyponym to holonym have the same relationship (R2),

**and thereby** the meronym is a *specialisation* of the holonym.

**furthermore** by asserting the premise that the hypernym has the same relationship to the holonym(R3) as the hyponym has with the meronym (R1), that a hypernym-hyponym relationship is thereby verified.

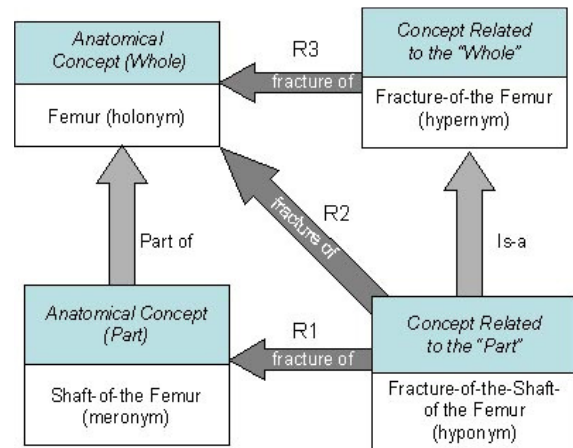


Fig 1. Part-Whole Specialisation as represented by Schulz et al. R1, R2, R3 are the relationship “fracture of”. R2 is a relationship from the hyponym to the holonym.

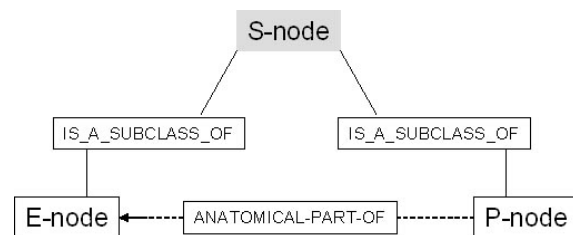


Fig 2. Structure of S-node for modelling Part-Whole Specialisation (after Schulz et al).

A general form of this argument is embodied in the SEP structure as used to model an ANATOMICAL-PART-OF relation for the relationship between physical parts of an organism (*op cit*). It consists of three nodes: S, E and P nodes. S is a structure node which subsumes the E-node and the P-node and there is an ANATOMICAL-PART-OF relation from the P-node to the E-node (fig 2).

The E-node denotes the *whole* anatomical part being modelled and the P-node denotes any part of the E-node entity. The P-node is then the parent of the same structure repeated down a generalisation hierarchy, so:

the intestine-structure (S-node) has an intestine (E-node) and intestine-part (P-node) which has a colon-structure (S-node) which has a colon (E-node) and colon-part (P-node) which has a caecum-structure (S-node) which has a caecum (E-node) and caecum-part (P-node) which has an appendix-structure (S-node) which has an appendix (E-node).

Schulz et al justify this structure in saying “by introducing a special data structure for part-whole encoding we build up specifically structured IS-A hierarchies which support the emulation of inferences typical of transitive PART-OF relationships. The same formalism allows for conditioned part-whole specialization”. They have in effect created a data structure that masks the PART-OF relationship between anatomical elements on the argument that they don't have a transitive relationship (whereas they are

transitive) so as to allow for inheritance (which is not permissible).

However they concede that the inheritance of the specialisation relationship down the holonymic hierarchy is not always found to be consistent with medical knowledge. The cited contrary example is:

**While**

Perforation of Appendix IS\_A\_SUBCLASS\_OF Intestinal Perforation **is true,**

**it is not true that,**

Inflammation of Appendix (Appendicitis) IS\_A\_SUBCLASS\_OF Intestinal inflammation (Enteritis).

Their arguments for introducing specialisation into the Aggregation hierarchy are not convincing and do not account for a number of matters and introduce the possibility of significant errors of reasoning. It appears that two other motivations for this configuration exist:

that “part-of” is meant to provide for referencing a portion of a body part that has no individual label, for example where the portion of the lung (say 35%) is removed (Spackman, personal communication), and also,

the logic engine used to verify the hierarchies only provided for transitivity verification through the implication operator in the generalisation hierarchy and not through the aggregation hierarchy (Schulz, personal communication).

Both of these explanations provide some reasoning for the structure and also create a motivation for wanting to review the solution to get a representation closer to a satisfactory ontological form. Considering the problem from another perspective we can say, firstly, a “fracture to the shaft of femur” is not a type of “shaft of a femur”, that is, it is not a subclass of either a “femur” or “shaft of femur”. To make an analogy, a broken car is not a broken wheel and a broken wheel is only a broken car through a metonymic substitution where the correct word, “wheel” is substituted by a word associated with it, “car” in this case. If the “shaft of the femur” is broken then it is not the “femur” that is broken but rather the “shaft” and on the basis of non-inheritance between holonym and meronym the one is not the other. The key confusion occurs because the term “femur” which is a holonym in the hierarchy is used as well as a metonym, that is, as a substitute for the correct word by virtue of association with it.

One weakness in the argument lies in the nature of the modelling of the physical world. A fracture to anything is not an entity of itself but a CHANGE OF STATE of an entity and is better modelled that way. No doubt clinicians at the point of care speak about “a fracture to the shaft of the femur” without paying attention to the linguistic structure. The “fracture to” is really serving as a descriptor of the entity as in “a fractured shaft of the femur” where the participle “fractured” is clearly serving in the epithet role.

Turning to the logic of the argument for the part-whole specialisation, there is no credible basis for any steps in the deduction. The argument hinges on the

notion that “specialisation” exists because a hypernym has the same relationship to each of the members of the holonym-meronym pair, that is R1 and R2 are the same in Fig 1. However the argument relies on deducing that the relationship between hyponym and holonym is created by virtue of the relationship of hyponym to meronym, then meronym to holonym, that is, because R1 and R2 are true and there is a transitive relationship between the hypernym and hyponym then R3 is true – this is a specious argument.

Our analysis questions the applicability of the SEP model to a holonymic structure and demonstrates that it is not applicable at least in the scope of the examples used so far. The explanation for the justification used by Schulz et al. is shown to be flawed because they shift from defining a term in a holonymic role and then change the role in usage of the holonym to a metonym.

Schulz et al. use another example to discuss the generalisation of their SEP model where their process is also manifest. They assert that if INTESTINE-STRUCTURE is an S-node then a PERFORATION-OF-APPENDIX is also a PERFORATION-OF-INTESTINE as Appendix IS-A-PART-OF Intestine. This is the same faulty reasoning. It is true that if the holonymic tree is only defined as far as “intestine” then there is no such thing as a PERFORATION-OF-APPENDIX and any perforation of anywhere in the Intestine is a PERFORATION-OF-INTESTINE. However if the holonymic tree is further described in all of its parts then the word “intestine” can only be used in a metonymic role and as such does not have any of the characteristics of the real perforated part, that is, it is acting as a surrogate. Taking the example of a car-motor-spark\_plug meronymy, if the spark plug is cracked then the motor won’t work and the car won’t work, but neither the motor nor the car is cracked. Both the motor and the car have emergent properties, that is properties they have by virtue of being a “whole” that none of their parts have, e.g. a motor runs and a car runs or works or can be driven (in some contexts). In English we have words that differentiate between a cracked spark\_plug and a motor that doesn’t work and car that’s broken down that help create the separation of metonymy from meronymy - we would not (normally) say “My car is cracked” or “My motor is cracked”.

This argument does not discount the legitimate use of the PERFORATION-OF-INTESTINE when the meronymic component is unknown as might well happen at point of care. But that situation is as above where the word is being used in the role of a metonym and not as a holonym. This issue leads into the further example used by Schulz et al. where they have at the top of a diagram an example of the S-node Intestine structure which descends down an IS\_A hierarchy successively to colon, caecum, and appendix. They assert that “enteritis” has the relation INFLAMMATION-OF with “intestine”, and “appendicitis” has the relation INFLAMMATION-OF with “appendix” yet “appendicitis” cannot be classified as a type of “enteritis”, even though the “appendix” is a PART\_OF the “intestine”. Importantly they argue that each of these are attached to E-nodes and thereby do not automatically have a subsumption relationship as is

true in real life, whereas they claim the Perforation example is a true subsumption and is attached to the **P**-nodes.

In such a structure if one were to assign to the intestines a property of 8metres long, as might well happen in an operational information system, we would have a colon, caecum and appendix all 8 metres and a very crowded pelvic zone. In terms of diseases in this diagram Enteritis is attached to the Intestine and Appendicitis to the Appendix, both attached by the relationship INFLAMMATION-OF. Apart from the convention that enteritis is a condition of the small intestine and colitis of the large intestine, the naming convention does not match any sensible characterisation of the real world. Furthermore the weakness in the SEP model is exposed. If the correct level of granularity is a match of the disease instance level to the anatomical location, that is Appendicitis IS\_AN\_INFLAMMATION\_OF Appendix, then Appendicitis is a specialisation of the class INFLAMMATIONS\_OF\_THE

\_INTESTINAL\_TRACT, that is the correct level of generalisation of the terminology that must match the correct level of meronymy, but nevertheless importantly they are independent of each other. So to say a patient has “intestinal inflammation” is holonomically correct, but to say they have an “inflamed small intestine” is a metonymic use of “inflamed”, the meronymically correct expression being “enteritis of the small intestine”. Hence on Schulz et al.’s diagram the correct entry for Enteritis should be Intestinal\_inflammation a sub-class of inflammations that has members {enteritis, colitis, appendicitis} each of which are DISEASES\_OF the anatomical components of the Intestine (small intestine, large intestine and appendix) respectively.

## 7. A Confluence of Aggregation and Generalisation of Hierarchies

A useful example to analyse the confluence of holonymy and metonymy hierarchies is the case of the “intestine” where each part supposedly inherits the characteristic of “hollow structure”.

The principle parts of the intestines are the small intestine (duodenum, jejunum and ileum), and the large intestine (appendix, caecum, colon and rectum). At the common sense level each of these structures is indeed “hollow”, but more relevantly open at both ends and so rightly can be assigned that characteristic, although the appendix is open only at one end. However common sense should be the last refuge for constructing descriptions in building an ontology. Importantly the intestine components are contiguous, that is they abut one another and they are open at each end and so form a continuum, which might lead one to presume that the intestines really do have the attributes of a hollow structure and open at both ends, that each component inherits. However a closer inspection shows a different story. We can say that the appendix does have a hollow structure, but is not open at both ends and so would that then negate the assertion that the intestines have a hollow structure open at both ends? The answer is clearly no, as the intestines would continue to perform

as expected allowing food to pass through. So in fact the components do not inherit the characteristic from their holonym superior. Rather the “inheritance” is in the other direction and is not a true inheritance at all. The intestine has a hollow structure by virtue of its components all having the same structure (hollow and open) AND because they exist in a contiguous configuration. The meaning of the expression “the intestine has hollow structure” is a shorthand for the saying each of the components have a hollow structure open at both ends and they are contiguous, and is in fact a metonymic use of the concept “intestine” not a holonymic use.

A more transparent example can be found from common experience. If I say “I have a red car” then the most common interpretation is the paint colour of the car exterior is mostly red. It is most unlikely that a listener would interpret that the wheels are red. On the other hand if I say “my car is completely red” then the listener is likely to assume the car body and interior decoration are red, with greater uncertainty as to whether the wheels are red or not. The listener would certainly not expect the engine or the underbody to be painted red. So in the car example the components are red by virtue of a truth statement about each component individually, but the red of each component is not inherited from the holonym (whole car), but rather they are each individually red. This interpretation is more easily recognised with the car example as there is no ready metonymic equivalent usage available in English as there is in the case of “intestines”. Hence it is much more difficult to construct the fallacy of inheritance by holonym-meronym transitivity.

Although ontological modelling does not allow inheritance in a formal sense we can define the concept of *metonymic inheritance*, which is inheritance by virtue of metonymic substitution of the whole for the parts, and is thereby a linguistic usage phenomena rather than an intrinsic ontological construct. In the intestines example this form of inheritance is from all of the parts to the holonym but we cannot rule out the possibility of metonymic inheritance from a single meronym rather than all parts of the holonym. It remains to be determined when such a construct would be useful and what sort of an algebra might be suitable for describing operations on it.

The modelling consequences for this metonymic interpretation is that devices such as SEP are not useful in modelling as they create redundant features and distort the picture as to what is identical but not inherited, that is replicated, from what is identical because it is inherited, that is duplicated.

The argument presented herein leaves an open question as to what might be an intrinsic property of a holonym given that they cannot be inherited. The only possibility can be an *emergent property*, that is a property that exists because of the intrinsic wholeness of the holonym and not a property of any of its meronyms. Such properties are common, for example, the human body is mobile or can move under its own power as we understand with a motor car. Emergent properties cannot be inherited by their nature and hence cannot be inherited down an aggregation hierarchy but

may well be inherited throughout a generalisation hierarchy. They exist because there is some aspect of the whole which is greater than the sum of the parts and so are attached to holonyms. Hence the confluence of aggregation and generalisation hierarchies occurs at the point of a holonym which has emergent properties. The emergent properties carry down the inheritance structure of the generalisation hierarchy but not down the aggregation hierarchy.

It remains a task for this model to be used to frame an algebra so as to provide a formal computational mechanism for applying metonymic substitution for logical deduction along the lines of Padgham & Lambrix's (1994) work for holonym-meronym relationships. It is not immediately evident how such an algebra would function but it must provide for any rank shift in holonymy to trigger the same in any attached relationships, and for emergent properties to be definable but inheritable only in a generalisation hierarchy. Under such circumstances it is also likely that a rank shift of the attributes has to be made for accurate modelling, that is movement from Appendix to Intestinal Tract requires the descriptor of disease to make a likewise rank shift up an aggregative terminology, for example from Appendicitis to Inflammation. In this manner we avoid saying Appendicitis is an inflammation of the intestine, but rather aggregate all the types of inflammation of intestinal parts under a single rubric "Inflammation of the Intestine" the members of which are {Appendicitis, Colitis, Enteritis...}. Then their use as a descriptor of a disease of the intestine can be identified as a metonymic use and the deduction that the true referent is a meronym, perhaps some distance down an aggregation hierarchy, can be computed reliably, and lead subsequently to semantically sensible responses and prompts from an operational decision support system.

## 8. Meronymy and Metonymy at Point of Care

The difference between using words in a holonymic role and metonymic is important and context dependent. If a patient attends a clinic and reports "I have a broken leg" the clinician will say "we need to X-ray that and find out what is broken". On the return of the X-ray the clinician will say to the patient "the femur is broken" however to an attending surgeon they will say "the shaft of the femur is broken" this being an important distinction to the "neck of femur being broken" as they have two entirely different treatment requirements. Once that definition is established they can readily revert to talking about the "fractured femur" without any risk of misunderstanding because they know they are really talking about the "shaft of the femur". The expression "femur" is thereby being used as a metonym and not a holonym.

The question remains as to the level of granularity a meronymic hierarchy needs to be referenced in the point of care. The answer is resolved clearly by the detail at which the care regime is determined. If the difference in care is defined by differentiating between "shaft of femur" and "neck of femur" then the clinician(s) need to establish their reference

terminology to that level, and use of the terminology at the level is literal and meronymic, however references to terms above that meronymic level are not holonymic but metonymic, unless otherwise asserted within the context, and hence the metaphorical role of the metonym needs to be accounted for.

None of these arguments are meant to deny the need for clinicians to talk at various levels of abstraction for different listener communities. Rather the arguments are intended to direct the need for the ontology to catch all aspects of the language variations in systematic ways that exploit not only our knowledge of medicine but our knowledge of language and our tools of logic (and sometimes statistics).

## 9. Conclusions

We have argued that the SEP model creates contradictions to common sense knowledge. What has been provided as a computational function in Cyc has been taken as a generally applicable principle in GALEN and then applied without consideration of its semantic constraints.

A consequence of this analysis is that the SEP model is not needed to model the transitivity relationship in a holonymic-meronymic hierarchy as it is intrinsically transitive and such usage sets a dangerous precedent of allowing attribute inheritance in a holonymic-meronymic hierarchy which can readily lead to nonsensical assertions.

The identification of the appropriate level in the holonymic-meronymic hierarchy for clinical reference needs to be at the point at which clinical care is determined, and references to holonyms at that point are literal. References about that point are to be considered metonymic unless otherwise resolved.

The confluence of generalisation and aggregation hierarchies occurs at the point at which a holonym has an identifiable emergent property that is inheritable by its sub class members in its generalisation hierarchy. Any application of properties of meronyms to holonyms can only be in the use of the holonym name in a metonymic role and hence the property is not a true property of the holonym but rather a pseudo property or a "metonymic property".

## 10. References

- Smith, J.M. & Smith, D.C.P. Database Abstractions: Aggregation and Generalisation, ACM Transactions on Database Systems, Vol.2, No. 2, pp105-133, 1977.
- Schulz, S., Romacker, M. & Hahn, U. Part-Whole Reasoning in Medical Ontologies Revisited – Introducing SEP Triplets into Classification-Based Description Logics. Proc AMIA. 830-4, 1998.
- Rector, A.L., Bechofer, S., Goble, C.a., Horrocks, I., Nowlan, W.A. & Solomon, W.D. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*. 9:139-171, 1997.
- Rogers, J. & Rector, A. GALEN's model of parts and wholes: Experience and Comparisons. In Overhage, J.M. (Ed.) Proc American Medical Informatics Assoc Symp. 714-8, 2000.

- Horrocks, I., Rector, A. & Goble, C. A description logic based schema for the classification of medical data. KRDB'96 –Proc of the 3<sup>rd</sup> Workshop on Knowledge Representation Meets Databases, pp 24-28, 1996.
- Padgham, L. & Lambrix, P. A framework for part-of hierarchies in terminological logics. J. Doyle, E. Sandewall and P. Tarasso (Eds.). Principles of Knowledge Representation and Reasoning. Proceedings of the 4th international Conference (KR'94), Bonn, Germany, pp485-496, 1994.



# Towards Semantic Interoperability in Healthcare: Ontology Mapping from SNOMED-CT to HL7 version 3

Amanda Ryan

School of Economics and Information Systems  
The University of Wollongong,  
Northfields Avenue, Wollongong, NSW, 2522,  
Australia  
Email: ajr883@uow.edu.au

## Abstract

One of the most successful Healthcare Information Models is version 2 of the Health Level 7 (HL7) standard. However, this standard has various problems, mainly its lack of semantic interoperability. This shortfall was addressed in HL7 Version 3, a newer standard which has been designed to solve this problem. Total semantic interoperability cannot be achieved without defined terminology, and to this end the use of the Systemised Nomenclature of Medicine - Clinical Terms (SNOMED-CT) is proposed. The difficulty arrives when deciding how to integrate the information model and the terminology. The line between where one ends and the other begins is often indistinct. This paper describes a proposal for normalising the two using ontology mapping and basing HL7 message models on SNOMED-CT concepts and their relationships, in an effort to further total semantic interoperability and seamless communication between healthcare entities.

**Keywords:** Ontology Mapping, Interoperability, HL7, SNOMED-CT, Health Informatics

## 1 Introduction

Information in the Healthcare domain is enormously complex, covering many different types of data. Patient administration, organisational information, clinical data and laboratory/pathology data are different but must be compensated. Add to this the integration of all of these areas, and storage in Electronic Health Records. As a result of this diversity and richness of the data, and also due to the fragmented nature of Health Informatics' implementation and research efforts, many different models have been designed to represent information in this field.

This publication includes SNOMED CT, a copyrighted work of the College of American Pathologists. ©2000, 2002 College of American Pathologists. This work is also protected by patent, U.S. Patent No. 6,438,533. SNOMED CT is used by permission of, and under license from, the College. SNOMED CT has been created by combining SNOMED RT and a computer based nomenclature and classification known as Clinical Terms Version 3, formerly known as Read Codes, Version 3, which was created on behalf of the U.K. Department of Health and is a crown copyright. SNOMED is a registered trademark of the College of American Pathologists.

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Australasian Ontology Workshop (AOW2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 72. M. A. Orgun and T. Meyer, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

One of the most successful of these information models is version 2 of the *Health Level 7* (HL7) standard. However, this standard does not achieve plug-and-play interoperability (Huff, Bidgood, Cimino & Hammond 1998, Klein 2005). In fact, any kind of interoperability between communicating HL7 systems can be time-consuming to establish and require careful attention to detail, especially between existing systems that have not been set up to communicate by design. Also, because HL7 was first developed in 1987 to suit technology at that time, it is in an out-dated format of different fields separated by "pipes" (|) and "carets" (^), where precision in the number of fields is a must - a single missing pipe could convert an otherwise meaningful message into a meaningless stream of text. The HL7 Organisation says this about the version 2 messaging standard:

These messages evolved over several years using a "bottom-up" approach that has addressed individual needs through an evolving ad-hoc methodology. There is neither a consistent view of that data that HL7 moves nor that data's relationship to other data (HL7 2006).

As such, no semantic inferences of the information can be determined automatically.

To overcome these problems, Version 3 of the HL7 standard was developed (HL7 2006). This standard incorporates a new paradigm for information representation and messaging in comparison to HL7 version 2 in terms of a new information model and intrinsic extensibility. This newer version is geared towards semantic interoperability between systems and is consistent with the technology developments, purpose and recommendations emerging from the Semantic Web. As such, implementing HL7 version 3 can be seen as a platform for the implementation of modern distributed systems standards, not just in Health Informatics, but as a platform to experiment with emerging distributed systems standards more generally.

The difficulty arises when trying to represent clinical concepts and constructs within the HL7 framework. One of the biggest obstructions to communication occurs when there are multiple ways of describing a single concept. For example, one person may write potassium as "pot" and another may write potassium as "K", which can make inferring semantics a complex problem. To overcome this, a standard vocabulary is needed to populate the model with meaningful data. If there existed a standard list of codes to represent many different clinical concepts, both people in the previous example could look up the code for potassium and use it, thus not allowing misunderstanding to occur. Huff *et al.* (1998) say "it is only by integrating the structure of a message with the vocabulary

sent in the message that unambiguous information exchange between systems can be achieved.”

To this end, there has been suggestions of HL7 consulting with domain experts to create standard vocabulary tables. In the same way that there exists many different information models for healthcare, this has already been done many times by domain experts who have created different terminologies for clinical and other healthcare data. Rather than repeating this time-consuming and exhaustive process, an existing terminology created by domain experts could be integrated with the HL7 information model.

In light of emerging Health Informatics standards use in Australia (NEHTA 2006), The terminology chosen to use in this project is the *Systemised Nomenclature of Medicine - Clinical Terms* (SNOMED-CT). SNOMED-CT is a result of a combination by the College of American Pathologists and the U.K. Department of Health of two existing terminologies (*SNOMED RT* and *Clinical Terms Version 3*) to create a unified terminology with a greater depth and coverage of healthcare data (SNOMED 2006). SNOMED-CT is widely regarded as the most comprehensive clinical healthcare terminology in the world and is even multilingual, covering both English and Spanish language concepts. The idea of using this terminology within the HL7 information model is not as simple as it sounds, as SNOMED-CT has its own rich information model. This paper outlines a strategy for normalisation between the two models involving mapping of concepts and relationships in SNOMED-CT to classes and attributes in HL7.

## 2 HL7 version 3

HL7 Version 3 uses an object-oriented development methodology and a Reference Information Model (RIM) to create messages (HL7 2006). The RIM is a UML<sup>1</sup>-style diagram based around six core types of classes and rules governing how they relate to each other. These rules, as well as further restrictions on allowable attributes for each class, make up the information model of HL7 version 3. Cardinality and optionality constraints also exist on relationships between classes and on attributes.

The six core classes of the RIM are:

- Act - an action of interest
- Entity - a class or instance of a specific thing capable of participating in Acts
- Role - An Entity, in a particular Role, can participate in an Act
- Participation - an association between a Role and an Act
- ActRelationship - an association between a pair of Acts
- RoleLink - a connection between two roles expressing a dependency

Three of these classes – Act, Entity and Role – are further represented by a set of specialized classes, or sub-types. E.g. specialisations of the Act class include Observation, Procedure and Substance Administration. As can be seen by the nature of the RIM classes, the HL7 information model takes an act-centred view, with processes and information in healthcare represented primarily in terms of the acts

performed within an organisational context (Vizenor 2004).

