Conferences in Research and Practice in Information Technology

Volume 90

Advances in Ontologies

AUSTRALIAN
COMPUTER
SOCIETY

Advances in Ontologies

Proceedings of the
Knowledge Representation Ontology Workshop,
Sydney, Australia, 17 September 2008

Thomas Meyer and Mehmet A. Orgun, Eds.

Volume 90 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.

Advances in Ontologies. Proceedings of the Knowledge Representation Ontology Workshop, Sydney, Australia, 17 September 2008

Conferences in Research and Practice in Information Technology, Volume 90.

Editors:
Thomas Meyer
Meraka Institute
PO Box 395
Pretoria, 0001
South Africa
E-mail: tommie.meyer@meraka.org.za

Mehmet A. Orgun
Intelligent Systems Group (ISG), Department of Computing
Macquarie University
Sydney, NSW 2109
Australia
E-mail: Mehmet.Orgun@mq.edu.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 90
ISSN 1445-1336
ISBN 978-1-920682-71-2

Printed August 2008 by Griffith University Uni Print, Nathan Campus, Kessels Rd. Nathan, 4111, QLD, 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 Knowledge Representation Ontology Workshop, Sydney, Australia, 17 September 2008

Preface vii

Programme Committee viii

Utilising Ontological Structure for Reasoning with Preferences 1
Gil Chamiel and Maurice Pagnucco

An Argumentative Approach to Reasoning with Inconsistent Ontologies 11
Sergio Alejandro Gómez, Carlos Iván Chesñevar, and Guillermo Ricardo Simari

Towards Distributed Tableaux Reasoning Procedure for DDL with Increased Subsumption Propagation between Remote Ontologies 21
Martin Homola and Luciano Serafini

Enhancing Subjective Ontologies with Social Tagging Systems 31
Dennis Hooijmaijers and Markus Stumptner

Heterogeneously Structured Ontologies - Integration, Connection, and Refinement 41
Oliver Kutz, Dominik Lücke, and Till Mossakowski

Supporting Coral Reef Ecosystems Research through Modelling Re-usable Ontologies 51
Trina S. Myers, Ian Atkinson, and Ron Johnstone

Ontology Evolution for Customer Services 61
Tho T. Quan and Thai D. Nguyen

Creating and Querying Linguistically Motivated Ontologies 71
Rolf Schwitter

Author Index 81

Preface

The Knowledge Representation Ontology Workshop (KROW 2008) took place in Sydney, NSW, Australia on 17 September 2008, in conjunction with KR 2008: the Eleventh International Conference on Principles of Knowledge Representation and Reasoning, Sydney, Australia, September 16-19, 2008. KROW 2008 is a continuation of the successful Australasian Ontology Workshop (AOW) series which were held in conjunction with the Australian Joint Conferences on Artificial Intelligence in the past few years. AOW will resume again next year.

The primary aim of KROW 2008 is to bring together active researchers in the broad area of ontologies and to promote interaction with the knowledge representation and reasoning community. 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.

The programme of KROW 2008 featured an invited speech and eight contributed presentations. The invited speaker, Enrico Franconi of the Free University of Bozen-Bolzano, summarised the issues that arise when trying to "integrate" ontologies with databases. He described the differences between standard (relational) database access technology and the technology for accessing (integrated) information sources mediated by global constraints or ontologies. He presented various current solutions to the problem, emphasising the advantages and the limitations of each of them, and their feasibility in real world data intensive applications.

The contributed papers deal with many aspects of ontology research, including the theoretical foundations, description logics, integration and evolution of ontologies, ontologies for the Semantic Web, and applications of real-world ontologies in diverse areas such as customer services, reef eco-systems, social networks, as well as those describing new challenges arising out of applications.

A program committee of international standing reviewed all contributed papers (full papers were reviewed). Each paper was reviewed by at least three program committee members, and additional reviews were sought to identify the most promising papers. As a result, eight out of sixteen submitted papers were included for publication in this proceedings, involving authors from twelve different countries.

Many individuals contributed to this workshop. We would like to thank the invited speaker, Enrico Franconi, the authors, the members of the Program Committee of KROW 2008, and the additional reviewers for their contributions to the quality of the workshop and this collection. Thanks are also due to Maurice Pagnucco, KR 2008 Local Chair, for his help with the smooth organisation of this workshop event, and to Vladimir Estivill-Castro, John Roddick and Simeon Simoff, the editors of the CRPIT series, for facilitating the publication of the KROW 2008 proceedings.

We would like to 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.

Thomas Meyer, Meraka Institute
Mehmet A. Orgun, Intelligent Systems Group (ISG), Department of Computing
Organisers of KROW 2008
September, 2008

Programme Committee

Programme Chairs

Thomas Meyer (Meraka Institute, South Africa)
Mehmet A. Orgun (Macquarie University, Australia)

Programme Committee

Franz Baader (TU Dresden, Germany)
Mike Bain (UNSW, Australia)
Richard Booth (Mahasarakham University, Thailand)
Arina Britz (Meraka Institute, South Africa)
Longbing Cao (UTS, Australia)
Werner Ceusters (SUNY Buffalo, USA)
Anne Cregan (UNSW, Australia)
Attila Elçi (Eastern Mediterranean University, Turkey)
Aurona Gerber (Meraka Institute, South Africa)
Manolis Gergatsoulis (Ionian University, Greece)
Dennis Hooijmaijers (University of South Australia, Australia)
Bo Hu (University of Southampton, UK)
Renato Iannella (NICTA, Australia)
Ken Kaneiwa (NICT, Japan)
Marijke Keet (Free University of Bolzano, Italy)
Kevin Lee (NICTA and UNSW, Australia)
Laurent Lefort (CSIRO, Australia)
Costas Mantratzis (University of Westminster, UK)
Lars Mönch (University of Hagen, Germany)
Deshendran Moodley (University of KwaZulu Natal, South Africa)
Maurice Pagnucco (UNSW, Australia)
Anet Potgieter (University of Cape Town, South Africa)
Debbie Richards (Macquarie University, Australia)
Rolf Schwitter (Macquarie University, Australia)
Murat Şensoy (Boğaziçi University, Turkey)
Barry Smith (SUNY Buffalo, USA)
Boontawee Suntisrivaraporn (TU Dresden, Germany)
Kerry Taylor (CSIRO, Australia)
Sergio Tessaris (Free University of Bolzano, Italy)
Kewen Wang (Griffith University, Australia)
Mary-Anne Williams (UTS, Australia)

Additional Reviewers

Shan Chen
Duygu Çelik
Guilin Qi
Behnam Rahnama
Laurianne Sitbon
Nwe Ni Tun
Toshio Ueda
Antoine Zimmermann

Utilising Ontological Structure for Reasoning with Preferences

Gil Chamiel **Maurice Pagnucco**

School of Computer Science and Engineering
The University of New South Wales
NSW, Sydney 2052, Australia and NICTA, Sydney, Australia.
Email: {gilc|morri}@cse.unsw.edu.au

Abstract

The ability to model preferences and exploit preferential information to assist users in searching for items has become an important issue in knowledge representation. On the one hand, accurately eliciting preferences from the user in the form of a query can result in a coarse recommendation mechanism with numerous results returned. The problem lies in the user's knowledge concerning the items among which they are searching. Unless the user is a domain expert, their preferences are likely to be expressed in a vague manner and so vague results (in the form of numerous alternatives) are returned.

In this paper we remedy this problem by exploiting ontological information regarding the domain at hand. This ontological information can be provided by a domain expert and need not concern the user. However, we show that it can prove useful in focusing query results and providing more meaningful and useful recommendations.

Keywords: Ontologies, Reasoning with Preferences

1 Introduction

We are faced with choices every day: which type of restaurant to go to; which radio station to listen to or TV channel to watch? In order to answer these questions we exercise our preferences. Preferences make effective reasoning possible since they encapsulate our everyday decisions. However, explicit preferences alone are only part of the story. In some situations our own understanding of the domain is poor and our preferences tend to reflect this. As a result, we are unable to effectively discriminate among the choices at hand. This paper addresses this problem by supplementing user preferences with ontological information so as to provide an effective user preference mechanism.

Research on modeling abstract notions of preference has provided a rich literature in logic and decision theory (Doyle & Thomason 1999, Fishburn 1999, Bradley et al. 2000) with applications such as modeling social choice in economics. With the growth of the World Wide Web as a major platform for purchase and consumption, the need for personalised product recommendation systems has become evident, and their development has become an important issue in Knowledge Representation and Reasoning and Artificial Intelligence. One method that has gained popularity in the last few years is *social-based product recommendation*, e.g. collaborative filtering (Sarwar et al. 2001), where the selection history of other customers is used to recommend products. Such techniques tend to ignore deeper information of the domain in favour of following the herd mentality.

The aim of this paper is to exploit information that is available in the structure of an OWL ontology to supplement user preferences in order to provide an effective choice mechanism. We consider a number of methods based on different definitions of concept similarity *as measured in a RDF graph representation of an OWL ontology. Furthermore, we provide an implementation of our schemes in the SPARQL query language that extends previous work by (Siberski et al. 2006) building on the preference mechanisms added to SQL by (Kießling 2002, Kießling & Köstler 2002).*

Previous work in this area is limited. An ontology based similarity system has been presented by (Middleton et al. 2004) but provides for only basic features. (Schickel-Zuber & Faltings 2006) is much closer in spirit to this paper, introducing the notion of *ontology filtering*. However, they propose a score prorogation system within an hierarchical graph, where we focus on the structural properties of the ontology. (Schickel-Zuber & Faltings 2007) supplement this approach by trying to learn the ontologies themselves. (Kiefer et al. 2007) has applied similarity to semantic data mapping, ontology mapping and semantic web service matchmaking using techniques from linguistic analysis.

The rest of this paper is arranged in the following way. In the next section we cover the necessary background material with a brief introduction to OWL and SPARQL. We then briefly look at the work by (Kießling 2002) and (Kießling & Köstler 2002) extending SQL and it's partial implementation in SPARQL by (Siberski et al. 2006). This is followed by our proposal based on concept similarity, exploiting the structure in an OWL ontology. We end with a discussion and conclusions.

2 Background

Reasoning with preferences has been studied and developed in many disciplines from social choice, decision and game theory, logic and philosophy through to artificial intelligence and machine learning. In our work, we adopt these basic principles together with their application to preference querying in database systems. In order to provide more accurate and sensible preference querying, we have chosen to use description logic based ontologies as our data model since they provide a richer description of the underlying domain. In this section, we briefly discuss these concepts.

2.1 Preference Querying in Database Systems

In the context of database systems, (Kießling 2002) presents a framework for dealing with preferences as soft constraints, named *Preference-SQL*. In this work, preferences are seen as strict partial orders over database tuples i.e., as transitive and irreflexive preference relations. It proposes an extension to the SQL query language (Kießling & Köstler 2002) which permits filtering the result set using soft constraints as opposed to classical SQL filtering which adopts hard constraints returning only those results that match the query *exactly*. A partial syntax of the extended query language is given below:

```
SELECT      <projection-list>
FROM        <table-reference>
WHERE       <hard-conditions>
PREFERRING  <soft-conditions>
ORDER BY    <attribute-list>
```

Using this syntax, the user can express their preferences as soft constraints and will receive tuples which *best match* those constraints. This approach is referred to as the BMO (*Best Match Only*) query model in which a tuple will find its way into the final result set if there does not exist any other tuple which *dominates* it, i.e. better satisfies the preference constraints. Preference constraints in this framework can be expressed through standard SQL operators in terms of likes/dislikes (e.g. `=`, `<>`, `IN`) and numeric constraints (e.g. `<`, `>=`, `BETWEEN`). This work also introduces a set of special operators which allow the user to express their preferences in terms of numerical approximations (through the operator `AROUND` which will favour values close to a given numerical target value) and in terms of minimization/maximization of numerical values (through the operators `LOWEST/HIGHEST` which will accept a lowest/highest value respectively over other values). In order to allow complex preference construction, two binary preference assembly operators are introduced: The *Pareto accumulation* (`AND`) treats both constituent preference constructs as equal and is defined as:

$$x \prec_{P_1 \otimes P_2} y \Leftrightarrow$$
$$(x \prec_{P_1} y \wedge (x \prec_{P_2} y \vee x \equiv y)) \vee$$
$$(x \prec_{P_2} y \wedge (x \prec_{P_1} y \vee x \equiv y))$$

Where x, y are database tuples and P_1, P_2 are preference constructs. Intuitively, this definition says that y is strictly preferred to x whenever it is strictly preferred by at least one of the two constituent preferences and equally or more preferred by the other.
The *Prioritize accumulation* (`CASCADE`) is used to treat two preference constructs in order of importance and is defined:

$$x \prec_{P_1 \&\& P_2} y \Leftrightarrow$$
$$x \prec_{P_1} y \vee (x \prec_{P_2} y \wedge (x \prec_{P_1} y \vee x \equiv y))$$

This definition says that x is strictly preferred to y whenever it is strictly preferred by the first preference in the cascade and otherwise it is strictly preferred by the second preference in the cascade and equally or more preferred by the first. Therefore, the first constituent preference in the cascade is given higher consideration.