Acts, Entities and Roles all have an attribute called **code**. This attribute could be populated with a SNOMED-CT code corresponding to the kind of Act, Entity or Role it is. This is a current subject for discussion by the TermInfo Project, sponsored by the HL7 Vocabulary Technical Committee (HL7 2006, SNOMED 2006). This will be taken into account as part of the ontology mapping process.

## 3 SNOMED-CT

SNOMED-CT is made up of concepts and their attributes, which consist of relationships to other concepts. As such, relationships in SNOMED-CT are modelled as a triple of (concept, attribute, concept). For example, Figure 1 shows the concept “O/E - Blood Pressure Reading 163020007” in the centre with arrows to other concepts showing its attributes. “O/E” stands for *On Examination*, and belongs to the concept group *Clinical Finding*. The number after the name of the concept is the *SNOMED-CT Concept ID* (SCTID). The SNOMED-CT relationships from Figure 1 have been tabulated in Table 1.

Some examples of concept groups in SNOMED-CT are as follows:

- Clinical Findings - the results of a clinical observation, assessment or finding
- Procedures - purposeful activities performed in the provision of health care
- Body Structures - normal and abnormal body structures
- Substances - active chemical constituents of drug products, food, chemical allergens, toxicity information, etc
- Physical Objects - natural and man-made objects
- Events - occurrences that result in injury
- Observable Entities - procedures or questions which, when combined with a result, constitute a finding
- Qualifier Values - concepts not contained elsewhere in SNOMED-CT which are required for attributes e.g. open, left, right, etc

These “concept groups” are concepts themselves and are the top-level concepts in SNOMED-CT, as SNOMED-CT is structured as a multiple-inheritance hierarchy of concepts. Top-level concepts are concepts that are the direct children of the root concept. The root concept is the single, topmost concept in SNOMED-CT and is “SNOMED CT Concept” and has the SCTID 138875005.

SNOMED-CT models concepts and their relationships to each other in clinical constructs. As can be seen from the example concept groups, in particular Observable Entities, this can involve information on procedures which do not necessarily have to occur. This is in contrast to the HL7 act-centred view of healthcare information, where a procedure will only be recorded if it is intended to occur, or has occurred. The impact of this will be assessed in the mapping process.

<sup>1</sup>Unified Modeling Language - <http://www.uml.org/>



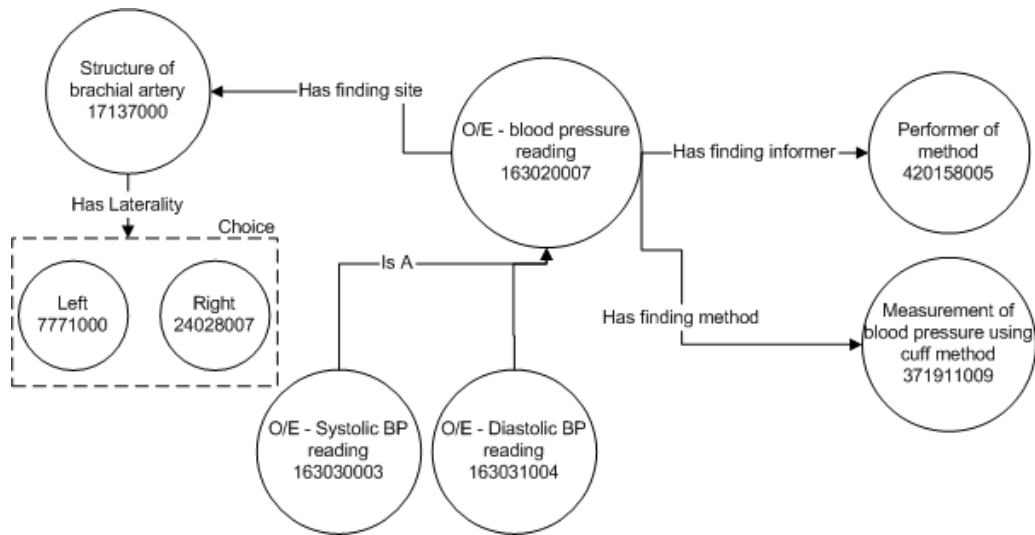


Figure 1: Blood Pressure Reading Concepts and Attributes in SNOMED-CT

Concept	Attribute	Concept
O/E - blood pressure reading	Finding site	Structure of brachial artery
Structure of brachial artery	Laterality	Left/Right
O/E - blood pressure reading	Finding informer	Performer of method
O/E - blood pressure reading	Finding method	Measurement of blood pressure using cuff method
O/E - systolic BP reading	Is a	O/E - blood pressure reading
O/E - diastolic BP reading	Is a	O/E - blood pressure reading

Table 1: SNOMED-CT relationships shown in Figure 1. Relationships in SNOMED-CT are modelled as a triple of (concept, attribute, concept).

#### 4 Ontology Mapping

An ontology  $O$  is defined by its set of Concepts  $C$  with a corresponding subsumption hierarchy  $H_C$ . Relations  $R$  exist between single concepts, which also have a corresponding hierarchy  $H_R$  (Ehrig & Staab 2004). Both the HL7 and SNOMED-CT information models can then be said to be ontologies.

Ontology mapping takes two ontologies as input and creates a semantic correspondence between the entities in the two input ontologies (Rahm & Bernstein 2001). Ehrig and Staab(2004) define ontology mapping:

Given two ontologies  $O_1$  and  $O_2$ , mapping one ontology onto another means that for each entity (concept  $C$ , relation  $R$ , or instance  $I$ ) in ontology  $O_1$ , we try to find a corresponding entity, which has the same intended meaning, in ontology  $O_2$ .

There are two kinds of conflicts between heterogeneous ontologies (Tang, Liang & Li 2005). The first conflict occurs when ontologies for the same domain knowledge have different semantic structures, which has been shown to occur between HL7 and SNOMED-CT in at least one respect - HL7's act-centred view and SNOMED-CT's main aim of modeling clinical constructs. This is to be expected as the two information models have different aims and purposes.

The second conflict occurs when either the same concept has different names in both ontologies, or the same name refers to different concepts in both ontologies. Both of these situations occur between the HL7 and SNOMED-CT information models. For example, one of the concept groups in SNOMED-CT is

'Event', referring to an occurrence which results in injury. Examples of this are "Accidental Fall", "Flood" and "Motor Vehicle Accident". HL7 also contains an 'Event', but this time it refers to an Act which has occurred. This could be that a patient's blood pressure was taken, or that a patient was admitted to hospital. In SNOMED-CT, a patient's blood pressure that has been taken is a Clinical Finding. These situations and conflicts will have to be taken into account during the mapping process.

Because SNOMED-CT models actual clinical constructs, the HL7 models will be based on existing SNOMED-CT models and the mapping will be in the direction of SNOMED-CT  $\rightarrow$  HL7.

As an example, the blood pressure reading concept from SNOMED-CT in Figure 1 has been mapped manually to HL7. Figure 1 could be represented in HL7 as shown in Figure 2.

The mapping of SNOMED-CT concepts to their container fields in HL7 for blood pressure reading is shown in Table 2. The first five rows show a relatively straightforward mapping between the two. E.g., the SNOMED-CT attribute **Finding site** is captured within an attribute of a similar name in HL7 (**targetSiteCode**), and the SNOMED-CT concept will become the data to populate this field. The last three rows in the table refer to attributes in SNOMED-CT which are mapped to classes in HL7- **ComponentOf** and **Informant**. This mapping is almost a reversal of the first mappings, with the SNOMED-CT attributes mapped to HL7 classes and the SNOMED-CT concepts mapped to an attribute in a separate HL7 class.

Another example, this time of the cannula insertion procedure, is shown in Figures 3 and 4. This example is more complex than the blood pressure

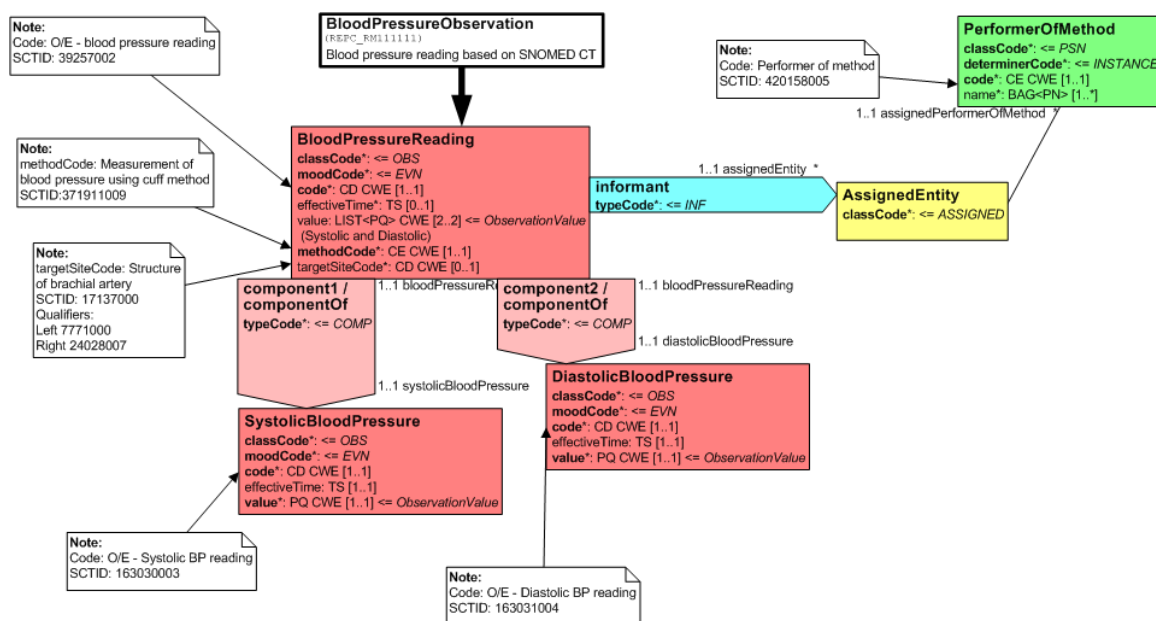


Figure 2: Blood Pressure Concept and its attributes from SNOMED-CT represented in HL7

SNOMED-CT		HL7	
Concept	Attribute	Class	Attribute
O/E - blood pressure reading	n/a	Observation	code
Structure of brachial artery	Finding Site	Observation	targetSiteCode
Left	Finding Site - Laterality	Observation	targetSiteCode
Right	Finding Site - Laterality	Observation	targetSiteCode
Measurement of blood pressure using cuff method	Finding method	Observation	methodCode
Performer of method	Finding Informer	Informant	Person::code
O/E - Systolic BP reading	Is a	ComponentOf	Observation::code
O/E - Diastolic BP reading	Is a	ComponentOf	Observation::code

Table 2: Mapping from SNOMED-CT to HL7 in Figures 1 and 2.

reading example and raises new issues. Blood pressure reading is an observation, and was relatively simple to model. Cannula Insertion is a procedure, and the differences in modelling between an observation and a procedure suggest that different considerations will have to be taken into account for every type of HL7 Act. The mapping of SNOMED-CT concepts to their container fields in HL7 for cannula insertion is shown in Table 3. Note that the mapping in this case is mostly in the form of SNOMED-CT codes fitting into different code attributes of the one HL7 Act subclass (procedure). The SNOMED-CT concepts marked with an asterisk (\*) denote higher level concepts in the SNOMED-CT hierarchy. In the blood pressure example, all concepts were at the lowest level, so no further choices or specialisations of the concepts could be made. For cannula insertion this was not possible, so the concepts shown in the diagram as “choice” are specialisations of the SNOMED-CT concepts shown in Table 3. This suggests that automatic mapping of this concept to HL7 may not be achievable, as some decisions will need to be human-made.

As can be seen from these simple examples, careful thought and study of the SNOMED-CT dataset will have to be put into the mapping process. To further complicate matters, HL7 has some vocabulary

tables of its own, used in fields such as `classCode` and `moodCode`. One suggestion that has been made in trying to draw the terminology and the message structure together is the inclusion of the HL7 vocabulary terms and concepts as an addition to SNOMED-CT.

## 5 Conclusion

The HL7 RIM and SNOMED-CT are ontologies which need to be able to work together, so what is needed is a mapping from one ontology to the other. The results of this mapping could facilitate the automatic generation of HL7 messages from the structure of SNOMED-CT's concepts and relationships. If fully automated generation of HL7 messages cannot be achieved, it may still be attainable with only a few human-made decisions, which would be as acceptable in a real world setting.

In trying to achieve this goal, the question of whether HL7 Version 3 can effectively model clinical concepts and their relationships can also be answered.

The ultimate goal of this research is to create a model for combining messaging and terminologies in a seamless way that promotes total semantic interoperability between systems and ease-of-use for healthcare systems developers and users.

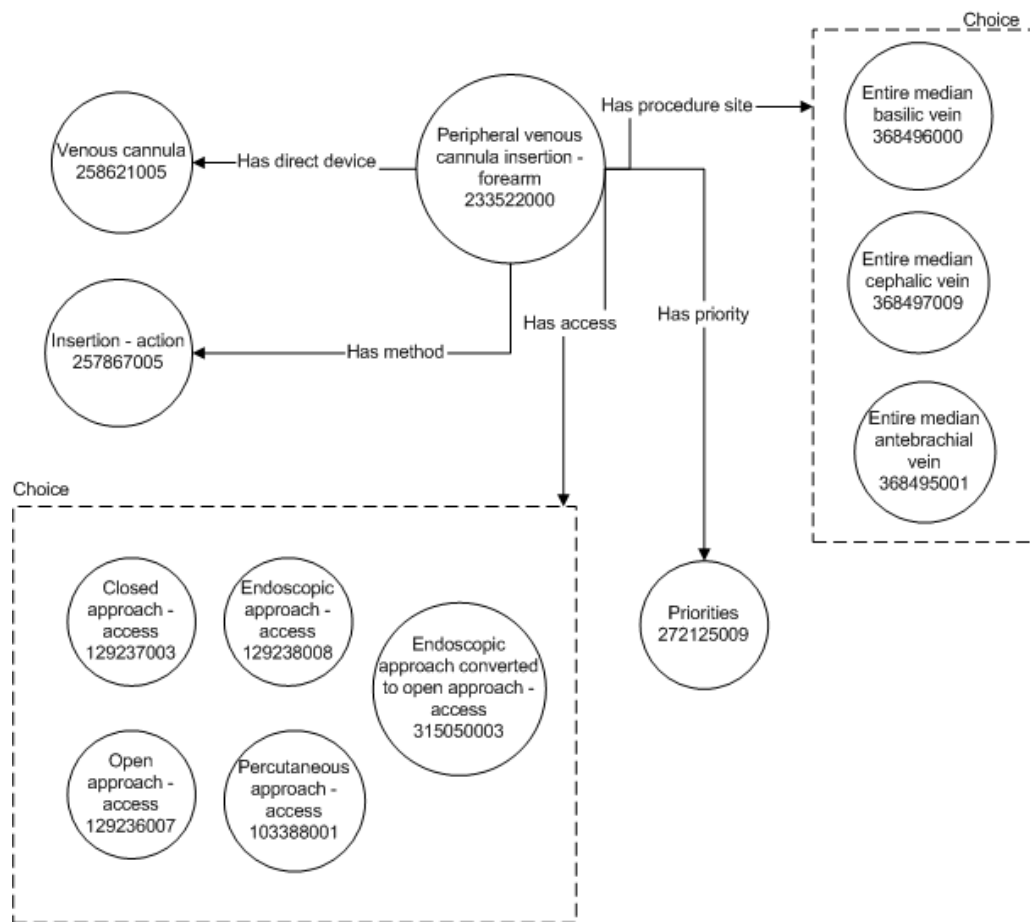


Figure 3: Cannula Insertion Concepts and Attributes in SNOMED-CT

## References

- Ehrig, M. & Staab, S. (2004), QOM - Quick Ontology Mapping, in S.A. McIlraith, ed, 'ISWC 2004', LCNS 3298, pp. 683-697.
- Health Level 7, 'About HL7' & 'HL7 Terminology', *Health Level 7 webpage*, <http://www.hl7.org/>, last accessed 07/09/06.
- Huff, S. M., Bidgood, W. D, Cimino, J. J. & Hammond, W. E. (1998), A Proposal for Incorporating Health Level Seven (HL7) Vocabulary in the UMLS Metathesaurus, in 'Journal of American Medical Informatics Association, AMIA Annual Fall Symposium Supplement', pp. 800-804.
- Klein, J. (2005), Integrating Electronic Health Records using HL7 Clinical Document Architecture, in 'Business Integration Journal'.
- National E-Health Transition Authority (2006), *National E-Health Standards Development: A Management Framework*, version 1.0.
- Rahm, E. and Bernstein, P.A (2001), A survey of approaches to automatic schema matching, in 'The VLDB Journal', Vol. 10, pp. 334-350.
- SNOMED International (2006), 'What is SNOMED-CT' & 'Terminology', *SNOMED International webpage*, [http://www.snomed.org/snomedct/what\\_is.html](http://www.snomed.org/snomedct/what_is.html) & <http://www.snomed.org/Terminology/Terminologylink.html>, last accessed 17/09/06.
- Tang, J., Liang, B.Y. & Li, J.Z., Toward Detecting Mapping Strategies for Ontology Interoperability, in 'WWW 2005', May 10 - 14, 2005, Chiba, Japan.
- Vizenor, L., Actions in Health Care Organizations: An Ontological Analysis, in M. Fieschi et al, eds, 'Medinfo 2004', pp. 1403-1407, 2004, IOS Press, Amsterdam,

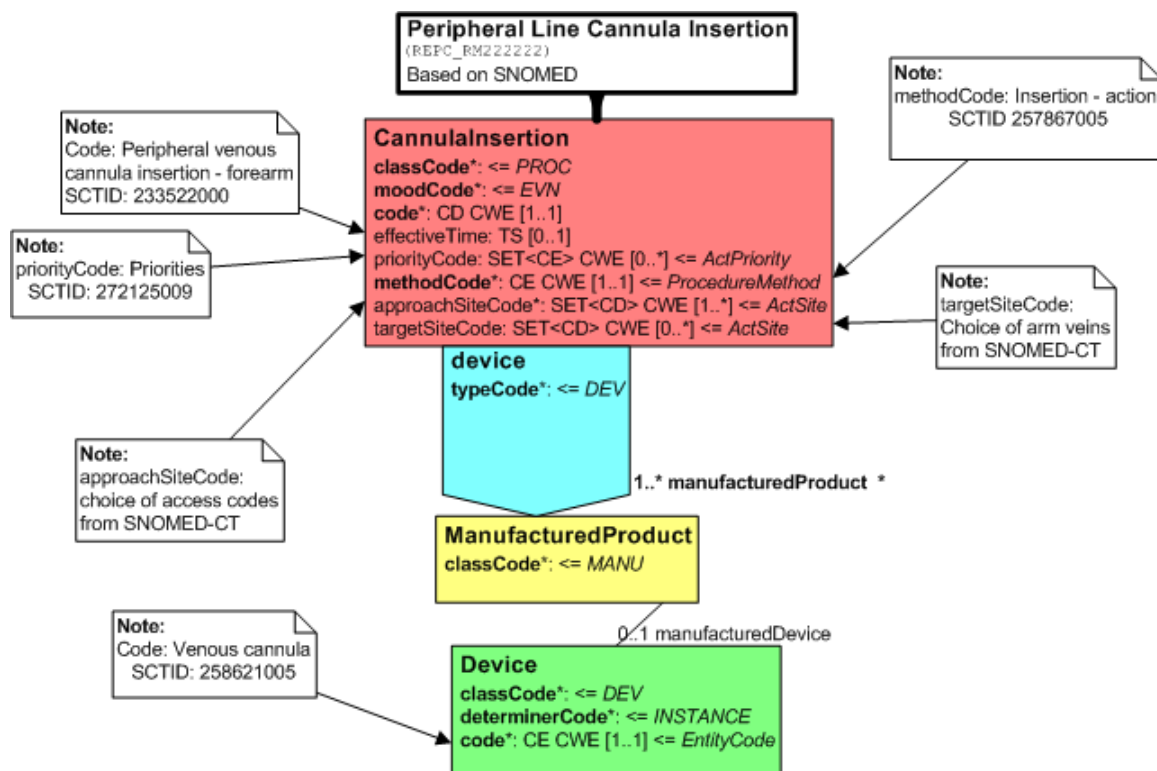


Figure 4: Cannula Insertion Concept and its attributes from SNOMED-CT represented in HL7

SNOMED-CT		HL7	
Concept	Attribute	Class	Attribute
Peripheral venous cannula insertion - forearm	n/a	Procedure	code
Insertion - Action	Method	Procedure	methodCode
Structure of superficial forearm vein*	Procedure Site	Procedure	targetSiteCode
Priorities*	Priority	Procedure	priorityCode
Surgical access values*	Access	Procedure	approachSiteCode

Table 3: Mapping from SNOMED-CT to HL7 in Figures 3 and 4.

# Annotating Websites with Machine-processable Information in Controlled Natural Language

Rolf Schwitter and Marc Tilbrook

Centre for Language Technology  
Macquarie University  
Sydney, 2109 Australia

{schwitt|marct}@ics.mq.edu.au

## Abstract

In this paper we present a user friendly approach to annotate websites with machine-processable information in controlled natural language. The controlled natural language serves as a high-level specification and knowledge representation language which allows human annotators to summarise individual web pages of a website and to express domain-specific ontological knowledge about that website in an unambiguous subset of English. The annotation process is backed up by an intelligent text editor which supports the writing process of the controlled natural language with the help of text- and menu-based predictive interface techniques. The text editor runs as a Java applet and is connected over the Internet to a controlled natural language processor and to a reasoning service (consisting of a theorem prover and a model builder). The controlled language processor translates the summaries of web pages and the ontological knowledge about a website into first-order predicate logic and the reasoning service combines this information into a set of micro theories for consistency and informativity checking as well as for question answering. Specification texts written in controlled natural language are both human-readable and machine-processable, and can be easily exported and distributed as web feeds.

**Keywords:** Knowledge Representation, Ontologies, Controlled Natural Languages, Theorem Proving, Model Building, Question Answering

## 1 Introduction

It has been argued that the current architecture for the Semantic Web, with its strong emphasis on RDF for syntactic and semantic compatibility, has severe problems when expressive Semantic Web (reasoning) languages are incorporated (Patel-Schneider 2005). An alternative approach is to use conventional first-order logic as the semantic underpinning for the Semantic Web. First-order logic is well understood, and well established subsets of first-order logic offer tradeoffs with respect to expressive power, complexity and computability (Horrocks and Patel-Schneider 2003). For example, the direct mapping of description logic-based ontology languages and Horn rule languages into subsets of first-order logic provides

immediate semantic interoperability and builds the prerequisite for efficient reasoning (Grosz, Horrocks, Volz, and Decker 2003). Instead of relying on RDF, we suggest using a machine-oriented controlled natural language which is based on first-order logic as an interface language to the Semantic Web. To promote our approach, we will introduce a prototype application, which uses a controlled natural language to summarise web pages and to augment these summaries with domain-specific ontological knowledge. The result is a web feed which is easy to read by humans in contrast to other formal languages and as easy to process by a machine as other formal languages.

The rest of this paper is structured as follows: In Section 2, we briefly explain what controlled natural languages are, for what they can be used, and what kind of problems they can solve. In Section 3, we present PENG, a machine-oriented controlled natural language that looks seemingly informal, but can be unambiguously processed as a formal specification language. In Section 4, we introduce PENG Online, an intelligent text editor, which supports the writing of web page summaries and the specification of ontological knowledge in controlled natural language. In Section 5, we look at some details of the controlled natural language processor which is used by PENG Online to translate the controlled natural language into first-order predicate logic. In Section 6, we touch on the reasoning service which accomplishes several reasoning tasks. In Section 7, we show that a web feed specification in controlled natural language can directly be exported as an RSS feed, and finally in Section 8, we conclude and summarise the advantages of our approach.

## 2 Controlled Natural Languages

In general, a controlled natural language is a subset of a full natural language with explicit restrictions on the grammar, lexicon, and style. These restrictions usually have the form of writing rules and help to reduce (or even exclude) ambiguity and to cut down the complexity of full natural language. Traditionally, controlled natural languages fall into two categories: human-oriented and machine-oriented controlled natural languages. Human-oriented controlled natural languages (for example ASD Simplified Technical English (ASD 2005)) aim at improving text comprehension for human readers while machine-oriented controlled natural languages (for example Common Logic Controlled English (Sowa 2004)) focus on improving text processability for machines. An important difference between human-oriented and machine-

oriented controlled natural languages is that the writing rules for machine-oriented controlled natural languages must be precise and computationally tractable (Huijsen 1998). However, as a rule of thumb, simplification works in both ways: human-oriented controlled natural languages are also easier to process by machines and machine-oriented controlled natural languages are also easier to understand by humans compared to full natural language.

### 3 PENG (Processable ENGLISH)

PENG is a machine-oriented controlled natural language designed for writing unambiguous and precise specification texts for knowledge representation (Schwitter 2002, Schwitter 2004, Schwitter 2005). PENG covers a strict subset of standard English and is precisely defined by a controlled grammar and a controlled lexicon. Specification texts written in PENG are incrementally parsed using a unification-based phrase structure grammar and then translated into first-order predicate logic via discourse representations structures (Kamp and Reyle 1993, Schwitter and Tilbrook 2004). In the general case, the result is a logic theory which can be checked for consistency and informativity as well as be used for question answering. In contrast to other machine-oriented controlled natural languages (Pullman 1996, Fuchs, Schwertel and Schwitter 1999, Holt, Klein and Grover 1999, Sowa 2004), the author of a PENG text *does not* need to know the grammatical restrictions of the language explicitly. The text editor of the PENG system dynamically enforces these restrictions while the text is written and displays the interpretation of a sentence in the form of a paraphrase in controlled natural language.

#### 3.1 The Philosophy of PENG

The language PENG can be used as a high-level specification and knowledge representation language. Specification texts written in PENG look seemingly informal on the surface level, similar to full English, but in contrast to full English the language is designed to bring about the same precision and formality as a formal specification language. All sentences in PENG are correct English but only an unambiguous subset of English sentence structures and verb form-types are allowed in PENG. For example, PENG restricts the use of verb form-types in contrast to full natural language. In PENG verbs can only be used in their active voice, in their indicative mood, and in their simple present tense. Furthermore, modal verbs (such as *can*, *must*, *should*, etc.) and intensional verbs (such as *believe*, *seek*, *want*, etc.) are not allowed, since the underlying formal language does not immediately support modalities or intensional contexts. All seemingly ambiguous constructions in PENG are interpreted in a principled way and the interpretation is reflected in an unambiguous paraphrase. In summary: PENG has been carefully designed to be easy for humans to read and to write and easy for machines to process.

#### 3.2 The Grammar of PENG

The grammar of PENG defines how words and their constituents combine to form simple sentences, complex

sentences and questions. In our scenario simple and complex sentences are used to summarise web pages and to specify ontological information about a website. Questions are then used to interrogate various aspects of the resulting micro theories, for example to query the existence of a situation or to find specific entities which are part of a situation.

##### 3.2.1 Simple Sentences

Simple sentences have a hierarchical structure consisting of words and constituents whereas each word is itself a constituent. Several constituents can be joined together in a controlled way to form simple PENG sentences. Constituents can be distinguished according to their function and their form. In the subsequent sentence:

1. *Bill Smith reboots the webserver on Monday.*

the constituent *Bill Smith* functions as the subject of the sentence and its form is a noun phrase. The constituent *reboots the web server on Monday* functions as the predicate of the sentence with the verb *reboots* as predicator while the constituent's form is a verb phrase. The functional dependents of the predicator within the verb phrase are of two kinds: complements and adjuncts. The constituent *the webserver* functions as a necessary complement of the predicator and its form is again a noun phrase. Finally, the constituent *on Monday* functions as an optional adjunct of the predicator (since the sentence is syntactically well-formed without this constituent) and its form is a prepositional phrase.

At the highest level, simple PENG sentences are composed of the following functional units:

*subject + predicator + complements + adjuncts*

Instantiations of this functional pattern are, for example, sentences such as:

2. *Bill Smith works.*
3. *Bill Smith maintains Apache.*
4. *Bill Smith works at Macquarie University.*
5. *Bill Smith is a diligent research programmer.*
6. *Bill Smith who is a diligent research programmer works at Macquarie University.*
7. *The supervisor of Bill owns a BMW.*
8. *The research programmer owns a Sony laptop.*

Sentence (2) shows the simplest possible structure of a PENG sentence consisting of a noun phrase (*Bill Smith*) in subject position and an intransitive verb (*works*) in predicator position. In sentence (3), a transitive verb (*maintains*) subcategorizes for a noun phrase (*Apache*) which occurs in complement position. In sentence (4), a prepositional phrase (*at Macquarie University*) occurs in adjunct position and modifies the verb (*works*) or better the underlying verbal event. In sentence (5), an adjective (*diligent*) occurs as a pre-nominal modifier of a complex noun (*research programmer*). In sentence (6), a relative sentence (*who is a diligent research programmer*) occurs as a post-nominal modifier of a proper noun (*Bill Smith*)

and constitute together a noun phrase. In sentence (7), the *of*-construction marks the noun (*supervisor*) as a relational noun with two arguments. In sentence (8), the noun phrase (*the research programmer*) in subject position is definite and the noun phrase (*a Sony laptop*) in complement position is indefinite. Definite noun phrases can be used to refer to previously introduced objects and indefinite noun phrases introduce new objects into the universe of discourse.

### 3.2.2 Complex Sentences

In PENG, complex sentences are built from simpler constituents and sentences with the help of a small number of constructors (coordinators, subordinators, quantifiers and negation markers). The subsequent sentences are examples of complex sentences:

9. *Bill Smith works at Macquarie University and maintains a webserver.*
10. *Bill Smith owns a Sony laptop or an Apple iPod.*
11. *Bill Smith is not a staff member.*
12. *No research programmer is a staff member.*
13. *Every research programmer owns a laptop.*
14. *If X is a research programmer then X is a programmer.*
15. *If X is a research programmer then X is not a staff member.*
16. *Every research programmer is a programmer.*
17. *If X buys Y then X acquires Y.*

In sentence (9), two verb phrases are coordinated by means of the conjunctive coordinator *and*. In sentence (10), two verb phrases are coordinated by means of the disjunctive coordinator *or*. In sentence (11), the negation marker *not* negates the entire verb phrase in complement position and in (12) the negation marker *no* negates the entire noun phrase in subject position. In sentence (13), the universal quantifier *every* is used to speak about all objects which belong to a specific class. In sentence (14), (15) and (17), the subordinator *if* introduces the antecedent of a conditional statement. Note that sentence (14) and sentence (16) are logical equivalent. The only difference between these two sentences is that the universally quantified variable (*X*) is made explicit in sentence (14) on the surface of the controlled natural language. This is a powerful mechanism to generate class hierarchies (see for example (14)) and property hierarchies (see for example (17)) in controlled natural language. As we will see later, this mechanism allows us to also – among other things – specify domain and range restrictions of properties.

### 3.2.3 Questions

In PENG, questions can be used to query the content of a specification text. Questions are systematically derived from simple and complex sentences to extract information from the constituents of these sentences and to guarantee wide coverage for question answering. Formally, *yes/no*-questions are built via subject-operator inversion

and *wh*-questions are built by moving the interrogative word (e.g. *where*, *when*, *how*) to the initial position in the sentence, and where needed, by inserting the dummy *do* operator after the interrogative word. The following are examples of questions which can be used to interrogate a specification text written in PENG:

18. *Does Bill Smith work?*
19. *Who maintains a web server?*
20. *Where does Bill Smith work?*
21. *When does Bill reboot the webserver?*
22. *Is Bill Smith a programmer?*
23. *Is every research programmer a programmer?*
24. *Who maintains a web server and owns a laptop?*

*Yes/no*-questions such as (18), (22) and (23) allow us to check whether a specific situation is true or not and *wh*-questions such as (19), (20), (21) and (24) allow us to interrogate a specific aspect of a situation (for example finding a person who is involved in an event, a specific location or a point in time).

## 3.3 The Lexicon of PENG

The controlled lexicon of PENG consists of a base lexicon and a user lexicon. The base lexicon contains the most frequent content words of English (*proper nouns*, *common nouns*, *verbs*, *adjectives*, and *adverbs*) and pre-defined function words (*determiners*, *prepositions*, *coordinators*, *subordinators*, *negation* and *disambiguation markers*) which build the syntactic scaffolding of the controlled natural language. The base lexicon also contains illegal words (which cannot be processed by the PENG system). The user lexicon can be extended with domain-specific content words by the annotator while a text is written in controlled natural language.

## 4 PENG Online

PENG Online implements the web-based version of the PENG editor. The editor features built-in browser functionality for viewing web pages. It also provides a layout for expressing ontological knowledge about a website and for summarising the content of individual web pages which belong to that website. The editor can be used to create and update machine-processable descriptions of websites and to export them as web feeds in RSS format.

### 4.1 Architecture

PENG Online is based on a client-server architecture which consists of three main components: an intelligent text editor, a controlled natural language processor, and a reasoning service.

The web-based editor is implemented as a Java applet which runs in a web browser and communicates with a Prolog server via a socket interface. The Prolog server implements the controlled natural language processor and the reasoning service.



The controlled language processor incrementally translates specification texts into first-order predicate logic via discourse representation structures and generates predictive look-ahead information for the text editor as well as paraphrases for the input text.

The reasoning service makes use of SRI's Open Agent Architecture (OAA) where a facilitator coordinates a number of agents (Martin, Cheyer, and Moran, 1999, Cheyer and Martin, 2001). In our case, the reasoning interface agent fuses the summaries of web pages and the ontological knowledge about the website into a set of micro theories. These micro theories are sent to the facilitator which utilises a model builder agent and a theorem prover agent. These two reasoning agents complement each other and can check the micro theory for either consistency or informativity. These can also be utilised as a starting point for question answering.

The ontological knowledge about a website and the textual summaries of the individual web pages can be exported as a web feed in RSS format. Since the information is available in controlled natural language and fully human readable, any RSS feed aggregator can subscribe to such a web feed. However, the full benefit of having a machine-processable controlled natural language can only be brought into effect by a PENG-compliant tool which can reprocess these web feeds.

## 4.2 The PENG Editor

The PENG editor provides a standard mode and a web feed mode. The standard mode can be used to write normal specification texts in controlled natural language. The web feed mode is specially designed to annotate websites in controlled natural language. When the annotator selects the web feed mode, the text editor asks if the current user lexicon should be used for the new task or if a new user lexicon should be created. Once selected, the editor displays the interface of the feed mode as shown in Figure 1:

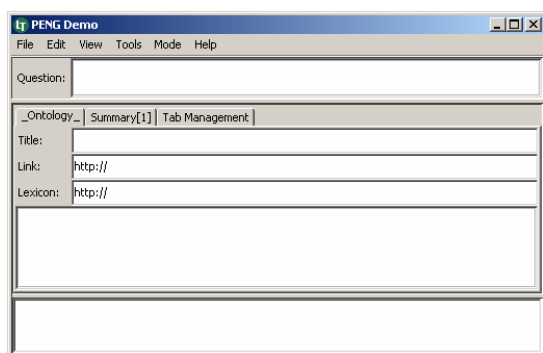


Figure 1: The PENG Editor in Web Feed Mode

This interface has a tabbed pane containing an ontology pane for the specification of the ontological knowledge about a website and one or more summary panes which are part of the website. Below the tabbed pane there is a message field for the system feedback and above the tabbed pane is a question field for asking questions about various aspects of a feed specification. The annotator can view a website using the built-in browser of the text editor which is available from the tools menu in the menu bar.

### 4.2.1 The Ontology Pane

In Figure 1, the ontology pane is active but still empty. This pane contains a title field for the name of the web feed, a link field for a URL to the HTML web page that corresponds to the channel, a lexicon field for a URL that points to the (exported) user lexicon of the controlled natural language, and a description field for the specification of the domain-specific ontological knowledge about a website. For example, the following complex PENG sentences express ontological knowledge about a website:

25. *If X is a research programmer then X is a programmer.*
26. *If X is a research programmer then X is not a staff member.*
27. *If X is a staff member then X is not a research programmer.*
28. *If X maintains Y then X is a programmer and Y is a webserver.*
29. *If X maintains Y then X looks after Y.*

Sentence (25) specifies a hierarchical class relationship between the subclass *research programmer* and the superclass *programmer*. The two sentences (26) and (27) specify that the two classes *research programmer* and *staff member* are disjoint. In Sentence (28), the verb (= property) is restricted in its domain by the class *programmer* and in its range by the class *webserver*. That means that only individuals that belong to the class *programmer* can occur in the subject position and only individuals that belong to the class *webserver* can occur in the complement position. Finally, in sentence (29), a hierarchical property relationship between the transitive verb *maintains* and the prepositional verb *looks after* is specified. Please note that all these sentences fall under the description logic subset of the controlled natural language (for details see Schwitter and Tilbrook 2006).

### 4.2.2 The Summary Pane

In Figure 2, the summary pane is active. This pane contains a title field for the name of a web page, a link field for the URL which points to the original web page and a description field for the summary of a web page in controlled natural language:

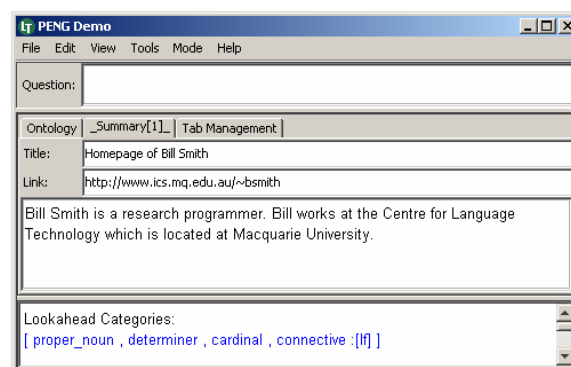


Figure 2: The Summary Pane



As this example illustrates, the annotator already added the title of the web page to the title field and the URL of the original web page occurs in the link field. The description field so far contains the following two sentences:

30. *Bill Smith is a research programmer.*  
 31. *Bill works at the Center for Language Technology which is located at Macquarie University.*

These sentences describe parts of the original web page (see Figure 3) that is currently open for annotating in the browser. Note that the message field in Figure 2 contains look-ahead information that informs the annotator about how the current specification text can be continued (for details see Section 4.2.3).



Figure 3: Excerpt of the Original Web Page

Note that not all of the information in the original web page can be represented in controlled natural language. The idea is to produce a machine-processable summary of a web page that can be easily read by humans and efficiently processed by a machine. This requires a careful tradeoff between expressiveness and processability of the controlled natural language.

### 4.2.3 Writing in PENG

The form of the input to the description field of both the ontology pane and the summary pane is restricted by the language processor of PENG. The language processor generates look-ahead information for each word form that the annotator enters while the specification text is written. This look-ahead information consists of syntactic categories which predict what kind of input can follow the current word form. The look-ahead categories are implemented as hypertext links. By clicking on a look-ahead category the author is able to access help information. The author composes a sentence either by typing the word forms which fall under the look-ahead categories or by selecting word forms from a cascade of menus (Schwitter, Ljungberg and Hood 2003, Thompson, Pazandak and Tennant 2005).

Please note that the look-ahead categories are generated on the fly and use linguistic information produced by the incremental chart parser of the controlled language processor. The processing of these look-ahead categories does not slow down the author significantly while typing the text and happens in near real-time (ca. 140 milliseconds on average per word form).

The look-ahead categories in Figure 2 indicate that the author can continue the specification text, for example, using a proper noun as in (32), a determiner as in (33), a

cardinal number as in (34), or a specific subordinator as in (35):

32. ... at Macquarie University. **Bill** ...  
 33. ... at Macquarie University. **The** ...  
 34. ... at Macquarie University. **Two** ...  
 35. ... at Macquarie University. **If** ...

Instead of typing an approved word form into the description field of the editor, the author can alternatively select a word form from the currently active look-ahead categories via the context menu as Figure 4 illustrates:

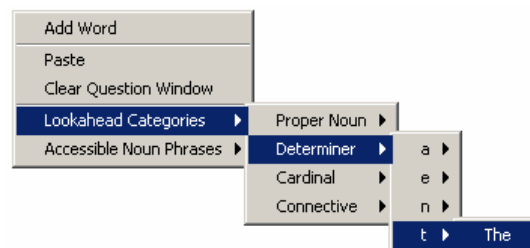


Figure 4: Active Look-ahead Categories

Once such a word form has been selected, it will be immediately inserted into the text at the current cursor position and the processing of the text is automatically resumed. Not only can approved word forms be inserted in this way, but also all noun phrases which are accessible in the specification text. Accessible noun phrases occur in the context menu and can be selected from there. Figure 5 shows that after the processing of sentence (30) and (31) the following three noun phrases are available in the context menu:

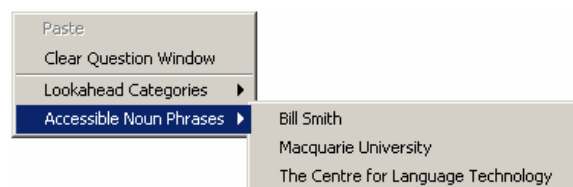


Figure 5: Accessible Noun Phrases

Please note that the noun phrase *a research programmer* is not accessible here, since it forms a property together with the copulative verb *be* and cannot be referred to by a definite noun phrase.

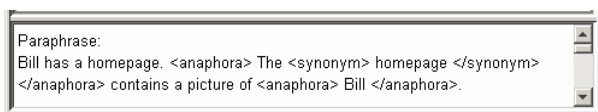
### 4.2.4 The Message Field

The message field displays a paraphrase for each sentence and clarifies the interpretation of the input – if this option is selected. The paraphrase indicates, for example, if synonyms or anaphoric expressions have been used in the text. Let us assume that the author added the following two sentences to the description field:

36. *Bill has a homepage.*  
 37. *The page contains a picture of Bill.*

And let us further assume that the noun *page* has previously been defined as a synonym of its main form *homepage* in the user lexicon. After processing this informa-

tion, the paraphrase in the message field will indicate – as Figure 6 illustrates – that the noun phrase *the homepage* and the proper noun *Bill* are two anaphoric expressions which have been previously introduced in the text and that the synonym *page* has been replaced, respectively normalized, by its main form *homepage*.

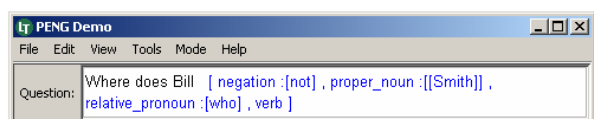


**Figure 6:** Paraphrase in Controlled Natural Language

Additionally, the message field displays the syntax tree for the last input sentence, the actual discourse representation structure for the entire text and its representation in first-order predicate logic. Furthermore, the message field shows the output of the reasoning engine (the proof or the model) and the specific result (for example the answer to a question). Not all of this information is relevant for the annotator and parts of this information can therefore be selectively removed.

#### 4.2.5 The Question Field

The purpose of the question field is to interrogate a web feed in controlled natural language. As Figure 7 illustrates, the question field uses the same kind of look-ahead mechanism to guide the writing process as the description fields of the ontology and the summary pane.



**Figure 7:** Question Field with Look-ahead Categories

Once a question is completely formulated, it is translated into first-order predicate logic via discourse representation structures (similar to simple and complex sentences), combined with the micro theories, and answered with the help of PENG's reasoning service.

#### 4.2.6 The Lexical Editor

Part of the text editor is a lexical editor for adding user-specific content words. If the author enters a content word (i.e. *proper noun*, *common noun*, *verb*, *adjective* or *adverb*) into the text editor which is not yet available in the lexicon and is not in the list of illegal words, then this content word needs to be added to the user lexicon of the PENG system. The interface to the lexical editor has been designed in such a way that only minimal linguistic knowledge is required by the author to add a new content word to the lexicon. As soon as a new content word is available in the lexicon, the parsing process is resumed. User-defined content words can also be deleted from the user lexicon, but the author cannot delete words in the base lexicon of the PENG system which contains the most frequent 3000 words of English as well as all predefined function words. Note that the existing user lexicon (or a new user lexicon) is exported once the web feed is complete.

## 5 The Controlled Language Processor

When the author types a word form into the text editor, this word form is immediately sent to the incremental chart parser of the controlled language processor. The chart parser uses a unification-based phrase structure grammar as syntactic scaffolding (Schwitter 2003, Schwitter and Tilbrook 2004). As Figure 8 shows, the phrase structure rules of the grammar are highly parameterised.

```
s([ coord:no, drs:D, para:P1-P4, tree:[s,T1,T2],
  gap:G, styp:decl, snum:N, sana:M1-M3 ])
-->
n3([ coord:_, arg:I, spec:Q, ana:A, drs:D,
  sco:S, para:P1-P2, tree:T1, gap:n3:[ ]-[ ],
  styp:decl, snum:N, sana:M1-M2 ]),
v3([ coord:_, vform:fin, arg:I, drs:S, para:P2-P3,
  tree:T2, gap:G, styp:decl, snum:N, sana:M2-M3 ]),
pm([ cat:pm, para:P3-P4, snum:N ]).
```

**Figure 8:** A Phrase Structure Rule

The beauty of this approach is that it allows us to deal with syntactic, semantic and pragmatic information concurrently and in the same logic-based framework. The grammar currently consists of about 150 such phrase structure rules. During parsing the incremental chart parser generates a chart which can be used to harvest the look-ahead information for the text editor. The other important information in the chart is the discourse representation structure which represents the meaning of the text. For example, the two sentences (36) and (37) repeated here as (38) and (39)

38. *Bill Smith has a homepage.*

39. *The page contains a picture of Bill Smith.*

result in the following (simplified) discourse representation structure:

```
[A,B,C,D,E]
obj([bill,smith],A),
pred(C,[have],A,B),
obj([homepage],B),
pred(E,[contain],A,D),
obj([picture],D,A).
```

**Figure 9:** Simplified Discourse Representation Structure

whereas the variables *A*, *B*, *C*, and *D* represent discourse referents and the predicates conditions which hold for these discourse referents. Discourse representation theory (Kamp and Reyle 1993) allows us to deal in an elegant way with anaphoric references between sentences. Such discourse representation structures can be translated in linear time into a set of first-order logic formulas. These first-order logic formulas can then be further processed by the reasoning service of the PENG system as we will describe in the next section.

## 6 The Reasoning Service

One possible setting of the reasoning service is to use the theorem prover Otter (McCune 2003a) in combination with the model builder Mace4 (McCune 2003b) for consistency and informativity checking as well as for question answering (Bos 2003, Blackburn and Bos 2003, Blackburn and Bos 2005). Another interesting option we

are currently exploring but that we will not further discuss here is to use Satchmo instead of Otter and Mace4 as reasoning service (see Manthey and Bry 1988 and in particular Fuchs and Schwertel 2003 for a discussion).

The idea of using a theorem prover and a model builder in combination has been explored for other (natural) language processing tasks, for example for solving logical puzzles (Schwitter 2002, Lev, MacCartney, Manning, and Levy, 2004) and as a spoken language interface to a robot and in an automated home environment (Bos 2006).

## 6.1 Otter and Mace

Otter is an automated theorem prover for first-order logic with equality that searches for a refutation of a set of formulas and is designed to detect inconsistency (or unsatisfiability) of a theory. Mace4 is a model builder that searches for finite models of first-order formulas for a given domain size and its task is to check for satisfiability of a theory.