2.2 Querying Ontological Information

Ontologies provide a standardised way of classifying terms related to a domain. They are central to the knowledge-based paradigm.

2.2.1 Description Logic Based Ontologies

In recent years, with the emergence of the Semantic Web, the importance of knowledge representation and reasoning techniques over ontological information has gained added significance. Ontologies (Antoniou & van Harmelen 2004) are a knowledge representation technique based on description logic (*DL*) (Baader et al. 2003) principles consisting of terminological entities, i.e. *Concepts* and *Properties* also referred to as *TBox* and *instances* (individuals) also referred to as *ABox*. A very important feature of ontologies is the inherent ability to define terminological objects in a hierarchical manner, that is, concepts and subconcepts, properties and sub-properties. We make no assumption about the nature of the ontology. For simplicity we concentrate on tree-like ontologies in this paper. However, the results can be straightforwardly extended to DAGs.

2.2.2 RDF and OWL

A very common methodology for creating ontological information is through the XML-based markup language OWL (*Web Ontology Language*) which allows the construction of the different entities of an ontology based on RDF graphs. In RDF, entities are represented using a triple pattern logic. Each element in the RDF graph has to be a literal or an RDF resource. For example, the following tag constructs an OWL class (concept) named MusicAlbum: `<owl:Class rdf:ID="MusicAlbum">`. Here, the subject is a class definition, the predicate is the class id and the object is the string "MusicAlbum". Properties and Individuals can be created in the same manner. For example, the following tag constructs a new individual music album `<MusicAlbum rdf:ID="Album_1">`. Suppose we have a property definition of a album's release year, the following tag will assign the number 2005 as `Album_1`'s release year: `<Album_1 :releaseYear=2005>`.

2.2.3 SPARQL

The SPARQL query language over RDF graphs (Prud'hommeaux & Seaborne 2006) is a W3C Recommendation and considered to be a vital tool for dealing with ontological information in the semantic web. SPARQL allows queries over RDF Graphs using *Triple Pattern Matching* by introducing variables and binding the appropriate RDF resources to the variables. A partial syntax for SPARQL is given below:

```
SELECT    <projection-var-list>
FROM      <ontology-reference>
WHERE     <var-bindings>
FILTER    <hard-conditions>
ORDER BY  <var-list>
```

As opposed to SQL, the projected entities are variables where the equivalent entity to a database tuple is a set of variables also referred to as a *solution binding* (or a *result binding*). The `WHERE` clause in the query is typically used to bind variables to RDF resources while hard constraints are given through the `FILTER` clause (although it is possible to perform certain filter operations through the `WHERE` clause itself). The `ORDER BY` clause acts as a solution modifier, i.e. a solution list transformation function, which sorts the result bindings according to the variables in a given list. Other solution modifiers offered by SPARQL are `LIMIT` to limit the number of results returned and `OFFSET` to instruct SPARQL to start the result set after a given number of bindings are found.

3 Querying Ontological Information with Preferences

A natural progression is to extend SPARQL with preferential queries.

3.1 P-SPARQL

In a similar fashion to the way that Kießling extended SQL to enable database querying with preferences, (Siberski et al. 2006) presents an extension to SPARQL to query ontological information with preferences. The fundamental idea here is similar; a new query element is introduced to allow the construction of preferences as soft constraints. The extended syntax of SPARQL (in the rest of this paper, we refer to this extension as P-SPARQL) is given below:

```
SELECT      <projection-var-list>
FROM        <ontology-reference>
WHERE       <var-bindings>
FILTER      <hard-conditions>
ORDER BY    <var-list>
PREFERRING  <soft-conditions>
```

In the preferring section, every filter operator supported by SPARQL can be used as well as two scoring operators, `HIGHEST/LOWEST` with similar semantics as Preference-SQL. Also, similarly to Preference-SQL, two complex preference assembly methods are implemented, i.e. the Pareto operator for treating two preference operators as equally important and the Cascade operator to prioritize one preference operator over the other. Finally, in this work the BMO (Best Match Only) query model was adopted where a solution binding is a best match if there is no other solution binding dominating it (i.e., strictly preferred). Each solution binding competes against every other solution binding where a solution binding will find its way into the final result set if it is a best match under this definition.

4 Exploiting Hierarchical Structure for Reasoning with Preferences

One of the most distinctive properties in representing knowledge using ontologies is the inherent ability to define the terminologies in the ontology in an hierarchical manner. In analogue to terminologies in database systems, i.e. the database schema, in this work we consider the terminological component of an ontology (*TBox*) to be the part which stores information created by experts on the domain at hand. This assumption will then enable us to supply (possibly recommend) to the user information about individuals (*ABox*) according to their preferences without having to assume a very high level of domain knowledge on behalf of the user, and exploit the level of knowledge the user does have in order to perform more accurate preference querying. This is important because in many product recommendation settings, the user possesses little if any domain knowledge.

4.1 Motivating Example

4.1.1 The Music Record Shop Ontology

Consider an ontology which describes and stores information about music albums. This may include musical genre, performing artists, country where the album was produced, price and so on. Let's also consider a concept hierarchy composed by musical experts given in Figure 1. Note that this information is taken from domain experts while we do not assume the user has complete knowledge of the domain. Still this information can be readily exploited when reasoning with preferences (e.g. for a consumer who would like to search for items in a more sophisticated manner). In (Chamiel & Pagnucco 2008*a*) we identify a class of hierarchical systems referred to as *data inheritance systems*. This means that each concept in the hierarchy (not only a leaf) can be referred to not only as an abstract concept (probably through a set of properties this concept contains) but also as a data concept—a concept which can be associated with asserted instances (i.e., real objects). This is the standard behaviour of ontologies in the semantic web. This musical genre hierarchical system is of this kind: an album can be identified as *Progressive Rock* which is a leaf concept but at the same time an album can be identified with the concept *Rock* even though it serves as an abstract concept.