Otter and Mace4 can work on the same problem at the same time and complement each other. If Otter can find a proof for the negation of a set of formulas, then Mace4 has to do an exhaustive search that does potentially not terminate. In this case, Otter can inform Mace4 to stop searching for a model as soon as it found a proof. In a similar way, if Mace4 can build a finite model for the formulas, then Otter has to do an exhaustive search that does potentially not terminate. In this case, Mace4 can inform Otter to stop searching for a proof. Of course, the problem of detecting whether a set of first-order formulas is valid is not decidable and therefore we have to assign a time limit on the search for both Otter and Mace4.

The input to Otter and Mace4 can be specified with first-order formulas or first-order clauses or a combination of both. If the input consists of non-clausal first-order formulas, then the input is immediately further translated into first-order clauses involving negation normal form conversion, skolemisation, quantifier operations, and conjunctive normal form conversion. In contrast to Mace4, Otter has an interactive and an autonomous mode for selecting search strategies and provides more options to control the processing, but both accept similar input files.

## 6.2 Otter and Mace in PENG

The PENG system translates the discourse representation structures (which have been derived from the ontological knowledge), the summaries, and the questions, into first-order formulas. These formulas in turn are combined in various ways depending on the reasoning task. In PENG, we distinguish three reasoning tasks: consistency checking, informativity checking and question answering. Each task requires a specific preparation of the formulas which results in a micro theory to be processed by Otter and Mace4.

In our case, Otter runs in the autonomous mode and takes a micro theory as input, translates the input into clauses, scans the clauses and automatically decides on inference rules and a search strategy. Mace4 takes the micro theory,

translates the input first into clauses and then into an equivalent propositional problem which is then given to a satisfiability procedure.

Before we discuss the various reasoning tasks in more detail, let us assume that  $\Phi$  is a set of first-order formulas derived from the text in controlled natural language which summarises a web page;  $X$  is a set of first-order formulas derived from the text in controlled natural language which describes the ontological knowledge about a website;  $\Psi$  is a first-order formula derived from a new sentence; and  $\delta$  is a first-order formula derived from a question stated in controlled natural language, and finally  $A$  is an answer literal. Answer literals record instantiations of variables during Otter's search for a refutation proof and can be used to answer *wh*-questions.

### 6.2.1 Consistency Checking

A micro theory  $(X \wedge \Phi)$  is consistent if and only if all formulas can be satisfied together in some model with the same variable assignment. In the case of Otter we need to find out if  $\neg(X \wedge \Phi)$  is valid and in the case of Mace4 we need to find out if  $(X \wedge \Phi)$  is satisfiable.

If we give the negation of the micro theory  $\neg(X \wedge \Phi)$  to Otter (thus we give it  $\neg\neg(X \wedge \Phi)$ ) and it finds a proof for this input, then we know that  $(X \wedge \Phi)$  is not consistent. If a micro theory is not consistent, then a theorem prover like Otter will always succeed in finding a proof.

If we give the micro theory  $(X \wedge \Phi)$  to Mace4 and it successfully builds a finite model for this input, then we know that  $(X \wedge \Phi)$  must be satisfiable (= consistent). If a micro theory is consistent and satisfiable on a finite model, then a model builder like Mace4 will always succeed in building a model.

### 6.2.2 Informativity Checking

A new formula  $\Psi$  is informative with respect to a context  $(X \wedge \Phi)$  if and only if it is not a logical consequence of this context (or not satisfiable in all models).

If we give the negation of the micro theory  $(X \wedge \Phi \rightarrow \Psi)$  to Otter and it finds a proof for this input, then we know that  $\Psi$  is not informative. If a new formula is not informative, then a theorem prover like Otter will always succeed in finding a proof.

If we give the micro theory  $(X \wedge \Phi \wedge \neg\Psi)$  to Mace4 and it builds a finite model, then we know that  $\Psi$  is informative. If a new formula is informative, then a model builder like Mace4 will always succeed in building a model.

### 6.2.3 Question Answering

The simplest type of questions are *yes/no*-questions which do not contain free variables. However, *wh*-questions contain free variables which need to be bound to specific values during a proof. In order to accomplish this, the translation of interrogative words in the case of Otter results in answer literals which can be used to record instantiations of variables during a search for refutation. Mace4 does not provide such a mechanism, since

Mace4 is in fact a model builder and not a model checker which could tell us whether the model satisfies a query or not. However, Mace4 builds minimal models which are not redundant and answers to questions can be looked up immediately in the model once such a model exists.

If we give the micro theory  $\neg\delta \wedge (X \wedge \Phi)$  to Otter and it finds a proof for this input, then we know that  $\delta$  results in a positive answer to a *yes/no*-question.

If we give the micro theory  $\neg(\delta \wedge A) \wedge (X \wedge \Phi)$  to Otter and it finds a proof for this input, then we know that the variable bindings in the answer literal  $A$  are results for a *wh*-question.

If we give the micro theory  $(X \wedge \Phi)$  to Mace4 and it builds a finite model, then we can start searching for answers to the question  $\delta$  in this model. However, this process requires a simple transformation of the model which Mace4 generates into a model which corresponds to the formal signature of the question.

#### 6.2.4 An Example

We will now illustrate the reasoning abilities of the PENG system by a few examples. Let us assume that  $\Phi_1$  represents the two sentences:

- *Bill Smith is a research programmer.*
- *Bill works at Macquarie University.*

$\Phi_2$  represents the two sentences:

- *Bill Smith is a research programmer.*
- *Bill Smith is not a programmer.*

$\Psi_1$  represents the new sentence:

- *Bill Smith is a programmer.*

$X_1$  represents the ontological background information:

- *If  $X$  is a research programmer then  $X$  is a programmer.*
- *If  $X$  is a research programmer then  $X$  is not a staff member.*
- *If  $X$  is a staff member then  $X$  is not a research programmer.*

Furthermore, let us assume that  $\delta_1, \delta_2, \delta_3, \delta_4$  and  $\delta_5$  represent the five subsequent questions:

- *Does Bill Smith work at Macquarie University?*
- *Is Bill Smith a programmer?*
- *Is Bill Smith a staff member?*
- *Who works at Macquarie University?*
- *Where does Bill Smith work?*

If we want to check the micro theory  $(X_1 \wedge \Phi_1)$  for *consistency* and feed the negation of  $\neg(X_1 \wedge \Phi_1)$  to Otter and  $(X_1 \wedge \Phi_1)$  to Mace4, then Mace4 will find a satisfiable model and we can stop Otter searching for a proof. That means we know that the micro theory is consistent.

If we want to check the micro theory  $(X_1 \wedge \Phi_2)$  for *consistency* and feed the negation of  $\neg(X_1 \wedge \Phi_2)$  to Otter and  $(X_1 \wedge \Phi_2)$  to Mace4, then Otter will find a proof and we can stop Mace4 looking for a finite satisfiable model. That means we know that the micro theory is not consistent.

If we want to check the formula  $\Psi_1$  for *informativity* with respect to the context  $(X_1 \wedge \Phi_1)$  and feed the negation of  $(X_1 \wedge \Phi_1 \rightarrow \Psi_1)$  to Otter and  $(X_1 \wedge \Phi_1 \wedge \neg\Psi_1)$  to Mace4, then Otter will find a proof and we can stop Mace4 looking for a model. That means we know that the theory is not informative.

If we want to answer questions such as  $\delta_1$ - $\delta_5$ , then we have to negate the formulas derived from the questions, before we combine them in a micro theory and feed them to Otter, since Otter conducts a resolution proof. This is not necessary for Mace4, since we can extract answers to questions from the model in a separate step. Figure 10 shows the input to Otter for the question (theorem)

- *Where does Bill Smith work?*

given the information (axiom)

- *Bill Smith works at Macquarie University.*

without any additional background knowledge.

```
set(auto).
assign(max_seconds,5).
assign(max_proofs,-1).
set(prolog_style_variables).

formula_list(sos).

((exists A (exists B
(exists C (exists D
(prop(A,[at],B,C) &
(role(A,location) &
(pred(B,[work],D) &
(evt1(B,event) &
(obj([macquarie,university],C) &
(struc(C,atomic) &
(obj([bill,smith],D) &
(struc(D,atomic)))))))))) &
-((exists E (exists F (exists G
(exists H (exists I (exists J (exists K
(prop(E,F,G,H) &
(role(E,location) &
(obj(I,H) &
(struc(H,J) &
(pred(G,[work],K) &
(evt1(G,event) &
(obj([bill,smith],K) &
(struc(K,atomic)))))) &
-$answer([[where],F,I,E,H]])))))))).

end_of_list.
```

**Figure 10:** Input to Otter with Answer Literal

As the input to Otter shows the question has been negated and an answer literal has been added. The answer literal

- $-$answer([[where],F,I,E,H]).$

retains the interrogative word and records the variable bindings during the proof for the subsequent answer generation.

## 7 RSS Export

A web feed written in controlled natural language can be exported as an RSS feed. RSS is a family of XML-based web feed formats designed for sharing and aggregating web content (RSS 2002). RSS feeds provide summaries of web content together with links to the full versions of the content. In our case, the specification texts written in the web feed mode can be exported as an RSS feed. Basically, an RSS feed is an XML document consisting of an `<rss>` element with a single `<channel>` element, which contains meta information about the channel and its content, and any number of `<item>` elements, which store the summaries of individual web pages. Let us have a closer look at the general structure of an RSS feed that is generated by PENG Online system:

```
<?xml version="1.0"?>
<rss version="2.0">

  <channel>
    <language> x-peng </language>
    <generator> PENG Online </generator>
    <title> ChannelTitle </title>
    <link> ChannelLink </link>
    <description> Ontology </description>
    <category domain=URILexicon> </category>

    <item>
      <title> WebpageTitle </title>
      <link> WebPageURI </link>
      <description> WebPageSummary
      </description>
    </item>

  </channel>
</rss>
```

**Figure 11:** Structure of RSS Feed

In our case the `<channel>` element uses six different subelements for storing the meta information and one or more `<item>` elements for storing information about individual web pages. The first subelement of the `<channel>` element is the `<language>` element which stores the information about the language the channel is written in. In our case, the value `x-peng` denotes an experimental language tag for the controlled natural language PENG. The second subelement is the `<generator>` element which indicates that the program used to generate the channel is PENG Online. The third subelement is the `<title>` element and specifies the title of the channel. The fourth subelement is the `<link>` element which contains the channel's URL. The fifth subelement is the `<description>` element which stores the ontological knowledge about the web feed. The sixth subelement is the `<category>` element which is empty in our case but uses an attribute with a URL as value. The URL points to the exported user lexicon which needs to be accessed when the RSS feed is reloaded by the PENG system. In our case, a `<channel>` element may contain one or more `<item>` elements - one for each summary of a web page which is part of the web site. The `<item>` element has a `<title>` element as subelement which stores the title of the web page and a `<link>` element which points to the full version of the web page. Finally, the `<description>` element of the `<item>` element stores the summary of the web page in controlled natural language.

## 8 Conclusions

In this paper, we presented a new approach that allows non-specialists to annotate individual web pages of a website with machine-processable information in controlled natural language and to augment these descriptions with domain-specific ontological information in controlled natural language. The writing process of these specification texts is supported by a text editor which uses predictive interface techniques. The text editor is implemented as a Java applet and communicates over the Internet with a language processor and a reasoning service. The language processor provides look-ahead information for the text editor and translates a specification text into first-order predicate logic via discourse representation structures. The resulting first-order formulas can be combined for various reasoning tasks into micro theories. These micro theories are processed by a reasoning service which combines a theorem prover together with a model builder. The theorem prover provides a negative check on consistency, informativity and questions, and the model builder provides a positive check for the same inference tasks. It is important to note that Web feeds written in PENG are both human-readable and machine-processable and can be maintained by non-specialists with the help of the PENG editor. Any RSS aggregator can subscribe to such a "seemingly informal" web feed, but the full processing power is only available via PENG Online or another PENG-compliant tool.

## Acknowledgments

The research reported here is supported by the Australian Research Council, Discovery Project No. DP0449928. We would also like to thank to three anonymous reviewers for their valuable comments.

## References

- ASD (2005): ASD Simplified Technical English, Specification ASD-STE100, *A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language*, Issue 3, January.
- Blackburn, P. and Bos, J. (2003): Computational Semantics, in: *Theoria* 18(1), pp. 27-45.
- Blackburn, P. and Bos, J. (2005): *Representation and Inference for Natural Language*, A First Course in Computational Semantics, CSLI Publications.
- Bos, J. (2003): Exploring Model Building for Natural Language Understanding, in: *Proceedings of ICoS-4*.
- Bos, J. (2006): Three Stories on Automated Reasoning for Natural Language Understanding, in: *Proceedings of ESCoR (IJCAR Workshop)*: Empirically Successful Computerized Reasoning, pp. 81-91.
- Cheyen, A. and Martin, D. (2001): The Open Agent Architecture, in: *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 4, no. 1, March, pp. 143-148.
- Fuchs, N. E. and Schwitter, R. (1996): Attempto Controlled English (ACE), in: *Proceedings of CLAW 96*,

- First International Workshop on Controlled Language Applications, University of Leuven, Belgium, March 1996, pp. 124-136.
- Fuchs, N. E., Schwertel, U. and Schwitter, R. (1999): Attempto Controlled English – Not Just Another Logic Specification Language, *Lecture Notes in Computer Science* 1559, Springer.
- Fuchs, N. E. and Schwertel, U. (2003): Reasoning in Attempto Controlled English, in: F. Bry, N. Henze and J. Maluszynski (eds.): *Principles and Practice of Semantic Web Reasoning, International Workshop PPSWR 2003*, Mumbai, India, pp. 174-188.
- Grosz, B., Horrocks, I., Volz, R. and Decker, S. (2003): Description Logic Programs: Combining Logic Programs with Description Logic, in: *Proceedings of WWW 2003*, Hungary, ACM, pp. 48-57.
- Holt, A., Klein, K. and Grover, C. (1999): Natural language for hardware verification, in: *Proceedings of ICoS-1 workshop: Inference in Computational Semantics*, Institute for Logic, Language and Computation (ILLC), Amsterdam, August, pp. 133-137.
- Horrocks, I. and Patel-Schneider, P. F. (2003): Three theses of representation in the semantic web, in: *Proceedings of WWW 2003*, Hungary, ACM, pp. 39-47.
- Huijsen, W. O. (1998): Controlled Language – An Introduction, in: *Proceedings of CLAW 1998*, Pittsburgh, pp. 1-15.
- Kamp, H. and Reyle, U., (1993): *From Discourse to Logic*, Kluwer Academic Publisher.
- Lev, I., MacCartney, B., Manning, C. D. and Levy, R. (2004): Solving logic puzzles: from robust processing to precise semantics, in: *Proceedings of the ACL-04 Workshop on Text Meaning and Interpretation*, pp. 9-16.
- Manthey, R. and Bry, F. (1988): SATCHMO: A Theorem Prover Implemented in Prolog, in: *Proceedings CADE 88*, pp. 415-434.
- Martin, D., Cheyer, A. and Moran, D. (1999): The Open Agent Architecture: A Framework for Building Distributed Software Systems, in: *Applied Artificial Intelligence*, vol. 13, no. 1-2, January-March, pp. 91-128.
- McCune, W. (2003a): Otter 3.3 Reference Manual. Tech. Memo ANL/MCS-TM-263, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, August 2003.
- McCune, W. (2003b): Mace4 Reference Manual and Guide. Tech. Memo ANL/MCS-TM-264, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, August 2003.
- Patel-Schneider, P. F. (2005): A Revised Architecture for the Semantic Web Reasoning, in: *Proceedings of PPSWR'05*, Dagstuhl, Germany, LNCS 3703, pp. 32-36.
- Pulman, S. G. (1996): Controlled Language for Knowledge Representation, in: *Proceedings of CLAW 96*, First International Workshop on Controlled Language Applications, University of Leuven, Belgium, March 1996, pp. 233-242.
- RSS 2.0 Specification (2002): Technology at Harvard Law, Internet technology hosted by Berkman Center, available at: <http://blogs.law.harvard.edu/tech/rss>.
- Schwitter, R. (2002): English as a Formal Specification Language, in: *Proceedings of the Thirteenth International Workshop on Database and Expert Systems Applications (DEXA 2002)*, W04: Third International Workshop on Natural Language and Information Systems - NLIS, 2-6 September 2002, Aix-en-Provence, France, pp. 228-232.
- Schwitter, R. (2003): Incremental Chart Parsing with Predictive Hints, in: *Proceedings of the Australasian Language Technology Workshop 2003*, December 10, University of Melbourne, Australia, pp. 1-8.
- Schwitter, R., Ljungberg, A. and Hood, D. (2003): ECOLE - A Look-ahead Editor for a Controlled Language, in: *Proceedings of EAMT-CLAW03*, Controlled Translation, Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Application Workshop, May 15-17, Dublin City University, Ireland, pp. 141-150.
- Schwitter, R. (2004): Dynamic Semantics for a Controlled Natural Language, in: *Proceedings of the Fifteenth International Workshop on Database and Expert Systems Applications (DEXA 2004)*, NLIS'04: 4th International Workshop on Natural Language and Information Systems, 30 August - 3 September 2004, Zaragoza, Spain, pp. 43-47.
- Schwitter, R. and Tilbrook, M., (2004): Dynamic Semantics at Work, in: *Proceedings of LENLS2004* (in conjunction with the 18th Annual Conference of the Japanese Society for Artificial Intelligence, 2004), in Kanazawa (Japan), May 31, pp. 49-60.
- Schwitter, R. (2005): A Layered Controlled Natural Language for Knowledge Representation, in: S. Cardey, P. Greenfield and S. Vienney (eds.), *Machine Translation, Controlled Languages and Specialised Languages: Special Issue of Linguisticae Investigationes*, Vol. 28, No. 1, pp. 85-106.
- Schwitter, R. and Tilbrook, M. (2006): Let's Talk in Description Logic via Controlled Natural Language, in: *Proceedings of the Third International Workshop on Logic and Engineering of Natural Language Semantics (LENLS2006)* in Conjunction with the 20th Annual Conference of the Japanese Society for Artificial Intelligence, Tokyo, Japan, June 5-6, pp. 193-207.
- Sowa, J. F. (2004): Common Logic Controlled English, *Draft*, 24 February 2004, available at: <http://www.jfsowa.com/lce/specs.htm>.
- Thompson, C. W., Pazandak, P. and Tennant, H. R. (2005): Talk to Your Semantic Web, in: *IEEE Internet Computing*, Vol. 9, No. 6, pp. 75-79.

# A Native Ontology Approach for Semantic Service Descriptions

Rajesh Thiagarajan and Markus Stumptner

Advanced Computing Research Centre  
University of South Australia  
Email: {cirsrt,mst}@cs.unisa.edu.au

## Abstract

A number of ontology-based approaches have been suggested for the description of service behaviors to be used in service composition and matching in service oriented architectures. We examine an approach based on classical software engineering notation and compare it to other approaches.

## 1 Introduction

Semantic Web technology has been a significant driver of recent research in a number of related areas such as data and application integration, distributed business process management, and service oriented architectures, all areas where declaratively describing the behavior of pieces of software can be used for the management, combination, and sometimes execution control of these pieces. These approaches typically build on approaches used in particular sub-communities, such as the use of OWL-S<sup>1</sup> as a language for specifying various ontologies used within Web Service based applications. Past work has examined the similarities and analogies holding between these domains and languages and classical OO principles used in software engineering (Koide, Aasman & Haflich 2005, Djuric, Gasevic & Devedzic 2005). In this paper we try to build the bridge back into Model Driven Architecture (MDA) territory by using the types of descriptive semantics employed in Semantic Web service technology, in the guise of classical OO design notations. We describe services in the style of Semantic web specifications, but use classical software engineering notations and sublanguages, in the shape of UML and OCL, and we reason directly on these service specifications.

Web Services are programs that can be remotely accessed using the protocols of the World Wide Web, with communications based on standards such as Simple Object Access Protocol(SOAP)<sup>2</sup>. The Web Service Description Language (WSDL)<sup>3</sup> is the current standard to describe Web Services. WSDL describes a service in terms of multiple *ports*, each of which defines sets of ingoing and outgoing messages that can be used to communicate with the service through

that port. A message is an XML structure that contains the parameters (called *parts*) necessary for execution of the service, or passes back the result. WSDL defines four operation modes, *notifications*, *one-way*, *solicit-response*, and *request-response*, each of which corresponds to a particular combination of output and input messages (e.g., *solicit-response* generates an output message and receives a return message). In addition, a WSDL description gives a *binding* for a service, specifying the actual communication protocol (e.g., SOAP, HTTP, or MIME) and its settings.

The service matching counterpart to WSDL's service descriptions is UDDI (Universal Description, Discovery and Integration). UDDI describes businesses in terms of physical attributes (name, address, and lists of services), plus extended attributes called *TModels* that describe services by reference to standard taxonomies such as NAICS (North American Industry Classification System). However, UDDI has been recognized to be severely limited in practice due to the fact that it solely supports keyword searches (in effect, string matching) on its attributes and does not permit any description of actual service semantics.

On the other side, WSDL has also been found too weak to fulfill the higher level tasks that were envisioned for it (Petrie, Genesereth, Bjornsson, Chirkova, Ekstrom, Gomi, Hinrichs, Hoskins, Kassoff, Kato, Kawazoe, Min & Mohsin 2003). It does not provide semantics for its operations in any machine interpretable way, merely specifying their names, nor does it specify the relationship (sequences) between different operations.

From this standpoint, WSDL and UDDI are therefore perfectly matched; both rely on correct naming and interpretation of names. As a result, UDDI and WSDL-related technologies descriptions are considered insufficiently powerful to capture the information required for effective tool support for combining services, and extensive research on other methods is being conducted.

Semantic Web (Berners-Lee, Hendler & Lassila 2001) technology offers the vision of Web of resources where each resource is annotated with machine interpretable descriptions. This capability would enable development of intelligent applications to perform a variety of different tasks on the resources automatically. Like any other resource that are part of the Semantic Web, Web services should also be annotated with machine interpretable descriptions - service descriptions that provide insight into the semantics of the service rather than just the standard port/parameter information described above, are referred to as *Semantic Web Service*. Figure 1 shows the descriptive associations of such a service. Representing the semantics of Web Services and utilizing the described semantic knowledge to develop and use within intelligent applications is an active research domain. Automatic composition of Web Ser-

Copyright © 2006, Australian Computer Society, Inc. This paper appeared at the Australasian Ontology Workshop (AOW 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 72. M. A. Orgun and T. Meyer, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup>OWL-S Version 1.1 Specification - <http://www.daml.org/services/owl-s/1.1/>

<sup>2</sup>SOAP 1.1 - <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

<sup>3</sup>WSDL 1.1 - <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>



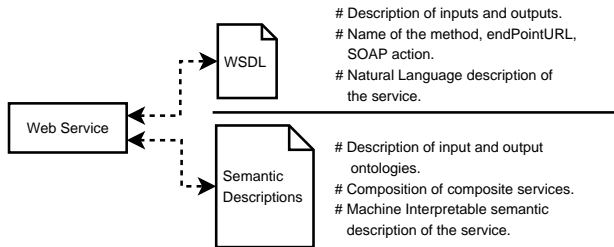


Figure 1: Semantic Web Service

vices and reconfiguration of composed Web Services are some of the likely functions the intelligent applications ought to be performing by exploiting the semantic descriptions of services. In contrast to the standard WSDL description, at this time there exists no official standard for representing semantic descriptions, but, OWL-S is the leading approach for representing the semantic descriptions of Web Services.

### 1.1 Matchmaking Example

The primary aims of semantic Web service descriptions are to facilitate automated discovery by semantic matchmaking, as well as to provide automated support for Web service composition (Albert, Henocque & Kleiner 2005, Medjahed, Bouguettaya & Elmagarmid 2003, Wu, Parsia, Sirin, Hendler & Nau 2003). Matchmaking is the process of selecting a set of candidate services that matches a given request. Figure 2 shows a simple example of a syntactic and semantic matchmaking process. In this paper we mostly concentrate on the matchmaking process for ease of exposition.

The syntactic candidate selection is the process of finding services with input and output data types matching the request. The syntactic matching process only takes into account the data types of inputs and outputs while making the selection. It may not always be the case that services exist that exactly match the request. Under those circumstances, the matchmaker can be allowed to make non-exact matches such as 'Plug-In' and 'Relaxed' (Kawamura, Blasio, Hasegawa, Paolucci & Sycara 2004). As a consequence, the matching process may result in a number of false positives. For example in Figure 2, the 'bookInfo' service is selected as a match for a request which intended to find a dictionary. The undesired result was returned because the input and output data types of the service matched with the request.

Semantic matchmaking processes take into consideration the functionality of the service while selecting candidates. The semantic matchmaking process shown in Figure 2 works under the assumption that all services in the directory are semantically described. For example, even though the input and output data types of the service 'bookInfo' matched with the request, the service is not selected because the functionality of the request did not match with the functionality of 'bookInfo'.

### 1.2 Motivation

Most of the approaches for semantic service definition and composition use OWL-S to define the service description and to define the ontologies within the service description. However, when applied to practical application scenarios, direct use of OWL-S is still subject to a number of restrictions due to the in-progress nature of the approach (Balzer, Liebig & Wagner 2004), with issues arising during profile

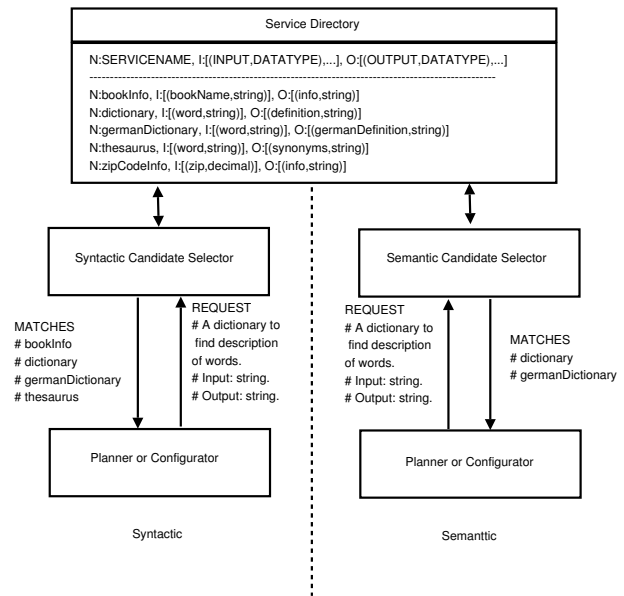


Figure 2: Syntactic and Semantic Matchmaking

matchmaking and process execution. Rather than suggesting local workarounds to counter these issues we approach this problem from a model based software engineering perspective. In this paper, we examine an approach based on classical software engineering notation and justify the aspects of our approach that addresses the issues raised in (Balzer et al. 2004).

Albert et al. (Albert et al. 2005) used a constrained object model, which was based on the configuration framework by Mailharro (Mailharro 1998), to devise work flow composition as a configuration problem. The constrained object model had a meta model of the components in the work flow, an ontology mapping of data types and a set of composition constraints such as 'at least one of the inputs of the choice node should be active to pass control to the next node'. Their work demonstrated the effectiveness of using configuration based approaches for composition of services whereas our approach is towards modeling the service descriptions using standard software engineering practices.

### 1.3 Example Scenario

A fictitious *HotelKroneBookingProcess* was defined in (Balzer et al. 2004). The booking process in Figure 3 was used to investigate the pitfalls of OWL-S in a practical situation. In this paper we use the same application scenario to examine the aspects of the languages in our approach that addresses the issues. We assume objects, both parameters and intermediate objects, involved in the process can have relationship between them. For example the objects *CreditCard* and *Customer* may be linked with *belongsTo* relationship. We also assume inheritance hierarchies between the objects in the process. For example *ShippingAddress* inherits from *Address*.

The rest of the paper is organized as follows. Section 2 is an account of the relevant related work about the role of UML and OCL on the Semantic Web. Section 3 outlines our examination of the aspects of our UML/OCL based approach that addresses the drawbacks of OWL-S identified in (Balzer et al. 2004). In Section 6 we summarize our observations.



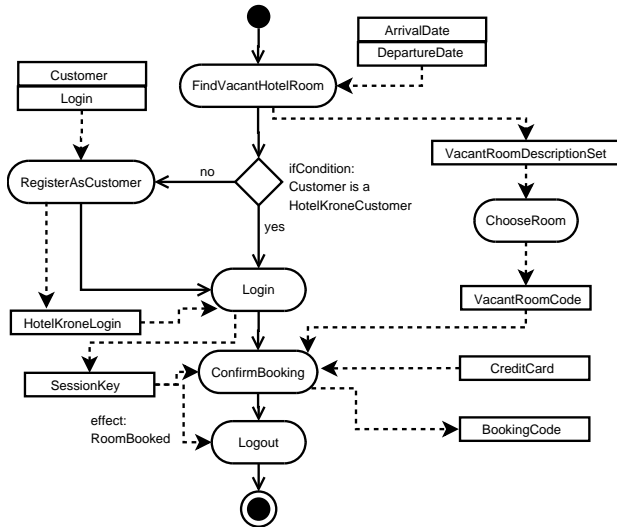


Figure 3: Example Scenario from (Balzer et al. 2004)

## 2 UML, OCL and the Semantic Web

The Object Constraint Language (OCL) is a part of the UML specification provided by the Object Management Group (OMG). OCL was introduced in UML to define constraints on the objects in a model based architecture and plays a crucial role in expanding the scope of UML models, by expressing complex relationships that are not easily captured through the various UML diagram notations. In some cases, OCL is also used to define business rules within the model. UML and OCL are widely accepted as a part of the object oriented software engineering approach by both the academic and industrial community.

The feasibility of using UML as a graphical definition language in a Semantic Web context has been investigated in (Falkovych, Sabou & Stuckenschmidt 2003), where the focus is placed mainly on solving the modeling problem and the transformation of UML diagrams to Web ontology languages. Drawbacks such as the absence of variables to represent procedural knowledge and the additional type mapping requirements discovered in (Balzer et al. 2004) were not addressed. Composition of services has been discussed in (Skogan, Grønmo & Solheim 2004) where UML Activity diagrams with minor modifications are applied. UML diagrams were used as a graphical paradigm to assist human comprehension but the service semantics were defined in OWL-S.

### 2.1 Ontology Representation

Djuric developed a meta model to represent systems centered around ontologies in a standard UML format (Djuric 2004). A UML profile for ontologies enables the modeling of semantic services within the MDA framework. A feasibility study and an approach for translating UML profiles defined in MDA to OWL-S is detailed in (Gasevic, Djuric & Devedzic 2005). A composition approach was proposed in (Grønmo, Jaeger & Hoff 2005) based on the UML profiles defined in (Djuric 2004). The main focus of the work mentioned here is to examine transformation between UML models and languages such as OWL-S. We investigate an approach based on UML and OCL in a practical application scenario, and consider how the approach addresses some of the issues raised in (Balzer et al. 2004).

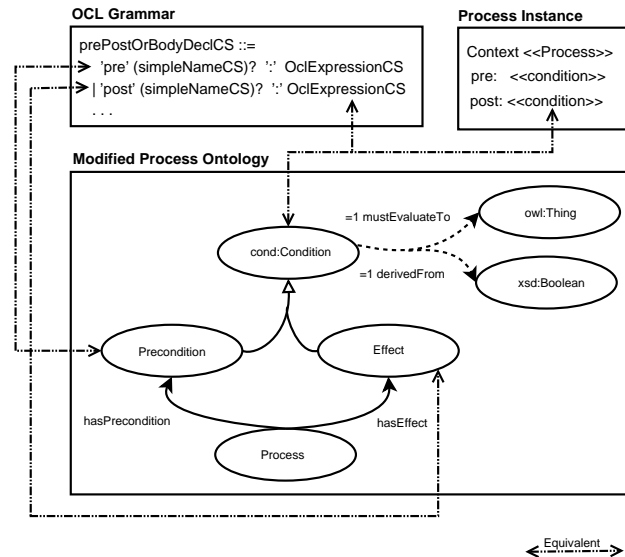


Figure 4: Modified Process Ontology (Balzer et al. 2004) and OCL Equivalents

## 3 OWL-S vs. UML/OCL

OWL-S does not yet provide means for expressing semantic properties in a standardised way. In particular, the description of effects and pre and post conditions rely on external languages such as Knowledge Interchange Format (KIF) and Semantic Web Rule Language (SWRL). This diversity adversely affects the integration and reuse of formal descriptions, as translation between formalisms in general implies loss of information. In this paper, however, our focus is only to examine a parallel approach for describing Semantic Web Services. In this section, we examine how the use of UML/OCL would address these issues raised in (Balzer et al. 2004).

### 3.1 Conditions in the Process Model

The absence of variables in OWL-S is a disadvantage (Balzer et al. 2004). Critical procedural knowledge such as the conditions and parameter instance bindings cannot be expressed in OWL-S. Integrating additional concepts to the OWL-S process ontology would bring in to OWL-S the ability to express conditions (Balzer et al. 2004).

Figure 4 shows a part of the modified process model proposed in (Balzer et al. 2004). The modified process ontology has a reified concept `cond:Condition` to represent process preconditions and effects. Figure 4 also shows a snippet of `prePostOrBodyDeclCS` expression from the OCL grammar and process instance template. As counterparts to the `hasPrecondition` and `hasEffect` properties in OWL-S, OCL has the `pre` and `post` aspects that enable qualifying a process described in OCL with Boolean constraints. Generally conditions are strictly bound to a specific instance of parameters. Variables in OCL can be used to identify specific parameter instances. For example a specific instance of the parameter `Customer` in the `HotelKroneBookingProcess` (see Figure 3) can be mapped to a variable `currentCustomer` and conditions such as `currentCustomer.isMember()` can be evaluated within condition expressions. The target of the process effect can also be effectively expressed using OCL variables. For example assume `accountCharged` to be an effect of the `HotelKroneBookingProcess` (see Figure 3). The target of the effect can be expressed

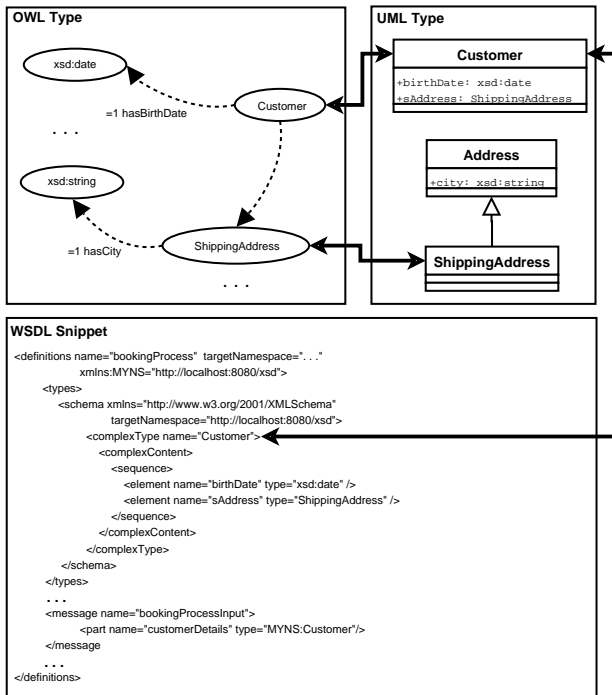


Figure 5: OWL Datatype Mapping to WSDL

as a variable `account` where `account` is a specific instance of `CreditCard` which is in turn linked to a specific instance of `Customer` class. OCL variables can hold specific parameter instances which does away with the limitation that preconditions should implicitly be modeled with the inputs. If the preconditions are modeled with the inputs then conditions involving multiple input objects cannot be expressed. Unlike (Balzer et al. 2004) where the preconditions are modeled with inputs, OCL expressions give the freedom to express conditions such as '(Customer credit card is valid) or (Customer is a frequent visitor and Customer has positive credit history)'. Unless reified concepts proposed in the modified process ontology (Balzer et al. 2004) are introduced, OWL-S will not have the expressiveness to represent critical procedural knowledge. On the other hand, OCL has sufficient expressiveness to represent procedural formalisms such as conditions and parameter instances (if desired).

### 3.2 Grounding

Balzer et al. noted that a different XSLT sheet would have to be developed for every serialization of an OWL type into XML Schema, making serialization of the OWL types into XML Schema using XSLT problematic (Balzer et al. 2004). They developed a semantic RDF mapping ontology to overcome this. In the mapping ontology an RDF class `XSDType` is used to map the corresponding OWL type. `ComplexType`, `SimpleType` and `ArrayType` are the three subclasses of `XSDType` class. The attributes of the OWL types are mapped using the `mapsTo` property of `SimpleType` class. UML can be used as an intermediate transformation language while transforming OWL to WSDL (Ha & Lee 2006). However, the transformation of OWL types into UML also suffers the earlier mentioned XSLT serialization problem.

Figure 5 shows a snippet of the OWL types `Customer` and `ShippingAddress`. The figure also shows, in our approach, the UML equivalents of these OWL

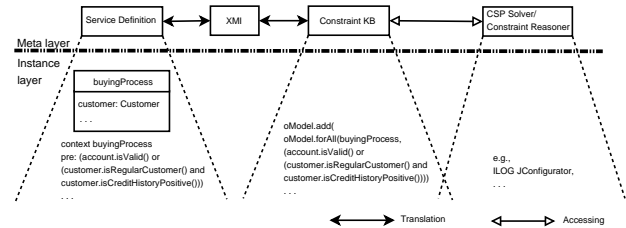


Figure 6: System Architecture

type definitions. XML Metadata Interchange (XMI)<sup>4</sup> is the standard XML-based format to interchange metadata such as class schema information. Strict serialization of UML classes can be configured in XMI. Once configured, for a given UML type there could possibly only be one serialization. This serialized structure can be mapped to either its equivalent RDF class or it can be mapped to a `complexType` in XML schema. Figure 5 also shows our mapping of UML type `Customer` into a `complexType Customer`. This type definition in XML schema can be incorporated into WSDL types and eventually into WSDL messages. The message `buyingProcessInput` in Figure 5 consists of a part whose data type is `complexType Customer`. Djuric developed a profile at the UML meta layer to encompass various ontology related operators, demonstrating the capabilities of the UML language to represent ontologies defined using OWL (Djuric 2004). This work provides evidence that OWL data type definitions with complex characteristics can be expressed using UML.

## 4 System Architecture

Existing processes can be adopted to synthesize knowledge base for configuration from UML and OCL specifications (Felfernig, Friedrich, Jannach & Zanker 2002). Information derived from UML diagrams can be used to create knowledge bases for configuration on the Semantic Web (Felfernig, Friedrich, Jannach, Stumptner & Zanker 2002a, Felfernig, Friedrich, Jannach, Stumptner & Zanker 2002b). Figure 6 shows our system architecture based on afore mentioned findings. The 'Meta layer' in the diagram shows the key components of the system. The 'Instance layer' shows some examples of the instances that fall under specific category. The ArgoUML<sup>5</sup> open source UML modeling tool is used to model the service descriptions. Figure 6 shows the complex constraint (introduced in Section 3.1) modeled in OCL. The constraint instance in KB, shown in the figure, is modeled using ILOG JConfigurator syntax (as specified in (Albert et al. 2005), (Felfernig, Friedrich, Jannach & Zanker 2002)). The translation from UML/OCL models to XMI can be achieved using the 'Export' feature of the ArgoUML tool. The OCL expressions are contained in the XMI file in OCL syntax, which can be easily converted to the input language of a constraint reasoner such as ILOG JConfigurator (Albert et al. 2005). The actual composition is then found by the configurator as demonstrated in (Albert et al. 2005).

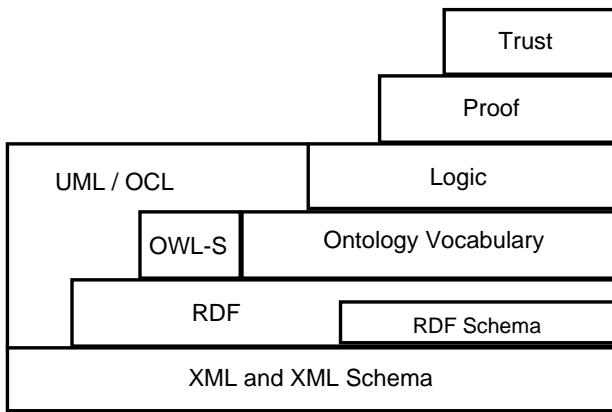


Figure 7: Semantic Web Protocol Stack: UML/OCL

## 5 UML/OCL and Semantic Web Protocol Stack

Earlier work has shown the possibility of transforming UML models into particular styles of Semantic Web specifications. For example, Djuric et al. have investigated the vision of Modeling Spaces (MS) (Djuric et al. 2005) and subsequently established the overlap of Technological Spaces (TS) (Gasevic, Djuric, Devedzic & Damjanovic 2004) between the Model Driven Architecture (MDA) and the Semantic Web. A transformation map between the UML Meta Object Facility (MOF) and RDF(S) within their respective MS was provided in (Djuric et al. 2005). This transformation can be used to map UML models into RDF. RDF Schema can then be used to integrate component specifications (Korthaus, Schwind & Seedorf 2005) defined using UML. However, RDF(S) is too restricted to represent the semantics such as constraints. OCL of course, along with UML are the OMG's intended standard for component specification and can do this with clarity and consistency.

In our approach, the UML Profile defined in (Djuric 2004) is used to define the structural and non-functional properties of a Web Service, while OCL is used for developing semantic descriptions of the functionality, pre conditions, effects of Web Services. The position of UML/OCL in the Semantic Web protocol stack is shown in Figure 7. In Section 3 we examined the aspects of UML and OCL that exhibits better expressiveness, in comparison with OWL-S under certain circumstances, of procedural knowledge and datatype mapping. UML and OCL have sufficient expressiveness to represent critical knowledge that cannot be expressed in OWL-S unless significant changes are adapted in OWL-S ontologies. However, in terms of ontology representations and reasoning capabilities OWL-S clearly shows potential. We acknowledge that OWL-S with suitable modifications will be a front runner for semantic web service descriptions. On the other hand, UML and OCL are widely accepted as a part of the object oriented software engineering approach by both the academic and industrial community. We expect that using standard practices, which have been actively used in software development, will ease the effort for development of ontologies and semantic descriptions for Web Services thereby encouraging development of more ontologies for the Semantic Web by businesses.

<sup>4</sup>XMI is an OMG specification - <http://www.omg.org/cgi-bin/apps/doc?formal/05-09-01.pdf>

<sup>5</sup>ArgoUML - <http://argouml.tigris.org/>.

## 6 Conclusion

In this paper, we have examined the use of UML and OCL for the semantic description of service specifications. We compare the approach to the widely suggested use of Semantic Web technology to support or automate composition or matching tasks in a service-oriented environment. Our approach is based on existing proposals to use UML as an ontology language (Djuric 2004, Gasevic et al. 2005) but the extension to include OCL specifications results in higher expressiveness than traditional Semantic Web formalisms that have shown to exhibit a number of drawbacks (Balzer et al. 2004). The resulting specifications also lend themselves directly to the composition task with constraint-based reasoning engines (Albert et al. 2005, Felfernig, Friedrich, Jannach & Zanker 2002). They therefore combine ease of use, familiarity with traditional notation, and expressive power in one approach.

## References

- Albert, P., Henocque, L. & Kleiner, M. (2005), Configuration-based workflow composition, in 'ICWS', Orlando, FL, pp. 285–292.
- Balzer, S., Liebig, T. & Wagner, M. (2004), Pitfalls of OWL-S: a practical semantic web use case., in M. Aiello, M. Aoyama, F. Curbera & M. P. Papazoglou, eds, 'ICSOC', ACM, pp. 289–298.
- Berners-Lee, T., Hendler, J. & Lassila, O. (2001), 'The semantic web', *Scientific American*.
- Djuric, D. (2004), 'MDA-based Ontology Infrastructure.', *Comput. Sci. Inf. Syst.* 1(1).
- Djuric, D., Gasevic, D. & Devedzic, V. (2005), Adventures in Modeling Spaces: Close Encounters of the Semantic Web and MDA Kinds., in 'International Workshop on Semantic Web Enabled Software Engineering (SWESE)'.
- Falkovych, K., Sabou, M. & Stuckenschmidt, H. (2003), UML for the Semantic Web: Transformation-Based Approaches., in 'Knowledge Transformation for the Semantic Web', pp. 92–106.
- Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M. & Zanker, M. (2002a), Acquiring Configuration Knowledge Bases in the Semantic Web Using UML., in A. Gómez-Pérez & V. R. Benjamins, eds, 'EKAW', Vol. 2473 of *Lecture Notes in Computer Science*, Springer, pp. 352–357.
- Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M. & Zanker, M. (2002b), UML as knowledge acquisition frontend for Semantic Web configuration knowledge bases., in M. Schroeder & G. Wagner, eds, 'RuleML', Vol. 60 of *CEUR Workshop Proceedings*, CEUR-WS.org.
- Felfernig, A., Friedrich, G., Jannach, D. & Zanker, M. (2002), Configuration Knowledge Representation Using UML/OCL., in J.-M. Jézéquel, H. Hußmann & S. Cook, eds, 'UML', Vol. 2460 of *Lecture Notes in Computer Science*, Springer, pp. 49–62.
- Gasevic, D., Djuric, D. & Devedzic, V. (2005), 'Bridging MDA and OWL Ontologies.', *J. Web Eng.* 4(2), 118–143.

- Gasevic, D., Djuric, D., Devedzic, V. & Damjanovic, V. (2004), Approaching OWL and MDA Through Technological Spaces, in 'Third Workshop in Software Model Engineering - WiSME2004'.
- Grønmo, R., Jaeger, M. C. & Hoff, H. (2005), Transformations Between UML and OWL-S., in A. Hartman & D. Kreische, eds, 'ECMDA-FA', Vol. 3748 of *Lecture Notes in Computer Science*, Springer, pp. 269–283.
- Ha, Y. & Lee, R. (2006), Semantic web service modeling using uml for e-business environment., in 'SNPD', IEEE Computer Society, pp. 368–374.
- Kawamura, T., Blasio, J.-A. D., Hasegawa, T., Paolucci, M. & Sycara, K. P. (2004), Public Deployment of Semantic Service Matchmaker with UDDI Business Registry., in S. A. McIlraith, D. Plexousakis & F. van Harmelen, eds, 'International Semantic Web Conference', Vol. 3298 of *Lecture Notes in Computer Science*, Springer, pp. 752–766.
- Koide, S., Aasman, J. & Haflich, S. (2005), OWL vs. Object Oriented Programming., in 'International Workshop on Semantic Web Enabled Software Engineering (SWESE)'.
- Korthaus, A., Schwind, M. & Seedorf, S. (2005), Semantic Integration of Business Component Specifications with RDF Schema., in 'International Workshop on Semantic Web Enabled Software Engineering (SWESE)'.
- Mailharro, D. (1998), 'A classification and constraint-based framework for configuration.', *AI EDAM* **12**(4), 383–397.
- Medjahed, B., Bouguettaya, A. & Elmagarmid, A. K. (2003), 'Composing web services on the semantic web', *The VLDB Journal* **12**(4).
- Petrie, C. J., Genesereth, M. R., Bjornsson, H., Chirkova, R., Ekstrom, M., Gomi, H., Hinrichs, T., Hoskins, R., Kassoff, M., Kato, D., Kawazoe, K., Min, J. U. & Mohsin, W. (2003), Adding AI to Web Services., in L. van Elst, V. Dignum & A. Abecker, eds, 'AMKM', Vol. 2926 of *Lecture Notes in Computer Science*, Springer, pp. 322–338.
- Skogan, D., Grønmo, R. & Solheim, I. (2004), Web Service Composition in UML., in 'EDOC', IEEE Computer Society, pp. 47–57.
- Wu, D., Parsia, B., Sirin, E., Hendler, J. & Nau, D. (2003), Automating DAML-S web services composition using SHOP2, in 'Proc. ISWC', Sanibel Island, FL.