Example Query Consider the following P-SPARQL query:

```
Prefix music:
  <http://example.com/music.owl#>
SELECT ?id ?genre ?country
FROM <http://example.com/music.owl>
WHERE {
  ?id music:hasGenre ?genre.
  ?id music:producedIn ?country.}
PREFERRING
  ?country = music:England
CASCADE
  ?genre = music:AlternativeRock
```

In this query, we prefer albums originating from England as our most important preference attribute before we prefer the alternative rock genre. In case no such album exists to perfectly answer our preference (i.e. no English Alternative Rock exists), we argue that it will be sensible to return an English album with genre similar to *Alternative Rock* over returning any arbitrary album which happens to be English. Running this query through P-SPARQL will have exactly this latter behaviour where any musical genre will be considered equal without examining its relationship to the target concept (English album). This will of course depend on what we mean by "similar to" which we explore next.

4.2 Querying over *IS-A* Relations

In Description Logic (Baader et al. 2003) concept inheritance can be viewed as an *IS-A* relation $C_1 \sqsubseteq C_0$ where a sub-concept C_1 is also of type C_0 by inheriting its properties and functionality. Even though OWL is based on description logic concepts, this behaviour is not inherited in SPARQL. Filtering using the equal operator will return only those objects which have that particular target concept. Furthermore, the filtering option `?type rdfs:subClassOf C` (for querying individuals which have a sub-concept of some target concept C), will result only in those that are a direct sub-class of C omitting individuals which have the type C itself (and thus not satisfying the axiom $C \sqsubseteq C$) as well as individuals which have a type which is not directly inherited from C (and thus not satisfying the transitivity property of the *IS-A* relation). In (Chamiel & Pagnucco 2008*a*) we introduce a new operator which allows the user to express preferences in terms of the *IS-A* relation.

Example 1. *In order to modify the previous example to prefer English albums and then albums of genre which* IS-A *Alternative Rock (i.e., Alternative Rock and all its descendants), we modify the* `PREFERRING` *section of our query to:*

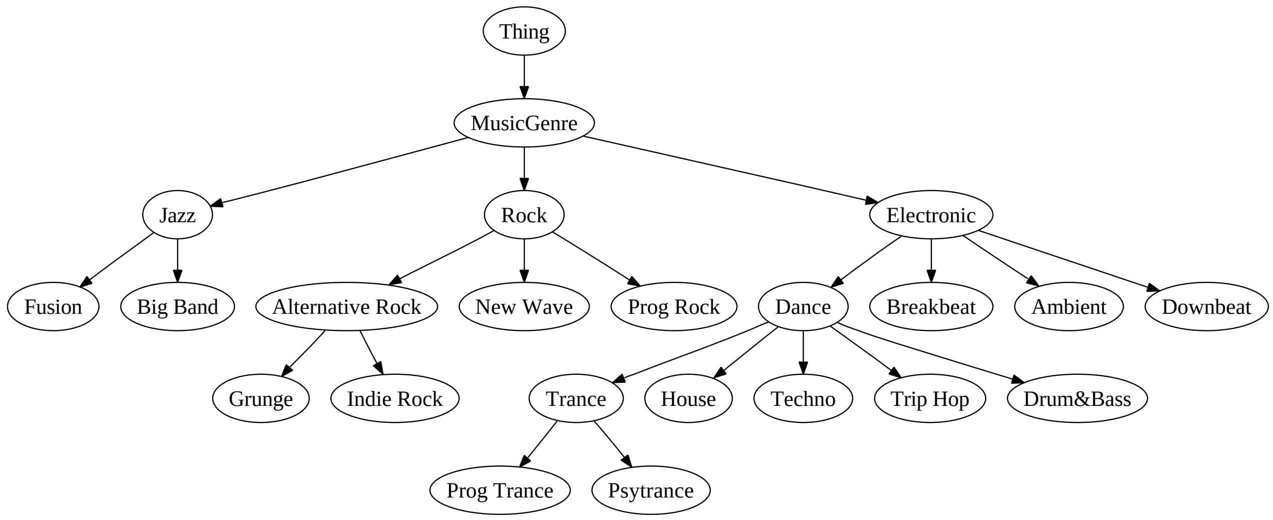


Figure 1: Ontological Concept Hierarchy of Music Genres

```
PREFERRING
  ?country = music:England
CASCADE
  ?genre ISA music:AlternativeRock
```

The results will thus be an albums made in England with genre in {AlternativeRock, Grunge, IndieRock}.

However, there are still a few problems with this method: what if there is no result under the target concept (or any of its sub-concepts)? Should we consider all other concepts outside the target concept as equal? Furthermore, is there any relation between different levels within a sub-hierarchy? Do we want to distinguish between general and specific concepts? We discuss these issues in the next section.

4.3 Similarity-based Querying

In order to exploit the hierarchical structure of an ontology, we present here a series of methods for computing categorical similarity between concepts in a $TBox$. We use these similarity methods for performing preference querying over ontological information. We introduce a new Boolean operator $Sim(C_1, C_2)$ (is similar to):

$$b_1 \prec_{P(C_0)} b_2 \Leftrightarrow Sim(C(b_1), C_0) < Sim(C(b_2), C_0) \quad (1)$$

Where $C_0 \in Concepts$ is the target concept, $b_1, b_2 \in ResultBindings$ and $C(b_i)$ is the value bound to the relevant variable in the result binding b_i w.r.t C. For example, suppose we would like now to query music albums while preferring English albums (as the first priority) and then albums of genre *similar to* Alternative Rock music. We modify the `PREFERRING` section of our query to:

```
PREFERRING
  ?country = music:England
CASCADE
  ?genre ∼= music:AlternativeRock
```

Where $\sim=$ is the syntactic version of the similarity operator $Sim(C_1, C_2)$. Note that by introducing a new Boolean operator we do not change the notion of domination querying. We still have the ability to compare two result bindings to obtain the preference domination relationship between them. There are many ways to compute similarity between concepts in an ontology, each reflecting a different rationale. In the rest of this section we discuss various methods and their rationales for computing this kind of similarity measurement.