# Semantic Enrichment in Ontologies for Matching

Nwe Ni Tun

Japan Advanced Institute of Science and Technology  
1-1 Asahidai, Nomi, Ishikawa 923-1292, JAPAN  
<http://www.jaist.ac.jp>  
Email: [nitun@jaist.ac.jp](mailto:nitun@jaist.ac.jp)

## Abstract

Matching (or mapping) between heterogeneous ontologies becomes crucial for interoperability in distributed and intelligent environments. Although many efforts in ontology mapping have already been conducted, most of them rely heavily on the meaning of entity names rather than the semantics defined in ontologies. In order to deal with semantic heterogeneity, we enrich the semantics of ontologies for content-based matching. In this paper, we propose a semantically-enriched model of ontologies (called MetaOntoModel) where the semantics of concepts are enriched by adding concept-level knowledge (called meta-knowledge) based on three philosophical notions: *identity*, *rigidity*, and *dependency*. Then, we develop a MetaOntoModel-based ontology matching method. Our novel idea is that if two concepts are semantically equivalent, then they have the same meta-knowledge. On the contrary, if two concepts possess different kinds of meta-knowledge, then they cannot be matched. We prove that meta-knowledge can determine not only the scope of matches, but also the closest corresponding properties between two similar concepts.

**Keywords:** Ontology Model, Semantic Enrichment, Ontology Matching, Semantic Heterogeneity, Interoperability

## 1 Introduction

Today, ontologies have become a silver bullet not only in the development of the Semantic Web, but also in several collaborative application areas such as intelligent environments (or smart spaces), e-commerce, social networks, multi-agent systems, etc., because they are respected as a means of consensus for intelligent reasoning and sharing capabilities. Since a single global ontology is no longer enough to support the variety of tasks pursued in distributed environments, the Web involves a proliferation of ontologies, and faced a trade off between interoperability and heterogeneity.

In order to keep a balance between heterogeneity and interoperability, ontology matching has become a plausible solution in various tasks, such as ontology merging, query answering, information retrieval, exchange, and integration, etc. Heterogeneity is generally distinguished in terms of syntactic heterogeneity and semantic heterogeneity. *Syntactic heterogeneity* is caused by using different ontology modeling

paradigms (e.g., RDF-based model or Frame-based model) and different ontology languages (e.g. DAML or OWL), while *semantic heterogeneity* is created by conceptualization divergence in describing the semantics of ontological classes. Research on resolving syntactic heterogeneity has been undertaken by many researchers so far (Bowers 2000, Chalupsky 2000). In this paper, we focus on the semantic heterogeneity between ontologies. Dealing with semantic heterogeneity is a recurrent issue for ontologies, like the problems related to information integration of heterogeneous databases and systems (Batini 1986, March 1990). Ceri and Widom listed four categories of semantic conflicts concerning schema matching: naming conflicts, domain conflicts, meta-data (or datatype) conflicts, and structural conflicts (Ceri 1993). Visser and colleagues classified ontology mismatches into two levels: conceptualization mismatches (class mismatches and property mismatches) and explication mismatches (abstraction level mismatches and categorization mismatches) (Visser 1997). According to the above works, we classify semantic heterogeneity in ontologies into four categories. For two semantically similar or equivalent classes, there is (a) *terminological* heterogeneity if they have different names or labels; (b) *taxonomical* heterogeneity if they have different subsumption structures; (c) *schematic* heterogeneity if they have different sets of properties and constraints; and (d) *instantiation* heterogeneity if they are interpreted using different sets of instances.

Most mapping tools are mainly intended to solve terminological heterogeneity between lightweight ontologies like Yahoo directories, by applying Natural Language Processing (NLP) techniques (Mitra & Wiederhold 2002, Doan, Madhavan, Domingos & Halevy 2002, Ehrig & Stabb 2004). In practice, a matching process between formally axiomatized ontologies with a variety of heterogeneities is a highly complex process, and a considerable amount of expert-interaction is still involved in verification. For ontology matching, our underlying assumption is *the more explicit semantics is specified in ontologies, the feasibility of matching will be greater*. Hence, an important step in handling semantic heterogeneity should be the attempt to enrich the semantics of concepts, with adequate conceptualization consistency.

The semantic enrichment techniques use a variety of knowledge sources, such as shared thesaurus like WordNet<sup>1</sup>, linguistic knowledge, and intensional and extensional knowledge (Su 2004). However, Mitra and Wiederhold claim that full automation for mapping using linguistic knowledge is not feasible, due to the inadequacy of today's NLP technology (Mitra & Wiederhold 2002). It is also obvious that the semantics of similar concepts described by either intensional knowledge (attributes and relations) or extensional knowledge (sets of instances), in two different ontolo-

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at Australasian Ontology Workshop (AOW 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology, Vol. 72. M. A. Orgun and T. Meyer, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup><http://wordnet.princeton.edu/>

gies, can possibly be heterogeneous according to the diverse knowledge of domain experts.

We focus our enrichment approach on meta-knowledge analysis using some philosophical notions. The idea behind our approach is that the meta-knowledge carries an identifiable link between two heterogeneous descriptions of a concept. Though the description of a concept can be slightly different according to domain experts, the meta-knowledge of the concept is not distinctive for the same semantics. For this purpose, we introduce a semantically enriched model of ontologies (called MetaOnto-Model) where concept-level knowledge (called meta-knowledge) is embedded into ontologies. Our novel idea is that if two concepts are semantically equivalent, then they have the same meta-knowledge, together with similar properties and constraints. On the contrary, if two concepts have different meta-knowledge, then they cannot be matched. We prove that the meta-knowledge can determine not only the scope of matches, but also the closest corresponding properties between concepts.

## 2 A First-order Modal Language in Kripke Semantics

In order to deal with semantic heterogeneity, we adopt three philosophical notions of OntoClean<sup>2</sup> (Guarino & Welty 2001). These notions (called meta-properties in OntoClean) are identity, rigidity, and dependency. Guarino & Welty mentioned that the notions of OntoClean were formalized in *S5* Quantified Modal Logic (QML)<sup>3</sup> with the Barcan formula ( $BF$ )<sup>4</sup>, which gives us a *constant domain* (every object exists in every possible world) and *universal accessibility* (every world is accessible from every other world) (Welty & Andersen 2005). The domain of quantification is *possibilia*, which when combined with *S5* +  $BF$  introduces a need for an *actual existence* predicate ( $E$ ), as opposed to logical existence, that indicates some objects actually exist in the possible worlds (Miller 1987, Cress 2001).

In order to express a precise semantics of each notion, we provide a formal language of first-order Quantified Modal Logic (QML) and apply Kripke's semantics.

### 2.1 Syntax

Let  $\mathcal{L}^E$  be a first-order modal language (Cress. 2001, Belardinelli 2006, Modal 2003) which consists of **alphabet**  $\mathcal{A}^E = \{\mathcal{X}, \mathcal{P}_n, \mathcal{F}_n, E\}$  for countable infinite sets of individual variables,  $n$ -ary predicate symbols,  $n$ -ary function symbols, and the actual existential predicate symbol  $E$ , where  $n$  is a finite natural number. In  $\mathcal{L}^E$ , propositional connectives ( $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$ , and  $\leftrightarrow$ ), quantifiers ( $\forall$  and  $\exists$ ), and modal operators ( $\Box$  and  $\Diamond$ ), are also used. Terms of  $\mathcal{L}^E$  are either constants, variables or constructed terms  $f_n(t_1, \dots, t_n) \in \mathcal{F}_n$  where  $t_1, \dots, t_n$  are terms.

**Definition 1 (Modal Formulas)** Modal formulas in alphabet  $\mathcal{A}^E$  are defined as follows:

- If  $p_n$  is an  $n$ -nary predicate symbol and  $\langle t_1, \dots, t_n \rangle$  is an  $n$ -tuple of terms, then  $p_n(t_1, \dots, t_n)$  is an atomic modal formula.

<sup>2</sup>OntoClean is a domain-independent methodology for ontological analysis—a framework for cleaning taxonomic structure of ontologies.

<sup>3</sup>QML is known for the integration of First-order predicate logic and modal logic.

<sup>4</sup> $\forall x \Box \phi \rightarrow \Box \forall x \phi$  [Barcan formula]

- If  $\phi, \psi$  are modal formulas, then  $\neg \phi$ ,  $\phi \rightarrow \psi$ , and  $\Box \phi$  are modal formulas.
- If  $\phi$  is a modal formula and  $x$  is a variable, then  $\forall x \phi$  is a modal formula.

Falsehood  $\perp$ , propositional connectives  $\wedge$ ,  $\vee$ ,  $\leftrightarrow$ , existential quantifier  $\exists$ , and modal operator  $\Diamond$ , are defined in the usual way (Modal 2003).

### 2.2 Semantics

A **kripke frame** in QML is  $F = \langle W, R \rangle$  where  $W$  is a non-empty set, and  $R$  is a binary relation on  $W$ . Set  $W$  is intuitively interpreted as the domain of *possible worlds*, whereas  $R$  is the *accessibility relation* between worlds (Cress. 2001). **Universe**  $U$  includes a set of individuals  $U_{ind}$  and a set of datatype values  $U_{dtp}$ , such that  $U = U_{ind} \cup U_{dtp}$ , regarding owl:ObjectProperty and owl:DatatypeProperty<sup>5</sup>.

**Definition 2 (Kripke Model)** A Kripke model given in universe  $U$ , is a quintuple  $M = \langle F, D, d, \mathcal{I} \rangle$  where  $F$  is a Kripke frame,  $D$  is a function assigning a non-empty set (called outer domain)  $D(w) \subseteq U$  to every  $w \in W$ ,  $d$  is a function assigning an inner domain to every  $w \in W$  such that  $d(w) \subseteq D(w)$ , and  $\mathcal{I} = \langle U, \cdot^I \rangle$  is the interpretation in frame  $F$  such that  $\mathcal{I}(p_n^I, w) \subseteq D(w)^n$  for any  $n$ -ary predicate  $p_n \in \mathcal{P}_n$ ,  $\mathcal{I}(f_n^I, w) : D(w)^n \rightarrow D(w)$  for any  $n$ -ary function  $f_n \in \mathcal{F}_n$ , and  $\mathcal{I}(E^I, w) = d(w)$  for existential predicate  $E$ .

Each *outer domain*  $D(w)$  contains the objects which it makes sense to talk about the possible domain of  $w$ , on the other hand in each *inner domain*  $d(w)$  there appear individuals actually existing in  $w$ . We assume that model  $M$  satisfies the *inclusion requirement* (Cress. 2001), that is, if  $wRw'$  then  $D(w) \subseteq D(w')$ . As frame  $F$  employs *S5*<sup>6</sup>, there is a constant outer domain between possible worlds such that  $D(w) = D(w')$ . In practice, we cannot expect that the same individuals actually exist in each arbitrary accessible world. Therefore, we regard that the inner domain of each world varies, depending on the actual existence of individuals in the world.

**Definition 3 ( $w$ -assignment)** To define truth conditions for atomic and quantified formulas with variables  $x \in X$  given in  $\mathcal{L}^E$ ,  **$w$ -assignment function**  $\partial$  into interpretation  $\mathcal{I}$  in world  $w$  is defined as  $\mathcal{I}^\partial(x^I, w) = \partial(x)$ . There is also a variant of  $w$ -assignment,  $\partial^{x,a}$ , which assigns individual element  $a \in D(w)$  to  $x$ .

**Definition 4 (Satisfaction)** For any world  $w \in W$  given in Kripke model  $M$ , the satisfaction relation of modal formulas with respect to  $\mathcal{I}^\partial$  is as follows:

- $(\mathcal{I}^\partial, w) \models p_n(t_1, \dots, t_n)$  iff  $\langle \mathcal{I}^\partial(t_1^I, w), \dots, \mathcal{I}^\partial(t_n^I, w) \rangle \in \mathcal{I}^\partial(p_n^I, w)$
- $(\mathcal{I}^\partial, w) \models \neg \psi$  iff  $(\mathcal{I}^\partial, w) \not\models \psi$
- $(\mathcal{I}^\partial, w) \models \phi \rightarrow \psi$  iff  $(\mathcal{I}^\partial, w) \models \neg \phi$  or  $(\mathcal{I}^\partial, w) \models \psi$
- $(\mathcal{I}^\partial, w) \models \Box \phi$  iff for every  $w' \in W$  such that  $wRw'$ ,  $(\mathcal{I}^\partial, w') \models \phi$

<sup>5</sup>owl:ObjectProperty relates two individuals, but owl:DatatypeProperty relates an individual and a datatype value (see <http://www.w3.org/TR/owl-guide/>).

<sup>6</sup>S5 is a system where accessibility relation  $R$  is reflexive:  $\Box \phi \rightarrow \phi$ , symmetric:  $\phi \rightarrow \Box \Diamond \phi$ , and transitive:  $\Box \phi \rightarrow \Box \Box \phi$ .

- $(\mathcal{I}^\partial, w) \models \forall x\phi$  iff for every individual  $a \in D(w)$ ,  $(\mathcal{I}^{\partial x, a}, w) \models \phi$

Truth conditions for modal formulas containing propositional connectives  $\wedge, \vee, \leftrightarrow$ , existential quantifier  $\exists$ , and modal operator  $\Diamond$  are defined from the formulas above in the usual way, such as  $\Diamond\phi = \neg\Box\neg\phi$ ,  $\phi \leftrightarrow \psi = \phi \rightarrow \psi \wedge \psi \rightarrow \phi$ . In particular  $(\mathcal{I}^\partial, w) \models \perp$  never holds. A modal formula is true in Kripke model  $M$  if and only if it is true in every possible world  $w \in W$  of  $M$ . Similarly, a modal formula is valid in Kripke frame  $F$  if and only if it is true in every Kripke model  $M$  given on  $F$ . Since language  $\mathcal{L}^E$  contains existence predicate  $E$ , the following two axioms are applied (Cress. 2001, Belardinelli. 2006).

- $\forall x[\phi \rightarrow (E(y) \rightarrow \phi[x/y])] \text{ [E-exemplification]}^7$
- $\frac{\phi \rightarrow (E(x) \rightarrow \psi)}{\phi \rightarrow \forall x\psi}$  where  $x$  is not free in  $\phi$  [Universal E-instantiation]

The soundness and completeness of QML have been proved by Corsi and Belardinelli (Corsi 2002, Belardinelli. 2006).

### 3 Identity, Rigidity, and Dependency

In the philosophical literature, ontological concepts are generally divided into two categories: sortal concepts (called sorts) and non-sortal concepts.

“**Sort** is an entity type<sup>8</sup> that carries a criteria for determining the individuation, persistence, and identity<sup>9</sup> of its instances (Guizzardi, Wagner & Sinderen 2004)”.

“A class is called a **sort** if it supplies or carries an Identity Condition (IC)” (Guarino & Welty 2001).

“No entity without identity” (Quine 1969).

According to the above-quoted statements, it is significant that the principles of identity and individuation supplied by sorts are essential in conceptual modeling, together with a universe of discourse. Therefore, Guizzardi and Wagner (Guizzardi et al. 2004) made the following postulate.

“Every object in a conceptual model (CM) of a domain must be an instance of a CM-class representing a sortal” (Guizzardi et al. 2004).

In this paper, we follow to the above postulate and treat ontological classes as sorts, and non-sortals as the attribute values of sorts. Some examples of sorts are **Person**, **Planet**, **Dog**, **House**, **Student**, **Wine**, **Book**, and **Car**, where individuals (or instances) are countable and identifiable. Unlike sorts, **Red**, **Happy**, and **Beautiful**, are non-sortals, which do not supply identity for their individuals. However, whether a concept is a sort or not should rely on possession of identity criteria, rather than the common sense of a concept’s name.

**Identity** is the logical relation of sameness, in which an individual identifies only to itself globally.

<sup>7</sup>For a proof for  $(\mathcal{I}^\partial, w) \models \phi[x/y]$  iff  $(\mathcal{I}^{\partial x, \partial(y)}, w) \models \phi$ , we refer to (Belardinelli. 2006).

<sup>8</sup>Entity type has an extension (instances) and an intension which includes an applicability criteria for determining whether an entity is an instance of it.

<sup>9</sup>An identity criteria (also called identity Condition) supports the judgment of whether two particulars describe the same entity or not.

**Definition 5 (Identity Condition)** *Identity Condition (IC) of a sort is a datatype property, which provides a unique IC value to each individual of the sort. Formally, if  $\iota$  (unary function of language  $\mathcal{L}^E$ ) is an IC of sort  $s$  (denoted by  $p_s$ ), then it satisfies one of the following conditions<sup>10</sup>.*

$$\begin{aligned} \Box \forall x, y [p_s(x) \wedge E(x) \wedge p_s(y) \wedge E(y) \wedge x = y \rightarrow \iota(x) = \iota(y)] & \quad (1) \\ \Box \forall x, y [p_s(x) \wedge E(x) \wedge p_s(y) \wedge E(y) \wedge \iota(x) = \iota(y) \rightarrow x = y] & \quad (2) \end{aligned}$$

Equation (1) states that “The IC of a sort must necessarily provide the same IC value for the same individual of the sort”. Equation (2) states that “The IC of a sort must be necessarily sufficient to determine two individuals with the same IC value as the same individual”.

**Example 1** Suppose that *hasISBN* is the IC of sort **PublishedBook**. Then, it is necessary to have the same ISBN for the same published book, or two individual books with the same ISBN can be identified as the same published book in every possible world. Someone may use a global product bar-code to identify each copy of the same **PublishedBook** (say an individual of **PublishedBookCopy**). For other examples, *hasFingerprint*, *hasURL*, and *hasLatitudeLongitude* can be used as the ICs of **Person**, **WebResource**, and **Location**, respectively. Note that ICs should be globally identifiable for individuals. For example, **Student** possesses property ‘*hasStudentID*’, however it is world-variant and can not be used as an IC. We call it *local IC*, and use it to identify individuals inside a possible world.

In Definition 5, we use unary predicates of language  $\mathcal{L}^E$ , by adding predicated names corresponding to sort names, such as  $p_s \in \mathcal{P}_1$ . Then, the fundamental semantics of a **subsumption relationship**  $\sqsubseteq$  between two sorts  $s_1$  and  $s_2$ , can be interpreted in the form of implication relation, that is, if  $s_2 \sqsubseteq s_1$  then  $\forall x [p_{s_2}(x) \rightarrow p_{s_1}(x)]$  (Beierle 1992, Kaneiwa 2001). This is read as “If sort  $s_2$  is subsumed by sort  $s_1$  then every individual of  $s_2$  is an individual of  $s_1$ ”. In this case,  $s_1$  is a *super-sort* and  $s_2$  is a *sub-sort*. The IC of a sort allows inheritance through subsumption relationships.

**Definition 6 (OwnIC and CarriedIC)** *If sort  $s$  originates an IC, then the IC is called the ownIC of  $s$  denoted by  $\iota_s$ . If a sort inherits an IC from a super-sort through subsumption relationship, then the IC is called “carriedIC” denoted by  $\iota$ .*

**Example 2** Suppose that *hasFingerPrint* is the ownIC of sort **Person** because every person is identifiable by such fingerprint. According to **Student**  $\sqsubseteq$  **Person**, *hasFingerprint* is a carriedIC for **Student**. In this case, we say **Person** supplies its ownIC to **Student** and **Student** carries the IC of **Person**.

**Rigidity** provides the modality of a sort. In general, the rigid designation in modal context is “*it designates the same thing in all possible worlds*”. However, Kai-Yee Wong (Wong 2003) mentioned that what Saul Kripke likes to say about rigidity is with existence conception: “...a designator rigidly designates a certain object if it designates that object wherever the object exists” (Kripke 1971). Regarding this quoted reference, we apply the actual existence of rigidity (Welty & Andersen 2005).

<sup>10</sup>In OntoClean, Guarino & Welty used a non-modal time parameter ‘ $t$ ’ to mention a time line in each possible world. In our definition of IC, we omit time parameter ‘ $t$ ’ of original definitions (Guarino & Welty 2001) by considering every state of possible affairs by time, space, etc., as possible worlds in modalities, and make IC explicit as a datatype property of a sort.

**Definition 7 (Existential Rigidity)** For any sort  $s$ ,  $s$  is existentially rigid iff

$$\forall x[\Diamond p_s(x) \rightarrow \Box(E(x) \rightarrow p_s(x))], \quad (3)$$

otherwise,  $s$  is existentially anti-rigid iff

$$\forall x[\Diamond p_s(x) \rightarrow \Diamond(E(x) \wedge \neg p_s(x))]. \quad (4)$$

In the rigid case, if every individual of a sort in world  $w$  exists in every world  $w'$  such that  $wRw'$ , the individual is always a member of the sort. In the anti-rigid case, this is not so.

**Example 3** We can define **Person** as a rigid sort and **Student** as an anti-rigid sort, by expecting every person is a person in every possible world if s/he exists there, and a person is not always a student.

**Dependency** expresses the external dependent relation of a certain sort to another disjoint sort.

**Definition 8 (Externally Dependent)** Sort  $s$  is externally dependent on another sort  $s'$  if, for all individuals  $x$  of  $s$ , necessarily some individual  $y$  of  $s'$  exist, which is neither a part nor a constituent of  $x$ :

$$\forall x[\Box[p_s(x) \wedge E(x) \rightarrow \exists y(p_{s'}(y) \wedge E(y) \wedge p_d(x, y)) \wedge \neg \text{Part}(y, x) \wedge \neg \text{Constituent}(y, x)]] \quad (5)$$

where  $s$  and  $s'$  are disjoint:  $\forall x[p_s(x) \rightarrow \neg p_{s'}(x)]$ .

We make explicit the original dependent definition (Guarino & Welty 2001) by adding an External Dependency Relationship (EDR) denoted by  $p_d$ .

**Example 4** **Student** is externally dependent on **School** with “EnrollIn” relationship. This means we do not define a person who does not enroll in a school as a student. Thus, **EnrollIn** is called the EDR of **Student** on **School**. Similarly **Parent**, **Child**, **Customer**, and **Supplier**, are externally dependent sorts.

There is an issue: ICs are either *intrinsic* or *extrinsic*. Guarino & Welty discussed this issue as follows:

“Global unique IDs are used either in object-oriented systems to uniquely identify an object or in database systems to identify data records. Our notion of IC is based mainly on intrinsic properties. However, this is not to say that the former type never uses intrinsic properties nor the latter never uses extrinsic ones. In practice, conceptual modellers may need both” (Guarino & Welty 2001).

Therefore, in this research, both intrinsic and extrinsic properties are used as ICs if they satisfy Equation (1) or (2). For examples, fingerprint is intrinsic but ISBN is rather extrinsic.

#### 4 Modeling Semantically-Enriched Ontologies

Regarding both Frame and OWL specifications, there are three fundamental modeling components in developing ontologies: *classes* for concepts, *properties* for attributes and relations (called intensional knowledge) of concepts, and *individuals* for instances (called extensional knowledge) of each concept. In addition, ontological axioms and constraints can be defined on classes and properties. *Taxonomy* is a structure of classes mainly with subsumption relationships. An ontology with a universe of discourse constitutes a *populated ontology* or ontology base.

Let  $S$  be a set of sorts. A set of properties belonging to any sort  $s \in S$  is denoted as  $P^D(s)$ . For example,  $P^D(\text{Person}) =$

$\{\text{hasName}, \text{hasMother}, \text{hasDOB}, \text{hasFingerPrint}\}$ . For every property  $p \in P^D(s)$ , there is a specific domain ( $\mathcal{D}_s$ ) and range ( $\mathcal{R}_p$ ) such that  $p: \mathcal{D}_s \rightarrow \mathcal{R}_p$ . We divide  $P^D(s)$  into two subsets: (a) individual properties that relate two individuals, e.g., **hasMother**, and (b) datatype properties that relate an individual to a datatype value, e.g., **hasName**, **hasDOB**, **hasFingerPrint**. IC is distinguished from other properties by one-to-one relationship between its domain and range, e.g., **hasFingerPrint**. The subsumption relationship allows the inheritance of properties between sorts, that is, if  $s_2 \sqsubseteq s_1$  then  $P^D(s_2) \supseteq P^D(s_1)$ . A hierarchy of sorts with subsumption relationships is called a **sortal taxonomy**  $\langle S, \sqsubseteq \rangle$  where  $S$  is a set of sorts and  $\sqsubseteq$  is a collection of subsumption relationships on  $S$ .