4.3.1 Similarity via Direct Graph Distance

A very simple method for measuring similarity in an ontology graph is by looking at the direct distance between a candidate concept and the target concept:

$$Sim(C_1, C_0) = \frac{1}{Dist(C_1, C_0)} \quad (2)$$

Where distance is defined as the distance of each concept to the *most recent common ancestor* of the two concepts:

$$Dist(C_1, C_0) = N_1 + N_0 + 1 \quad (3)$$

where $N_i = len(C_i, MRCA(C_1, C_0))$ and $MRCA(C_1, C_0)$ is the most recent common ancestor of C_1 and C_0 (i.e., the *meet* in lattice theoretic terms). Note that the distance between a concept and itself here is equal to 1 to avoid division by zero. The rationale behind this simple method is to consider concepts which are "closer" to the target concept more similar. For example, the music genre *Alternative Rock* will be considered more similar to the target concept *Rock* than the genre *Dance*. The main problem with this approach is the fact that it is not consistent with the *IS-A* relation assumption. It may consider concepts which are not a sub-concept of the target concept more similar than concepts under the target concept. We can see this directly in Figure 3(a):

Example 2. *Consider the user's target concept to be* Dance. *Concepts within the same distance from this target concept (e.g. Trance, Techno, Electronic) are all considered at the same level regardless of their IS-A relation with* Dance.

This naïve approach, being less intuitive, can however serve well as a benchmark method for comparing other methods.

4.3.2 Most Specific Shared Information

So far we have concentrated on the similarity between sub-concepts under the target concept. We saw different ways of treating this information. Another very important issue when dealing with hierarchy-based similarity methods is the similarity between the different concepts not inherited from the target concept and that target concept. Consider the following query over our music ontology: the user has asked for a music album of the genre Trance (possibly among other criteria). There is no album of this genre or a sub-concept of Trance which satisfy the request. We do consider an album of genre Electronic and an album of genre Techno. Up until now, these two concepts will be considered as equally similar to Trance (both with distance 3 from Trance). The problem here is that it may well be argued that Techno is more similar to Trance than Electronic since Trance and Techno *share more specific information.* Both Trance and Techno inherit from Dance which means that they share the special properties of Dance that distinguish it from other types of Electronic music. (Wu & Palmer 1994) presented a similarity measurement (in the context of linguistics) which takes the level of shared information into account:

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3} \tag{4}$$

Where N_1, N_2 are the distances from the concepts C_0 and C_1 to their MRCA respectively and N_3 is the distance from this MRCA and the root of the ontology (assuming the most general concept is the OWL concept *Thing*).

Example 3. *In our example,* $Sim(Trance, Electronic) = \frac{2*2}{2+0+2*2} = 0.67$ *while* $Sim(Trance, Techno) = \frac{2*3}{1+1+2*3} = 0.75$ *and thus Techno will be considered more similar to Trance than Electronic and will dominate it in the context of music genres.*

As for (2), a major drawback in using this method to compute similarities in ontology hierarchies is that it does not respect the *IS-A* relation axiom, as can be seen directly in Figure 3(b):

Example 4. *Let us consider again the user's target concept to be* Dance. *The concept* Electronic *has a similarity measurement of* 0.8 *to the target concept while the concept* ProgressiveTrance, *being a sub-concept of* Dance *has a smaller similarity measurement of* 0.75 *and thus considered less similar to* Dance *than* Electronic.

It may be argued that even though Progressive-Trance, being a very specific type of Dance music, is indeed Dance, due to its specific characteristics it may have gone far enough from this target concept to be considered less similar to it compared to a concept outside that target concept. We find no justification for making such an argument mainly due to the fundamental significance of the *IS-A* relation in description logic conceptual inheritance. We propose here a method for measuring similarities between concepts while considering specific shared information more similar and preserving the *IS-A* DL axiom.

4.3.3 Most Specific Shared Information w.r.t Ontology Depth

In this paper we measure similarity between concepts while considering concepts that share more specific information to be more similar and preserve the DL semantics of the *IS-A* relation. We modify (4) by reducing the similarity measurement in relation to the depth of the compared concepts and the maximal depth of the relevant part of the ontology (further details are in Section 5.1). We propose a novel similarity method, based on (Wu & Palmer 1994), which has three interesting properties:

1. It considers two concepts more similar if they share more specific information.
2. It respects the *IS-A* relation axiom as described above.
3. Within a sub-graph, it will consider a concept more similar to a target concept according to the communicated level of specificity described by this target concept.

Property 3 means that when the user specifies a certain target (most preferred) concept, the sub-concepts below this target concept will be ordered according to their distance to the target concept (the 'closer' distance, the more similar they are). The intuition behind this is to respect the user's communicated level of specificity (given by the depth of this target concept in the ontology) considering concepts which are 'closer' to this level of specificity to be more similar. This is measured by the following similarity metric which determines the similarity of concepts C_0 and C_1.

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG} \tag{5}$$

Where N_1, N_2 and N_3 are defined as in (4), AVG is the average distance of MAX to the depth of the concepts C_0 and C_1 and MAX is the length of the longest path from the root of the ontology to any of its leaf concepts. Note that in practice we then normalise the similarity measurement to be between 0 and 1 by dividing it by the similarity of the target concept to itself. This way we ensure that the similarity between the target concept and itself will always be 1 (which accords with intuition).

Example 5. *In our running example,* $Sim(Trance, Electronic) = \frac{2*2}{2+0+2*2+2} = 0.5$ *while* $Sim(Trance, Techno) = \frac{2*3}{1+1+2*3+1} = 0.67$ *and thus, as for the previous method, Techno will be considered more similar to Trance than Electronic. But now, as opposed to (4), this methods will still give a smaller similarity measurement (*0.53*) between Dance and Electronic and a greater similarity measurement between Dance and any sub-concept of Dance, e.g.* 0.67 *for the similarity between Dance and ProgressiveTrance.*

We can prove that (5) guarantees that sub-concepts are always preferred to non-sub-concepts w.r.t a target concept.