The objective of modeling semantically enriched ontologies is to provide a well-structured taxonomy and adequate semantics for ontologies. Therefore, we provide a classification of sorts according to this classification. Then, we define a conceptual model (called **MetaOntoModel**) of semantically enriched ontologies. Based on the classification of sorts in **OntoClean** and by Guizzardi (Guizzardi et al. 2004), we define four categories of sort: **type\_sort**, **quasi-type\_sort**, **role\_sort**, and **phase\_sort**.

**Definition 9 (Type\_sort)** If a sort is existentially rigid and it originates (or supplies) an IC, then the sort is called a **type\_sort**.

Some examples of **type\_sort** are **Person**, **PublishedBook**, and **Wine**, with ICs **hasFingerPrint**, **hasISBN**, and **hasWineName**<sup>11</sup>, respectively.

**Definition 10 (Quasi-type\_sort)** If a sort is existentially rigid but it does not originate an IC, then it is called a **quasi-type\_sort**.

More precisely, a quasi-type sort is a partition<sup>12</sup> of a **type\_sort**, specialized with a Common Value Attribute (CVA)—an attribute of a sort which has a common attribute value for every individual of the sort, e.g., every red wine has red color, the gender of every man is male. For example, (a) **MalePerson** and **FemalePerson** are the quasi-type sorts of **Person**; (b) **RedWine** and **WhiteWine** are of **Wine**; (c) **PublishedBookInLogic** and **PublishedBookInNon-Logic** are of **PublishedBook**. The quasi-type sorts of a certain **type\_sort** are disjoint to each other, that is, if an individual person is modeled as an instance of **MalePerson** then he cannot be an instance of **FemalePerson**.

**Definition 11 (Role\_sort)** If a sort is existentially anti-rigid and it is externally dependent on another sort by holding an EDR, then the sort is called a **role\_sort**. Moreover, the domains of role sorts are not necessarily disjoint.

**Student**, **Employee**, **Customer**, and **Supplier**, are examples of **role\_sorts**, such as a customer is a person who buys a product from a supplier, an employee is a person who is hired by an organization to perform a job. An individual can be a member of more than one **role\_sort** subsumed by the same **type\_sort**, that is, a person can be a student as well as an employee.

**Definition 12 (Phase\_sort)** If a sort is existentially anti-rigid and does not need an EDR like

<sup>11</sup>hasWineName includes winery, appellation, and a vintage.

<sup>12</sup>The partitions (sorts) of a **type\_sort** form a complete generalization and they are disjoint from each other.



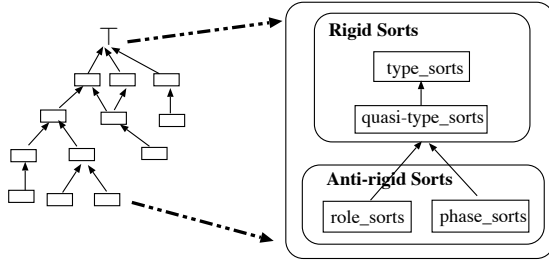


Figure 1: A typical structure of sortal taxonomy

*role\_sort*, then the sort is called a *phase\_sort*. *Phase\_sorts* constitute possible stages in the history of a super-sort they specialize, by holding a *Common Constraint (CC)*. Thus, they are disjoint to each other.

Some examples are: (a) *Girl*, *Teenager*, and *Woman*, as the possible stages of *FemalePerson* by holding age constraint; (b) *Caterpillar* and *Butterfly*, of a *Lepidopteran* by holding wing constraint; (c) *UndergraduateStudent*, *MasterStudent*, and *DoctoralStudent*, of a university student life by holding degree constraint. Contrary to *role\_sort*, an individual cannot belong to more than one *phase\_sort*.

According to the above definitions,  $S$  can be divided into four subsets:

$$S = S_{\text{type}} \cup S_{\text{quasi-type}} \cup S_{\text{role}} \cup S_{\text{phase}}$$

where  $S_{\text{type}}$  is a set of *type\_sorts*,  $S_{\text{quasi-type}}$  is a set of *quasi-type\_sorts*,  $S_{\text{role}}$  is a set of *role\_sorts*,  $S_{\text{phase}}$  is a set of *phase\_sorts*. We claim that each subset of  $S$  is disjoint to each other, because their modality and identifiable characteristics are different. This disjointness is proved as follows.

- By Definition (7), if sort  $s \in S$  is existentially anti-rigid, then  $s$  is not rigid, and vice versa.
- By Definition (9) & (10), if sort  $s \in S$  is a quasi-type sort, then it is not a type\_sort, and vice versa.
- by Definition (11) & (12), if sort  $s \in S$  is a phase sort, then it is not a role\_sort, and vice versa.

A typical structure of the above classification is depicted in Figure 1. It can also be called a skeleton of sortal taxonomies which preserve the condition “*anti-rigid sorts never subsume rigid sorts*” (Guarino & Welty 2001). We do not mean that every ontology needs to complete this classification. In addition, we employ the two assumptions described below.

- **Assumption1:** Every top-most sort of a sortal taxonomy in a given ontology must be a *type\_sort* which originates (or supplies) an IC to identify an individual globally in multiple worlds, regarding identity for every individual.
- **Assumption2:** A *type\_sort* is not allowed to have multiple subsumption relationships,  $s_3 \not\sqsubseteq s_1$  and  $s_3 \not\sqsubseteq s_2$  where all are *type\_sorts*, because no individual possesses two incompatible IC values (e.g. an alcoholic drink cannot be defined as both wine and whisky).

**Definition 13** *Meta-knowledge*, denoted by  $P^M$ , is a sort-level (functional) property that defines a meta-knowledge value for each sort. The range of  $P^M$  is restricted by an enumerated set  $\{“type”, “quasi-type”, “role”, “phase”\}$ . For sort  $s \in S$ , the meta-knowledge of  $s$  is denoted by  $P^M(s)$ .

Table 1: The meta-knowledge (M-K) and conceptual constraints by sort classification

Classification	M-K	Constraints
type_sort	“type”	ownIC: $\iota_s \in P^D(s)$
quasi-type_sort	“quasi-type”	CVA: $p_a \in P^D(s)$
role_sort	“role”	EDR: $p_d \in P^D(s)$
phase_sort	“phase”	CC: $p_c \in P^D(s)$

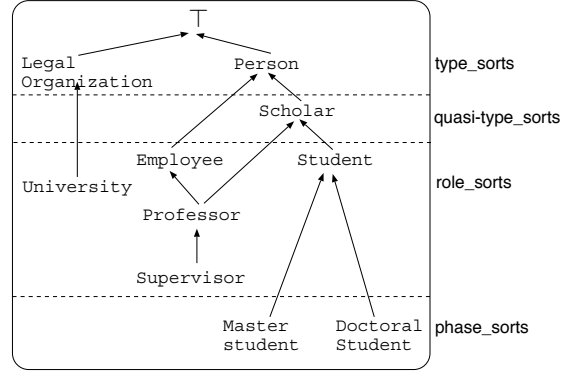


Figure 2: The taxonomy of Ontology1

For example,  $P^M(\text{Person}) = “type”$ . Each sort possesses only one meta-knowledge. Moreover, the meta-knowledge of a sort does not allow inheritance through subsumption relationships. According to the classification of sorts, we define the meta-knowledge of a sort together with a specific conceptual constraint, as listed in Table 1. This classification supports not only subsumption consistency among sorts, but also adequate individual-level properties for the precise semantics of sorts.

#### Definition 14 (MetaOntoModel)

*MetaOntoModel* is a quintuple  $O = \langle S, \sqsubseteq, P^M, P^D, A \rangle$  where  $S$  is a non-empty set of sorts,  $\langle S, \sqsubseteq \rangle$  is a taxonomic structure of  $S$  with subsumption relationship  $\sqsubseteq$ ,  $P^M$  is a function that defines the meta-knowledge of each sort  $s \in S$ ,  $P^D$  is a set of domain-level properties such that  $P^D = \{P^D(s) \mid s \in S\}$ , and  $A$  is a set of ontological axioms and conceptual constraints.

**Example 5 (Ontology1)** We present a simple ontology of people in university domain (Ontology1) based on the *MetaOntoModel* as follows:

$S = \{\text{Organization, University, Person, Scholar, Employee, Professor, Supervisor, Student, MasterStudent, DoctoralStudent}\}$

$P^M(S) = \{type, role, type, quasi-type, role, role, role, role, phase, phase\}$

$S_{\text{type}} = \{\text{Organization, Person}\}$

$S_{\text{quasi-type}} = \{\text{Scholar}\}$

$S_{\text{role}} = \{\text{University, Employee, Professor, Supervisor, Student}\}$

$S_{\text{phase}} = \{\text{MasterStudent, DoctoralStudent}\}$

The set of individual-level properties  $P^D$  with specific domains and ranges, is given in Table 2, where  $\{S, T, R\}$  denotes  $\{\text{study, teaching, research}\}$ . A certain university will be a model of this ontology. We assume that every ontological entity (class, property, individual) has a unique name, in each ontology.

The taxonomic structure of Ontology1 is described in Figure 2. In Ontology1, *Legal Organization*

Table 2: The individual-level properties of Ontology1

Property	Domain	Range
hasOrgName	Organization	String
hasLocation	Organization	String
giveDegree	University	Degree
hasName	Person	String
hasBirthDate	Person	Date
hasFingerPrint	Person	String
hasActivity	Scholar	{S,T,R}
hasActivity	Student	{S,R}
hasActivity	Professor	{T,R}
enrollIn	Student	University
hasStudentID	Student	Integer
workIn	Employee	University
enrollForDegree	MasterStudent	M.S
enrollForDegree	DoctoralStudent	PhD
supervise	Supervisor	Student
is-supervised	Student	Supervisor

is defined as a *type\_sort* by expecting every school, institute, or company, is a legal organization until its registration is valid with a unique name or ID, and *University* is defined as a *role\_sort* by considering a certain university may switch its status to another (college or vocational institute) with a relevant change in education standards and policy. *MasterStudent* and *DoctoralStudent* are classified as *phase\_sorts* because they are considered as the possible stages (or phases) of student life. In an alternative ontology, suppose Ontology2, these concepts may be defined including more than one kind of semantic heterogeneity—including use of the same name with different semantics, or different names with the same semantics. Recall that deciding whether a sort is a *type\_sort* or another kind of sort, does not fully depend on the common sense of its name. More precisely, a sort is classified according to the properties and constraints defined for it. Two similar domain ontologies may have different taxonomies with some common sorts. However, we claim that the meta-knowledge of a semantically common sort in both ontologies should be the same. On the contrary, if two sorts have different kinds of meta-knowledge, then they cannot be the same sort, because their semantics have different conceptual constraints such as *ownIC*, *CVA*, *EDR*, or *CC*.

## 5 Implementing MetaOntoModel-based Ontologies

We define semantic enrichment as a process to provide adequate semantics for ontological concepts by developing structured and consistent taxonomies. In this section, we demonstrate an implementation framework of MetaOntoModel-based ontologies using Protégé OWL API and a representation of these ontologies in OWL-Ontology Web Language.

Protégé<sup>13</sup> is a Java-based free open source ontology editor and knowledge base framework, that provides a plug-and-play environment for rapid prototyping and application development. The Protégé platform supports two main ways of modeling ontologies, via Protégé-Frames and Protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. Moreover, Protégé-OWL editor supports creation of customized *meta-classes*<sup>14</sup>. These are the basic reasons why we selected Protégé-OWL editor for the im-

<sup>13</sup><http://protege.stanford.edu/>

<sup>14</sup>A meta-class is a frame template that is used to define new classes in an ontology.

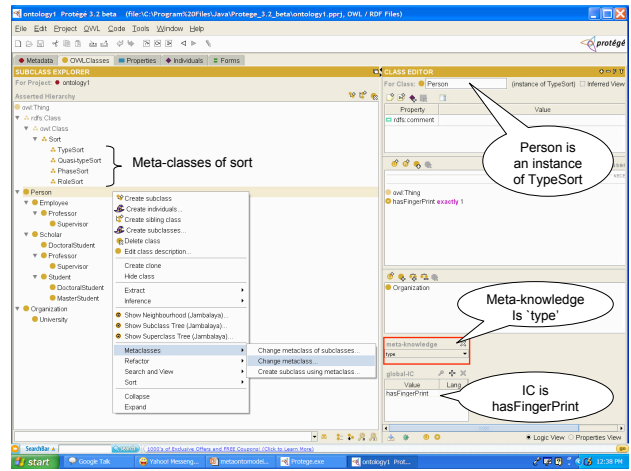


Figure 3: A screenshot of semantic enrichment in Protégé

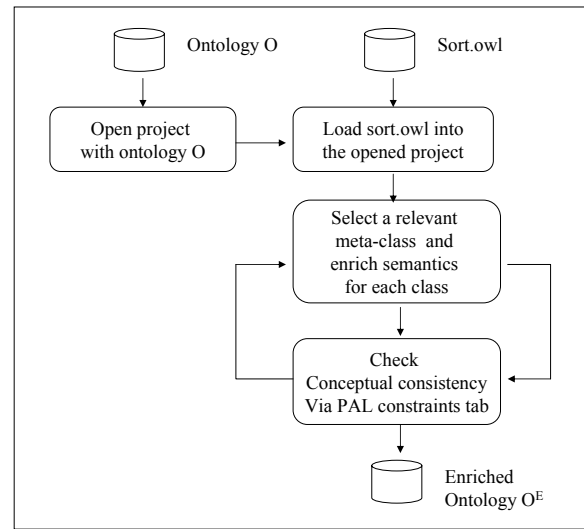


Figure 4: Major steps of semantic enrichment process

plementation of MetaOntoModel-based ontologies. In addition, Protégé supports Protégé Axiom Language (PAL) to create internal constraints, and to embed these constraints in OWL format.

We built a MetaOntoModel-based meta-class ontology named 'sort.owl', and uploaded it in Protégé ontology library<sup>15</sup> as an open source ontology. Our meta-class ontology consists of four meta-classes labeled *TypeSort*, *Quasi-typeSort*, *RoleSort*, and *PhaseSort*, respectively, as shown in Figure 3. Each meta-class has two concept-level properties: 'meta-knowledge' and 'conceptual constraint' shown in Table 1. We also develop five PAL constraints in 'sort.owl' for the purpose of subsumption consistency. The meanings of these PAL constraints are (a) a *quasi-type\_sort* never subsumes a *type\_sort*; (b) a *phase\_sort* never subsumes a *type\_sort*; (c) a *phase\_sort* never subsumes a *quasi-type\_sort*; (d) a *role\_sort* never subsumes a *type\_sort*; and (e) a *role\_sort* never subsumes a *quasi-type\_sort*.

The major steps of our semantics enrichment process are illustrated in Figure 4.

1. First, users need to open a project in Protégé for OWL ontology *O*.

<sup>15</sup><http://protege.cim3.net/cgi-bin/wiki.pl?ProtegeOntologiesLibrary>

```

<rdf:RDF xml:base="http://www.owl-ontologies.com/ontology1.owl"><owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://protege.stanford.edu/plugins/owl/protege/"></owl:Ontology>
  <owl:Class rdf:ID="TypeSort">
    <rdfs:subClassOf><owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
      <owl:onProperty><owl:DatatypeProperty rdf:ID="global-IC"></owl:onProperty></owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf><owl:Restriction>
      <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string">type</owl:hasValue>
      <owl:onProperty><owl:FunctionalProperty rdf:ID="meta-knowledge"></owl:onProperty>
    </rdfs:subClassOf></rdfs:subClassOf>
  </owl:Class>
  <TypeSort rdf:ID="Person">
    <rdfs:subClassOf><owl:Restriction>
      <owl:onProperty><owl:DatatypeProperty rdf:ID="hasFingerPrint"></owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    </owl:Restriction></rdfs:subClassOf>
    <meta-knowledge rdf:datatype="http://www.w3.org/2001/XMLSchema#string">type</meta-knowledge>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing">
    <global-IC rdf:datatype="http://www.w3.org/2001/XMLSchema#string">hasFingerPrint</global-IC>
    <owl:disjointWith rdf:resource="#University">
  </TypeSort>
  <RoleSort rdf:ID="Student">
    <rdfs:subClassOf rdf:resource="#Scholar">
    <rdfs:subClassOf><owl:Restriction><owl:onProperty rdf:resource="#Enrollin">
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
    </owl:Restriction></rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#Professor">
    <externalDependentRelation rdf:datatype="http://www.w3.org/2001/XMLSchema#string">role</externalDependentRelation>
    <meta-knowledge rdf:datatype="http://www.w3.org/2001/XMLSchema#string">role</meta-knowledge>
  </RoleSort>
  <owl:FunctionalProperty rdf:about="#meta-knowledge">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty">
    <rdfs:range><owl:DataRange><owl:oneOf rdf:parseType="Resource">
      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">type</rdf:first>
      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">quasi-type</rdf:first>
      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">phase</rdf:first>
      <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string">role</rdf:first>
    </owl:oneOf></owl:DataRange></rdfs:range></owl:FunctionalProperty>
  </owl:FunctionalProperty>
  <owl:DatatypeProperty rdf:about="#hasFingerPrint">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string">
    <rdfs:domain rdf:resource="#Person">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty">
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:ID="Enrollin">
    <rdfs:range><RoleSort rdf:ID="University"><rdfs:subClassOf><TypeSort rdf:ID="Organization">
    <rdfs:domain><RoleSort rdf:ID="Student"><owl:disjointWith><RoleSort rdf:ID="Professor">
    </RoleSort></rdfs:domain>
  </owl:ObjectProperty>
  <protege:PAL-CONSTRAINT rdf:ID="PAL-CONSTRAINT_2">
    <protege:PAL-NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      notRoleToType</protege:PAL-NAME>
    <protege:PAL-STATEMENT rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      (forall ?sub
        (orall ?super
          (=> (and ('meta-knowledge' ?super "role")
            (subclass-of ?sub ?super)
            (own-slot-not-null 'meta-knowledge' ?sub)))
          (not ('meta-knowledge' ?sub "type")))))
    </protege:PAL-STATEMENT>
  </protege:PAL-CONSTRAINT>

```

Figure 5: The enriched Ontology1 generated in OWL

- Second, it is necessary to load sort.owl into the opened project via the import service of Protégé.
- Third, the meta-class of each ontological class needs to be changed from standard class, `owl:class`, to one of the meta-classes via 'change metaclass' option of Protégé as shown in Figure 3. For this selection, users need the background knowledge of sort classification that we presented in Section 4. By the selection of meta-class, the meta-knowledge of each ontological class will be assigned automatically. Then, the user needs to enrich the semantics of each ontological class by providing necessary individual-level properties.
- Fourth, the consistency of semantic enrichment can be evaluated by invoking the PAL constraints defined in 'sort.owl', via the PAL constraints tab of Protégé, and by running a DIG reasoner Racer<sup>16</sup> or Pellet<sup>17</sup>. An iterated process may be needed between Steps 3 and 4.
- Finally, the semantically-enriched ontology,  $O^E$ , can be successfully generated in OWL.

Figure 3 shows a screenshot of semantic enrichment in Protégé. A part of enriched Ontology1 generated in OWL is shown in Figure 5, where each meta-class is represented using `owl:class` and each sort  $s \in S$  is represented as an instance of a relevant Sort meta-class, e.g., `Person` is a sort which is an instance of meta-class `TypeSort`.

<sup>16</sup><http://www.sts.tu-harburg.de/~r.f.moeller/racer/>

<sup>17</sup><http://www.mindswap.org/2003/pellet/>

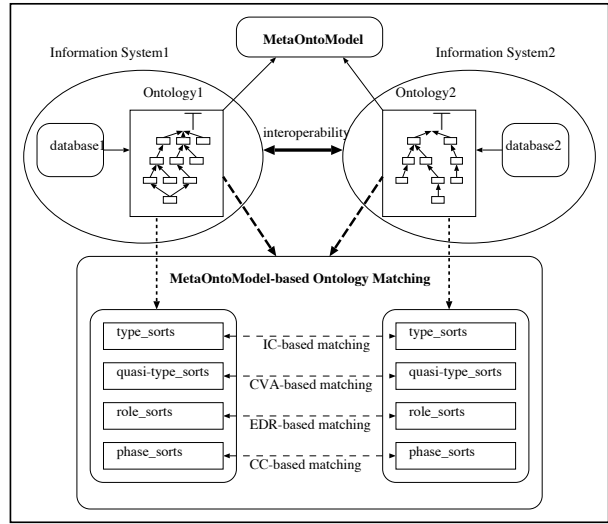


Figure 6: A general architecture of MetaOntoModel-based ontology matching

## 6 MetaOntoModel-based Ontology Matching

A general architecture of MetaOntoModel-based ontology matching is given in Figure 6. Suppose that there are two heterogeneous ontologies: Ontology1 and Ontology2, which are enriched in the form of MetaOntoModel, and also populated. In order to achieve interoperability between two information systems through heterogeneous ontologies, the responsibility of ontology matching is to find semantically similar sorts between two ontologies.

### 6.1 Matching Method

*For ontology matching, we claim that there is no semantic correspondence between rigid sorts and anti-rigid sorts, nor between rigid sorts (type\_sorts and quasi-type\_sort) nor between anti-rigid sorts (role\_sorts and phase\_sorts), because their modality and conceptual constraints are different.*

Thus, our matching process is driven by direct sort matching between the same meta-knowledge groups. Consequently, it can flatten iterations of a matching process and possibly reduce complexity.

Let  $O = \langle S, \sqsubseteq, P^M, P^D, A \rangle$  and  $O' = \langle S', \sqsubseteq', P'^M, P'^D, A' \rangle$  be the logical view of Ontology1 and Ontology2. In our matching method, we consider mapping function  $f : s \in O \rightarrow s' \in O'$  to find the semantically corresponding sort  $s'$  for  $s$ . Then, we divide mapping function  $f$  into four sub-functions as follows:

- typeMatching* is a mapping function that finds correspondence of a type\_sort  $s \in S_{type}$  in  $S'_{type}$ . Since each type\_sort originates an IC, the main idea in determining correspondence between type\_sorts is analyzing whether the own-ICs of two type\_sorts can export and import interchangeably or not (Tun & Tojo 2005).

**Exportability:** If the ownIC of sort  $s \in S_{type}$ ,  $\iota_s$ , can identify and distinguish all the individuals of another sort  $s' \in S'_{type}$ , then the IC is called *exportable* to  $s'$ , formally,  $\forall x, y [p_{s'}(x) \wedge p_{s'}(y) \wedge x = y \rightarrow \iota_s(x) = \iota_s(y)]$ .

**Importability:** If the ownIC of a sort  $s' \in S'_{type}$ ,  $\iota_{s'}$ , can identify and distinguish all the individuals defined for sort  $s \in S_{type}$ , then the IC

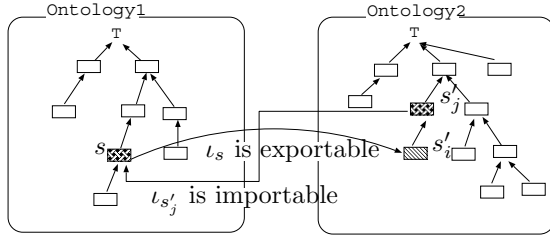


Figure 7: IC-based type\_sort matching

is called *importable* to  $s$ , formally,  $\forall x, y[p_s(x) \wedge p_{s'}(y) \wedge x = y \rightarrow t_{s'}(x) = t_{s'}(y)]$ .

Correspondence (or semantic equality) between two type\_sorts is determined by mutuality or sameness relation between their ownICs. The *mutuality* between two ownICs is decided when they are both exportable and importable, that is, the IC values provided by the ownICs for each individual are different but they are unique. In the case of *sameness*, both ownICs must provide the *same* IC value for the same individual, in addition to being exportable and importable.

**Example 6** Suppose that ‘hasFingerPrint’ and ‘hasIrisPattern’<sup>18</sup> are defined as the ICs of **Person**, and ‘hasOrgName’ and ‘TitleOfOrganization’ for the ICs of **Organization**. Then, ‘hasFingerPrint’ and ‘IrisPatternOf’ have *mutuality* relation, while ‘hasOrgName’ and ‘TitleOfOrganization’ have *sameness* relation.

We summarize the procedure of IC-based type\_sort mapping, below. A bottom-up searching approach is applied as shown in Figure 7, because ICs are inherited from top to bottom.

1. Choose a sort  $s'_i$  from  $S'_{type}$ .
  2. Test whether the own IC of sort  $s$ ,  $t_s$ , is exportable to  $s'_i$ .
  3. If yes, find  $s'_j$  such that  $s'_j \sqsupseteq s'_i$ , and test the top-most exportable sort  $s'_j$  for  $t_s$ , then test whether the ownIC of sort  $s'$ ,  $t_{s'}$ , is importable to  $s$  or not.
    - (a) If yes, there is mutuality, then test for sameness.
      - i. If yes, there is sort equality by sameness between  $s$  and  $s'_j$ .
      - ii. Otherwise, there is sort equality by mutuality between  $s$  and  $s'_j$ .
    - (b) Otherwise, try such  $s'_j \sqsubseteq s'_i$  on other branches for mutuality.
  4. Otherwise, go to (1) to select a next possible sort  $s'_i$ .
- *quasi-typeMatching* is a mapping function that finds the correspondence of quasi-type\_sort  $s \in S_{quasi-type}$  in  $S'_{quasi-type}$ . First, the scope of possible matches in  $S'_{quasi-type}$  is decided by finding type\_sort  $s'_1 \in S'_{type}$  which has a matched type\_sort  $s_1 \in S_{type}$  such that  $s \sqsubseteq s_1$ . After that, the correspondence of  $s$  is determined by a similar CVA in both  $P^D(s)$  and  $P^D(s')$ .

<sup>18</sup>We admit that a type\_sort can originate more than one IC (say multiple ownICs), e.g., **Person** has three ownICs: hasFingerPrint, IrisPatternOf, and hasPalmVeinPattern.

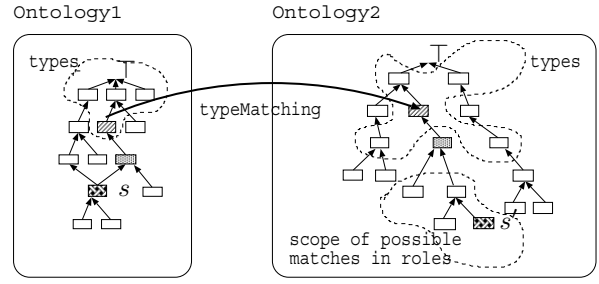


Figure 8: A general view of roleMatching function

- *roleMatching* is a mapping function that finds the correspondence of role\_sort  $s \in S_{role}$  in  $S'_{role}$ . There are three steps in the roleMatching function.
  - First, select sort  $s_1 \in S$  such that  $s \sqsubseteq s_1$ , to find a corresponding type\_sort  $s'_1 \in S'_{type}$  through typeMatching function.
  - Second, if typeMatching successfully returns a corresponding type\_sort  $s'_1$ , then determine the scope of possible matches by searching a corresponding quasi-type\_sort  $s'_2 \in S'_{quasi-type}$ .
  - Third, according to  $s'_2$ , examine the scope of possible matches among role\_sorts again, by selecting role\_sorts  $s'$  such that  $s' \sqsubseteq s'_2$ , and then determine the correspondence of  $s$  by a common EDR in both  $P^D(s)$  and  $P^D(s')$ .
- *phaseMatching* is a mapping function that finds the correspondence of phase\_sort  $s \in S_{phase}$  in  $S'_{phase}$ . The process of phaseMatching is almost similar to roleMatching, in analyzing a common CC except for EDR.

The corresponding sorts between Ontology1 and Ontology2 can be found through the above four matching functions.

## 6.2 Evaluation

We evaluate the MetaOntoModel-based matching method by calculating the mathematical complexity of matching function  $f$ . Suppose that the maximum number of sorts in  $O$  and  $O'$  are  $N$ . Let  $m$  and  $n$  be the number of individuals for sorts  $s$  and  $s'$ . The test for exportability and importability would take  $m$  and  $n$  comparisons respectively. For the convenience of estimation, if we regard a binary tree for taxonomies, then the average depth would be  $\log N$ , and the approximate number of leaves would be  $N/2$ .

The mathematical complexity of  $T_f$  is calculated based on the complexity of four sub-functions: typeMatching, quasi-typeMatching, roleMatching, and phaseMatching. Let the maximum number of type\_sorts be  $k$ ,  $1 \leq k \leq N$ . Then, the maximum number of quasi-type\_sorts, role\_sorts, and phase\_sorts, will be  $N - k$ . The worse case complexity of type\_sort matching is

$$T_{type} = ((k/2 + \log k) \times \mathcal{O}(m) + \mathcal{O}(n)) = \mathcal{O}(N \times m)$$

where  $(N/2 + \log N/2) \times \mathcal{O}(m)$  for exportability and  $\mathcal{O}(n)$  for importability when the total number of sorts is  $N$ . Then, the complexity of other sub-functions are as follows:

$$T_{quasi-type} = T_{type} + ((N - k) \times \mathcal{O}(1)) = \mathcal{O}(N \times m)$$

$$T_{\text{role}} = T_{\text{quasi-type}} + ((N-k) \times \mathcal{O}(m+n)) = \mathcal{O}(N \times m)$$

$$T_{\text{phase}} = T_{\text{quasi-type}} + ((N-k) \times \mathcal{O}(1)) = \mathcal{O}(N \times m)$$

Note that the complexity of CVA or CC, is assumed to be  $\mathcal{O}(1)$  because of direct attribute value or constraint matching. In the case of roleMatching, matching between two EDRs will cost  $\mathcal{O}(m+n)$ , due to checking whether each EDR provides the same range for each sort, or not. Finally,  $T_f$  in the worse case is

$$T_f = T_{\text{type}} + T_{\text{quasi-type}} + T_{\text{role}} + T_{\text{phase}} = \mathcal{O}(N \times m).$$

If  $T_f$  is applied for the complete ontology matching,  $T_c$ , of all available sorts between  $O$  and  $O'$ , then  $T_c$  would be reduced from  $\mathcal{O}(N^2 \times m)$  to  $\mathcal{O}(N \log N \times m)$ , because the matching functions need not be executed for the sub-sorts of every unmatched type\_sort.

## 7 Related Work

The development of methods and tools for ontology matching, alignment, and merging, has focused on a variety of works originating from diverse communities over a number of years. Kalfoglou and Schorlemmer (Kalfoglou & Schorlemmer 2005) conducted a comprehensive survey on a total of 35 mapping-related works. Noy and Musen (F. Noy & Musen 2002) also provided an evaluation-oriented analysis of some mapping tools, comparing them with their own experience in PROMPT for ontology merging.

Not only the major tasks but also the assumptions employed in each work are more or less different. Here, we provide an objective style review concerning how each tool deals with semantic heterogeneity and to what extent. For that purpose, we first present some existing methods and tools in brief, with respect to their background theory, mapping approach, and the level of expert-interaction.

- **PROMPT** (F. Noy & Musen 2003) is a semi-automatic interactive tool suit for performing ontology merging. For the phase of matching, AnchorPROMPT first detects linguistic similarity matches (called anchors) between ontology components, and then determines their semantic correspondences using related structural knowledge such as subsumption relations and properties (or slots). User approval is considered for merging between all possible kinds of correspondences. The limitation of PROMPT is that the two ontologies in the mapping (and merging) process should be different versions of the same ontology.
- **IF-Map** (Kalfoglou & Schorlemmer 2002) is a channel-theory-based automatic ontology mapping method. It uses logic isomorphism between ontologies with concept-to-concept mapping and relation-to-relation mapping. There are two assumptions in IF-Map, which are (a) a common reference ontology for all ontologies, and (b) considering an equal set of instances for the decision of concept matching, that is, if two concepts share the same set of instances then they are determined to be the same concept; otherwise not.
- **GLUE** (Doan et al. 2002) is an automatic ontology matching system that employs a multi-strategy machine learning technique with joint probability distribution. First, the similarity estimator determines the similarity between instances using multiple *base learners* and a *meta learner*. GLUE contains two kinds of base learner: *name learner* and *content learner*. Name

learner uses linguistic knowledge to calculate similarity between the names of two entities, by exploiting the frequency of words. Each content learner focuses on a certain type of information belonging to instances. Meta-learner is used to linearly combine the predictions of all base learners. Then, *relaxation labeler* determines the best mapping—which best satisfies the given domain constraints and heuristic knowledge—for each entity by analyzing the similarity results of all neighborhood entities.

- **QOM-Quick Ontology Mapping** (Ehrig et al. 2004) constitutes a straightforward name-based similarity computation between entities defined in two ontologies. QOM focus on less run-time complexity for the mapping efficiency of large-size, light-weight ontologies.

We learned that most matching tools rely much on name-based matching between ontological entities, rather than semantics (or content) defined for each entity. For very complex names, the tools need expert's verification or user's approval.

Our matching is focused on content-driven matching between two concepts. As we have discussed two concepts with the same name may have different semantics. Suppose that an ontology developed by a certain university, where only graduate courses are available, uses concept name **Student** for a set of graduate students. Another ontology developed by a different university, where only undergraduate courses are available, may use the same name for different (but overlapped) semantics. According to the name-based matching methods, the accuracy of matches or unmatches is rather risky.

Two concepts might have different names. However, they can have semantic correspondence because the meaning of concept names cannot completely express the semantics of concepts. Moreover, other kinds of heterogeneity may be involved between two concepts. In that case, content-based matching of all available properties and instances will become complex. In our approach, we could remove unnecessary complication by analyzing only the most closely corresponding properties between two concepts.

The similarity analysis by the content learners of GLUE is similar to our matching approach. The similarity between two nodes (classes) is determined by the similarity of their attributes and relations with their neighbour nodes. Then, the similarity between two attributes is calculated by the similarity between their corresponding instances. Suppose that  $N_c$ ,  $N_p$ , and  $N_i$  are the maximum number of nodes, properties (attributes & relations), and instances. Let us assume that the complexity of comparing two attribute values between two instances is  $\mathcal{O}(1)$ . Then, the complexity of calculating similarity between two instances will be  $\mathcal{O}(N_p)$ . And,  $\mathcal{O}(N_p^2 \times N_i)$  will be the complexity for the similarity between two nodes. Finally, the matching between two ontologies will take  $\mathcal{O}(\log N_c \times N_p^2 \times N_i)$ . In order to compare GLUE with our matching approach, let us substitute  $N$  for every parameter; the cost of GLUE will become  $\mathcal{O}(N^3 \log N)$ , while our matching approach costs  $\mathcal{O}(N^2 \log N)$  because our method does not require comparing all properties belonging to each class.

## 8 Conclusion

We conclude this paper with three main points: (a) the summary of our contributions, (b) the advantages and limitations of our semantic enrichment and matching method, and (c) our future work. First of all, our contributions are listed below.

- We provided MetaOntoModel in order to provide well-conceptualized and semantically-enriched ontologies for matching between heterogeneous ontologies, with less expert-interaction.
- We developed an open source meta-class ontology file (named 'sort.owl') which includes the frames of four sort meta-classes, and five PAL constraints to check subsumption consistency.
- We demonstrated the MetaOntoModel-based enrichment process using Protégé OWL API.
- We presented a content-based matching method that can reduce the cost of matching between heterogeneous ontologies.
- We evaluated our matching technique in terms of mathematical time complexity, and provided a comparison with other mapping tools.

The advantages of the MetaOntoModel-based matching method over other mapping methods are (a) the time cost can be reduced by direct matching between the same meta-knowledge groups; and (b) semantic correspondence between two sorts can be decided by matching between the most closely corresponding properties such as ownICs, CVA, EDR, and CC, instead of comparing all the properties belonging to the sorts. Our approach brings together techniques in philosophy, conceptualization, formal ontologies, mathematical logic, and knowledge representation.

The limitation of this work is that ontological classes must be sortal, and they should be enriched according to MetaOntoModel. We admit that users need sufficient background knowledge for the classification of sorts, particularly based on rigidity and IC.

In future work, we will consider an intelligent assistant to users in deciding whether a sort is a rigid designator for its individuals, or not. Also, we will present MetaOntoModel-based ontology merging together with an alignment system of mapping results.

## References

- Batini, C. & Lenzerine, M. & Navathe, S. B. (1986), Comparison of Methodologies for Database Schema Integration, *ACM Computing Surveys*, vol. 18(4), pp. 323–364.
- Beierle, C. (1992), An order-sorted logic for knowledge representation systems, *Artificial Intelligence*, vol. 55, pp. 149–191.
- Belardinelli, F. (2006), Quantified Modal Logic and Ontology of Physical Objects, *PhD Thesis*, Scuola Normale Superiore (SNS) in Pisa.
- Bowers, S. & Delcambre, L. (2000), Representing and transforming model-based information, the First Workshop on the Semantic Web at the Fourth European Conference on Digital Libraries.
- Ceri, S. & Widom, J. (1993), Managing Semantic Heterogeneity with Production Rules and Persistent Queues, *Proceedings of the 19th VLDB Conference*, pp. 108–119.
- Chalupsky, H. (2000), A translation system for symbolic logic, *Principles of Knowledge Representation and Reasoning (KR2000)*, pp. 471–482.
- Corsi, G. (2001), A unified completeness theorem for quantified modal logics, *Journal of Symbolic Logic*, vol. 67, pp. 1483–1510.
- Cresswell, M. J. (2001), *A Blackwell Guide to Philosophical Logic*.
- Doan, A., Madhavan, J., Domingos, P. & Halevy, A. (2002), Learning To Map between Ontologies on the Semantic Web, *WWW2002*.
- Ehrig, M. & Stabb, S. (2004), QOM-Quick Ontology Mapping, University of Karlsruhe.
- F. Noy, N. & Musen, M.A. (1999), Evaluating Ontology-Mapping Tools: Requirements and Experience, *Proceedings of the Workshop on Evaluation of Ontology Tools at EKAW'02*.
- F. Noy, N. & Musen, M.A. (2003), The PROMPT suite: interactive tools for ontology merging and mapping, *International Journal of Human-Computer Studies*, vol. 59(6), pp. 983–1024.
- Guarino, N. & Welty, C. (2001), Supporting Ontological Analysis of Taxonomic Relationships, *Data and Knowledge Engineering*, vol. 39, pp. 51–74.
- Guizzardi, G., Wagner, G. & Sinderen, M. (2004), A Formal Theory of Conceptual Modeling Universals, *proceedings of the workshop on philosophy and informatics (WSPI)*, Cologne, Germany.
- Hughes, G. E. & Cresswell, M. J. (2003), *A New Introduction to Modal Logic*, Third Edition.
- Kalfoglou, Y. & Schorlemmer, M. (2002), IF-Map: An Ontology Mapping Method based on Information-Flow Theory, *1st International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'02)*.
- Kalfoglou, Y. & Schorlemmer, M. (2005), Ontology Mapping: The State of the Art, *the journal of Semantic Interoperability and Integration*.
- Kaneiwa, K. (2001), Order-Sorted Logic Programming with Predicate Hierarchy, *Artificial Intelligence*, vol. 158, pp. 155–188.
- Kripke, S. (1971), Identity and Necessity, In M. Munitz (ed.), *Identity and Individuation*, New York, New York University Press, pp. 135–164.
- March, S. T. (1990), Special Issue on Heterogeneous Databases, *ACM Computing Surveys*, vol. 22(3).
- Miller, B. (1987), "Exists" and Existence, *Review of Metaphysics*, vol. 40, pp. 237–270.
- Mitra, P. & Wiederhold, G. (2002), Resolving Terminology Heterogeneity in Ontologies, *ECAI'02*.
- Quine, W. V. O. (1969), Ontological Relativity and Other Essays, *Columbia University Press*.
- Su, X (2004), Semantic Enrichment for Ontology Matching, *PhD Thesis*, Norwegian University of Science and Technology.
- Tun, N. N. & Tojo, S. (2005), IC-based Ontology Expansion in Devouring Accessibility, *Australian Ontology Workshop (AOW 2005)*, Australian Computer Society, vol. 58, pp. 99–106.
- Visser, P. R. S. & Jones, D. M. R. & Bench-Capon, T. J. M. & Shave, M. J. R. (1997), Assessing Heterogeneity by Classifying Ontology Mismatches, *Proceedings of AAAI'97 Spring Symposium on Ontological Engineering*.
- Welty, N. & Andersen, W. (2005), Towards OntoClean 2.0: A Framework for Rigidity, *Journal of Applied Ontology*.
- Wong, K. (2003), Rigid designation, Existence and Semantics for Quantified Modal Logic, Department of Philosophy, The Chinese University of Hong Kong.

## Author Index

Brusa, Graciela, 7

Caliusco, María Laura, 7

Chiotti, Omar, 7

Dras, Mark, 41

Hooijmaijers, Dennis, 17

James, Geoff, 41

Kühnberger, Kai-Uwe, 51

Lambert, Dale, 25

Lefort, Laurent, 31

Meyer, Thomas, iii

Nayak, Abhaya, 41

Nowak, Chris, 25

Orgun, Bhavna, 41

Orgun, Mehmet A., iii

Ovchinnikova, Ekaterina, 51

Patrick, Jon, 61

Ratcliffe, David, 31

Ryan, Amanda, 69

Schwitter, Rolf, 75

Stumptner, Markus, 17, 85

Taylor, Kerry, 3, 31

Thiagarajan, Rajesh, 85

Tilbrook, Marc, 75

Tun, Nwe Ni, 91

## Recent Volumes in the CRPIT Series

ISSN 1445-1336

Listed below are some of the latest volumes published in the ACS Series *Conferences in Research and Practice in Information Technology*. The full text of most papers (in either PDF or Postscript format) is available at the series website <http://crpit.com>.

**Volume 53 - Conceptual Modelling 2006**

Edited by Markus Stumptner, *University of South Australia*, Sven Hartmann, *Massey University, New Zealand* and Yasushi Kiyoki, *Keio University, Japan*. January, 2006. 1-920-68235-X.

Contains the proceedings of the Third Asia-Pacific Conference on Conceptual Modelling (APCCM2006), Hobart, Tasmania, Australia, January 2006.

**Volume 54 - ACSW Frontiers 2006**

Edited by Rajkumar Buyya, *University of Melbourne*, Tianchi Ma, *University of Melbourne*, Rei Safavi-Naini, *University of Wollongong*, Chris Steketee, *University of South Australia* and Willy Susilo, *University of Wollongong*. January, 2006. 1-920-68236-8.

Contains the proceedings of the Fourth Australasian Symposium on Grid Computing and e-Research (AusGrid 2006) and the Fourth Australasian Information Security Workshop (Network Security) (AISW 2006), Hobart, Tasmania, Australia, January 2006.

**Volume 55 - Safety Critical Systems and Software 2005**

Edited by Tony Cant, *University of Queensland*. April, 2006. 1-920-68237-6.

Contains the proceedings of the 10th Australian Workshop on Safety Related Programmable Systems, August 2005, Sydney, Australia.

**Volume 56 - Vision in Human-Computer Interaction**

Edited by Roland Goecke, Antonio Robles-Kelly, and Terry Caelli, *NICTA*. November, 2006. 1-920-68238-4.

Contains the proceedings of the HCSNet Workshop on the Use of Vision in Human-Computer Interaction (VisHCI 2006).

**Volume 57 - Multimodal User Interaction 2005**

Edited by Fang Chen and Julien Epps *National ICT Australia*. April, 2006. 1-920-68239-2.

Contains the proceedings of the NICTA-HCSNet Multimodal User Interaction Workshop 2005, Sydney, Australia, 13-14 September 2005.

**Volume 58 - Advances in Ontologies 2005**

Edited by Thomas Meyer, *National ICT Australia, Sydney* and Mehmet Orgun *Macquarie University*. December, 2005. 1-920-68240-6.

Contains the proceedings of the Australasian Ontology Workshop (AOW 2005), Sydney, Australia, 6 December 2005.

**Volume 60 - Information Visualisation 2006**

Edited by Kazuo Misue, Kozo Sugiyama and Jiro Tanaka. February, 2006. 1-920-68241-4.

Contains the proceedings of the Asia-Pacific Symposium on Information Visualization (APVIS 2006), Tokyo, Japan, February 2006.

**Volume 61 - Data Mining 2006**

Edited by Simeon Simoff, *University of Technology, Sydney* and Graham Williams *Australian Tazation Office and University of Canberra*. December, 2006. 1-920-68242-2.

Contains the proceedings of the Australasian Data Mining Conference (AusDM 2006), December 2006.

**Volume 62 - Computer Science 2007**

Edited by Gillian Dobbie, *University of Auckland, New Zealand*. January, 2007. 1-920-68243-0.

Contains the proceedings of the Thirtieth Australasian Computer Science Conference (ACSC2007), Ballarat, Victoria, Australia, January 2007.

**Volume 63 - Database Technologies 2007**

Edited by James Bailey, *University of Melbourne* and Alan Fekete, *University of Sydney*. January, 2007. 1-920-68244-9.

Contains the proceedings of the Eighteenth Australasian Database Conference (ADC2007), Ballarat, Victoria, Australia, January 2007.

**Volume 64 - User Interfaces 2007**

Edited by Wayne Piekarski, *University of South Australia*. January, 2007. 1-920-68245-7.

Contains the proceedings of the Eighth Australasian User Interface Conference (AUIC2007), Ballarat, Victoria, Australia, January 2007.

**Volume 65 - Theory of Computing 2007**

Edited by Joachim Gudmundsson, *NICTA, Australia* and Barry Jay *UTS, Australia*. January, 2007. 1-920-68246-5.

Contains the proceedings of the Thirteenth Computing: The Australasian Theory Symposium (CATS2007), Ballarat, Victoria, Australia, January 2007.

**Volume 66 - Computing Education 2007**

Edited by Samuel Mann, *Otago Polytechnic* and Simon *Newcastle University*. January, 2007. 1-920-68247-3.

Contains the proceedings of the Ninth Australasian Computing Education Conference (ACE2007), Ballarat, Victoria, Australia, January 2007.

**Volume 67 - Conceptual Modelling 2007**

Edited by John F. Roddick, *Flinders University* and Annika Hinze, *University of Waikato, New Zealand*. January, 2007. 1-920-68248-1.

Contains the proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM2007), Ballarat, Victoria, Australia, January 2007.

**Volume 68 - ACSW Frontiers 2007**

Edited by Ljiljana Brankovic, *University of Newcastle*, Paul Coddington, *University of Adelaide*, John F. Roddick, *Flinders University*, Chris Steketee, *University of South Australia*, Jim Warren, *University of Auckland*, and Andrew Wendelborn, *University of Adelaide*. January, 2006. 1-920-68249-X.

Contains the proceedings of the ACSW Workshops - The Australasian Information Security Workshop: Privacy Enhancing Systems (AISW), the Australasian Symposium on Grid Computing and Research (AUSGRID), and the Australasian Workshop on Health Knowledge Management and Discovery (HKMD), Ballarat, Victoria, Australia, January 2007.

**Volume 72 - Advances in Ontologies 2006**

Edited by Mehmet Orgun *Macquarie University* and Thomas Meyer, *National ICT Australia, Sydney*. December, 2006. 1-920-68253-8.

Contains the proceedings of the Australasian Ontology Workshop (AOW 2006), Hobart, Australia, December 2006.

**Volume 73 - Intelligent Systems for Bioinformatics 2006**

Edited by Mikael Boden and Timothy Bailey *University of Queensland*. December, 2006. 1-920-68254-6.

Contains the proceedings of the AI 2006 Workshop on Intelligent Systems for Bioinformatics (WISB-2006), Hobart, Australia, December 2006.