Theorm 1. *Let* $\prec$ *be defined by (1) with* Sim *defined as in (5). Let* C_0*,* C_1*,* C_2 *be concepts in an ontology such that* $C_1 \sqsubseteq C_0$ *and* $C_2 \not\sqsubseteq C_0$*. For all result bindings* b_1*,* b_2 *such that* $C(b_1) \in C_1$ *and* $C(b_2) \in C_2$ *it holds that* $Sim(C_0, C_1) > Sim(C_0, C_2)$ *(hence* $b_2 \prec_{P(C_0)} b_1$*).*

It can be observed that Equation 5 induces a total preorder over concepts satisfying asymmetry and transitivity. The asymmetry property comes directly from point 2 above. The power of this method is that it encapsulates all the discussed intuitions. It satisfies the *IS-A* relation axiom, always considering sub-concepts of a target concept more similar than non-sub concepts w.r.t this target concept. It also considers concepts that share more specific information more similar unless the previous statement does not hold. We discuss this method further in the analysis (6).

4.4 Exploiting Property Structure for Querying with Preferences

Similarly to the way concepts in an ontology can be organized in an hierarchical manner, an hierarchical structure can be applied to properties (roles) as well. RDF provides us with the ability to define sub properties through the *subPropertyOf* operator. Typically, these properties will have the same domain and range. Consider the property structure given in Figure 2 defining a (partial) structure over roles in music album production. All properties which inherit from the property *participates* will have the signature: $participates(MusicPerson) \rightarrow MusicAlbum$. It is a fact that some of the best known musicians, as well as creating their own music, also participate in the production of other albums (e.g. Brian Eno, Robert Fripp etc). By nature, the work engineered by some musicians, for example, will be significantly different to the music they compose or perform. On the other hand, it is safe to say that a certain individual's participation in an album as an electric guitar performer will have more similar contribution to an album where the same individual played the bass guitar than an album where they participated as a record producer. In our preference reasoning, we exploit this structure in order to provide more accurate results in a similar manner to the way we exploited conceptual structure. Similarly to structural-based operators available for preferences over concepts, the syntax for constructing preferences over properties in the ontology is available through the *Sim* and the *IS-A* operators:

$$\texttt{?prop} \sim= C_0$$

Where `?prop` is a variable which has to appear as the predicate in some triple block in the `WHERE` clause of the SPARQL query. The same syntax can be applied for using *IS-A*. For example:

```
Prefix music:
    <http://example.com/music.owl#>
SELECT ?id ?genre ?property
FROM <http://example.com/music.owl>
WHERE {
  ?id music:hasGenre ?genre.
  ?id ?property :MyMusicPerson.
  ?id rdf:type :MusicAlbum. }
PREFERRING
  ?property ~= music:keyboards_in
```

The semantics of the preference reasoner will be similar to the semantics for reasoning with concepts. From the technical point of view, SPARQL allows us to place variables as properties. These will be bound through the execution of the query to anything that will match the domain and range. In our example, the variable `?property` will be bound to any function from *MusicAlbum*[1] to *MusicPerson*.[2] Once the variable is bound we use the *RDFS* property *subPropertyOf* to analyze the property structure.

5 Implementation—The *SPOQE* System

An implementation of the methods proposed here have been completed based on the *ARQ* SPARQL query engine (Seaborne 2005) (a *Jena* based query engine) and building on the implementation of (Siberski et al. 2006) where the iterative processing of preference querying is performed as a *solution modifier* (similarly to the way the classical sorting functionality `ORDER BY` is done). A demo of the system – *SPOQE* (Similarity-based Preferences over Ontologies Query Engine) is available online. [3] On top of the described similarity measurement methods, we supplemented this implementation by adding some further numerical preference attributes mentioned in (Kießling & Köstler 2002) such as `AROUND` and `BETWEEN`.

5.1 Reasoning in Context—Conceptual Views

When a large ontology is available, we may wish to limit our search over similar concepts in the ontology purely because some parts of the ontology are not relevant to our query. For example, if part of our ontology relates to music and another part to sports, when looking for music recommendations there is no need to consider concepts in the sports part of the ontology. In such cases, it makes sense to limit the search for similar concepts and, in fact, denote that sections of the ontology are irrelevant to other sections and need not be searched over. (Balke & Wagner 2004) refers to these sections in an ontology as *Conceptual Views* and models these as the concepts which inherit directly from the top concept *Thing*. In this paper, we allow the ontology designer to be able to make the decision in regards to what constitutes a conceptual view. To effect this we adopt a unary function $isConceptView(C)$ that takes a concept as argument. This denotes that only concepts $D \sqsubseteq C$ are to be considered similar to one another (according to the metric adopted). Other concepts are considered to be "irrelevant" with $Sim(D, E) = 0$ for concepts $D \sqsubseteq C$ but $E \not\sqsubseteq C$. In other words, this feature serves to restrict searches to relevant portions of the ontology, i.e. Conceptual Views. This feature can also significantly limit the search space when performing our structure-based reasoning.

5.2 Computing Top-k Elements

Our system offers two query models: the user can either search for the elements which best match their preferences (BMO—Best Match Only query model) or, retrieve the top-k elements. Using top-k is useful not only in order to experiment with preference orderings but also since it may be the case that a possible result binding is dominated by one (or a few) other result bindings but still very similar to the optimal solution and may simply be 'good enough' (especially when dealing with similarity as the basis for comparison). In BMO, a 'competition loop' takes place where once a result binding (or a database tuple) has been dominated it is no longer considered a best match and thus will not be returned to the user. In our system, top-k selection is made without losing the qualitative nature of our reasoning, i.e. without using direct scoring of individuals: while performing the competition loop we give a score to an individual not according to its content but by examining how many 'competitions' it wins, how many it loses and how many resulted in a draw (neither individuals dominated the other). This is a round robin competition loop since we do not stop when a result binding was found to be dominated. For example, a result binding b_1 competing against another result binding b_2 will receive:

- 2 points if it dominates b_2, i.e. $b_2 \prec_P b_1$.
- 1 point if it does not dominate b_2 and b_2 does not dominate it i.e. $b_2 \equiv_P b_1$.

[1] We explicitly ensure this in the query by binding the variable *?id* to elements of type *MusicAlbum* although this is not mandatory.

[2] Assuming *MyMusicPerson* is of type *MusicPerson*.

[3] http://spoqe.web.cse.unsw.edu.au/

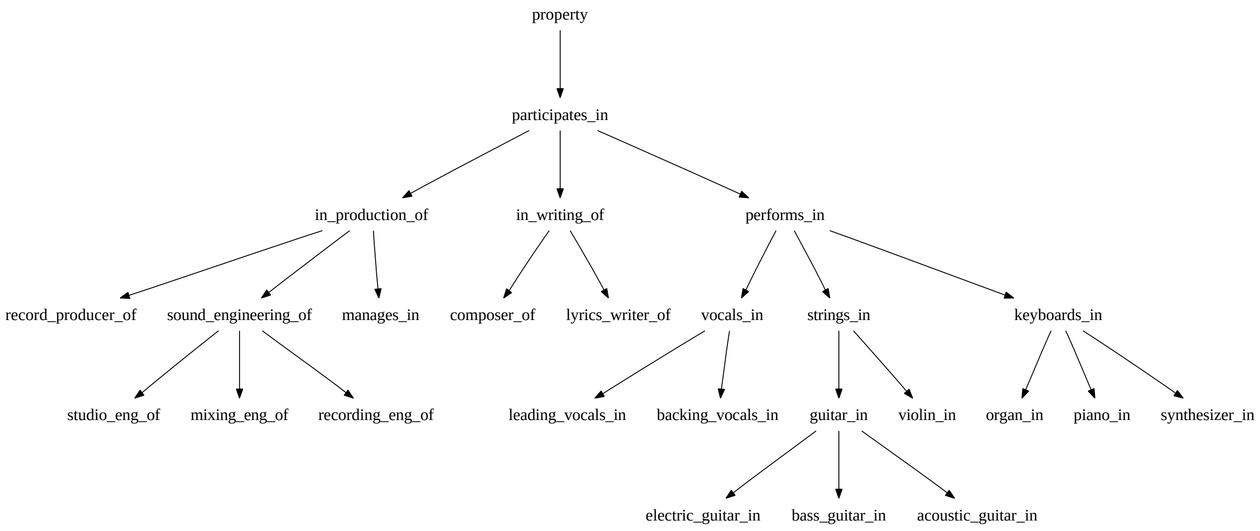


Figure 2: Ontological Property Hierarchy of Music Album Production

- 0 points if it is dominated by b_2, i.e. $b_1 \prec_P b_2$.

At the end of this process, we sort the individuals according to their accumulated score. As in BMO, the best match result bindings will appear at the top of the preference ordering but now just below them the result bindings which are closer to the preference query (but not perfectly matching it) will appear.

6 Analysis

We evaluate our similarity method by looking at some intuitive examples and examining the behaviour of the resulting total orders. We base our analysis on the concept similarity method defined in (5). Let us have a closer look at this operator. In order to preserve the properties we listed in Section 4.3.3, this operator was designed so it will have different behaviours when comparing a target concept with a descendant to when comparing a target concept to an ancestor. It can be deduced that if $C_1 \sqsubseteq C_0$ and $C_2 \sqsubseteq C_1$ then

$$Sim(C_1, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG}$$

$$Sim(C_2, C_0) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3 + AVG + \frac{1}{2}\Delta}$$

Where Δ is the the direct distance between C_1 and C_2. And in the case where $C_0 \sqsubseteq C_1$ and $C_1 \sqsubseteq C_2$:

$$Sim(C_2, C_0) = \frac{2 * N_3 + 2\Delta}{N_1 + N_2 + 2 * N_3 + AVG - 1\frac{1}{2}\Delta}$$

Example 6. *When assigning the target concept to be* Dance *(Figure 3(c)) we see that the similarity measurement linearly decreases for descendant concepts w.r.t* Dance *while every 'climb' up to an ancestor reduces the similarity measurement significantly (and non-linearly). Intuitively, you can say that the difference in similarity when comparing elements 'in the area of' Dance is bigger than the difference in similarity between elements which are quite 'far' from this target concept and thus the ordering according to it is less relevant. Note that for clarity we normalise the similarity measurement to be between 0 and 1 by dividing it by the similarity of the target concept to itself.*

7 Discussion

While the method that we provide here for exploiting ontological structure to enhance preferential querying leads to a much more discriminating recommendation system, there are some limitations that can be further explored. We consider two such issues here. One has to do with the expressiveness of user preference queries while the other has to do with the structure of the ontology that is used.

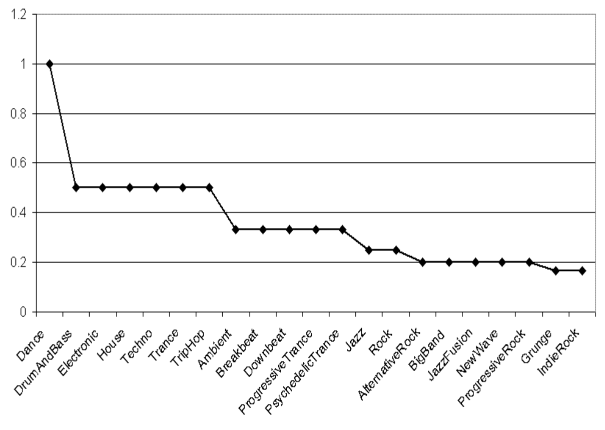
7.1 Receiving Partial Preorder from the User

In this paper the user is limited in the way they can express preferences. It is not possible to express that two classifications are equally preferred by the user for instance. Furthermore, `CASCADE` only allows for limited levels of preferences. In (Chamiel & Pagnucco 2008*b*) we propose a preference assembly method where the user can specify an ordering over concept classes which allows for classes to be equally preferred. That is, to specify a preference as a partial pre-order

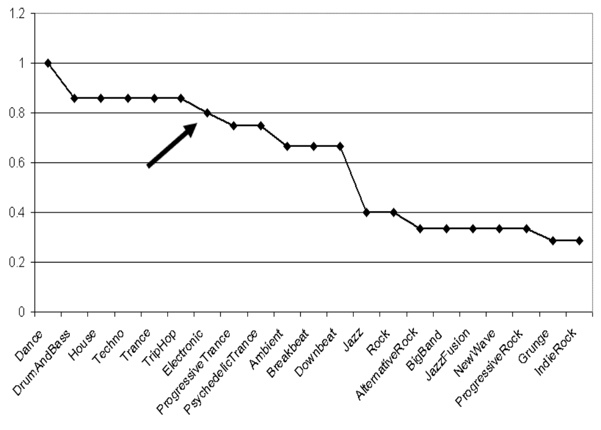
$$\{C_{1,1}, C_{1,2}, \cdots, C_{1,i}\} > \cdots > \{C_{n,1}, C_{n,2}, \cdots, C_{n,k}\}$$

Each set $\{C_{l,1}, C_{l,2}, \cdots, C_{l,n}\}$ represents concepts that the user equally prefers. We only require that preferences be consistent; i.e., transitivity and asymmetry is preserved. Furthermore, we have an ontology for the domain at hand in OWL format. We can turn this partial pre-order into a total order by "filling out" (or completing) user preferences with information from the ontology by utilising notions of similarity such as that described here. We end up with a total order $C_1 > \cdots > C_m$ over all concepts in the ontology with those corresponding to concepts specified in the user preference maintaining the order imposed by the user. Concepts not explicitly ordered by the user are arranged in a consistent way so that they occur after the concept in the user ordering to which they are most similar. In this way we end up with a fine-grained recommendation mechanism like the one developed in this paper but allowing the user to express more of their preferences and hence exert greater influence over the final orderings (and therefore the final recommendation). Syntactically, this could be expressed as follows (and we could introduce a similar syntax for the IS-A relation):

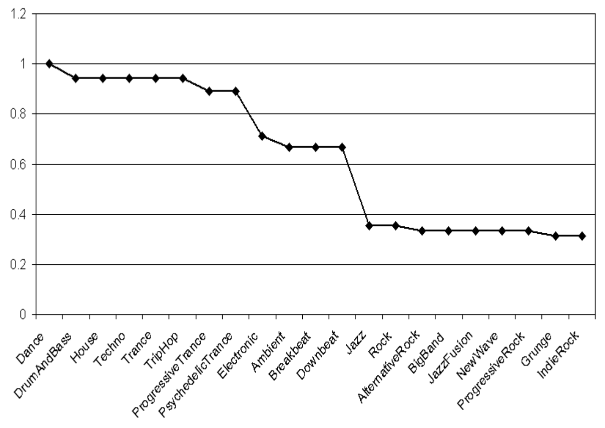
$$\texttt{?var} \sim= (C_{1,1}...C_{1,i})\ \texttt{THEN}\ (C_{2,1}...C_{2,j})\ \texttt{THEN}\ \cdots\ \texttt{THEN}\ (C_{n,1}...C_{n,k})$$



(a) Direct Graph Distance



(b) Wu and Palmer



(c) Method Proposed in (5)

Figure 3: Structure-based similarity methods comparison w.r.t the target concept *Dance*. The x-axis represents concept classes in the ontology while the y-axis represents the similarity of these concepts to the target concept *Dance*.

7.2 Weighted Edges in an Ontological Graph

The methods described in this paper utilize graph distance to perform reasoning and enhance preferential selection. However, so far all edges of the graph (i.e., ontology hierarchy) are considered to have the same distance (or weight). This relies on the underlying assumption that all sections of the ontology hierarchy have been developed to the same 'level of specificity'. In reality some edges might describe a 'closer' relation than others. For example, in our music example we may argue that not all concept classifications at the same level of the hierarchy represent the same level of specificity in their categorisation of music. This situation can result for a number of reasons. In some cases it represents uncertainty about the domain and the classifications made in it; there may be more information regarding certain parts of the ontology while less information for other parts and this is reflected in the depth of the hierarchy. In other cases, the ontology hierarchy cannot be uniformly developed throughout. The ontology engineer has no means of defining this (besides creating bogus hierarchical levels) and it may be very difficult for them to make the different distinctions and weight the edges manually. One way of dealing with this issue is to allow the ontology engineer to attach weights to the edges in the OWL ontology graph. This however is quite laborious and tedious task. Another way is to assign graph edge weights automatically by examining the depth of a sub-graph where an edge which has a deeper (and thus richer) sub-graph under it will receive a smaller weight to demonstrate its smaller distance from the parent concept. This assumes that all parts of the ontology have been developed to the same level of specificity and may lead to better recommendations when adopting the technique described here.

8 Conclusions

In this paper we have enhanced preference querying by supplementing the user's preferences with ontological information; in particular, by exploiting the structural properties of the ontology describing the domain at hand. We claim that this is significant for a vast class of ontologies that we refer to as *data inheritance systems*. By introducing a notion of similarity based on distance metrics, we free the user from having to possess a deep understanding of the underlying domain. In particular, we adapt one similarity measure (4) to introduce another (5). Ontologies are developed by domain experts who have intimate knowledge of their subject area and are exploited by naive users who can specify weaker preferences without the fear of being overwhelmed by the results. Moreover, the results are not biased by the whims of previous user consumption but by exploiting domain expertise. Our proposals are implemented on ARQ and enhance the SPARQL standard.

References

Antoniou, G. & van Harmelen, F. (2004), *A Semantic Web Primer (Cooperative Information Systems)*, MIT Press.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. & Patel-Schneider, P. F., eds (2003), *The description logic handbook: Theory, implementation, and applications*, Cambridge University Press, New York, NY, USA.

Balke, W.-T. & Wagner, M. (2004), Through different eyes: assessing multiple conceptual views for querying web services, *in* 'WWW '04: Proceedings of the 13th international World Wide Web', ACM, New York, NY, USA, pp. 196–205.

Bradley, K., Rafter, R. & Smyth, B. (2000), 'Case-based user profiling for content personalisation', *Lecture Notes in Computer Science* **1892**, 62–72.

Chamiel, G. & Pagnucco, M. (2008*a*), Exploiting ontological information for reasoning with preferences, *in* 'Proceedings of the Fourth Multidisciplinary Workshop on Advances in Preference Handling'.

Chamiel, G. & Pagnucco, M. (2008*b*), Querying the semantic web by assembling complex user preferences and ontological structure, Technical Report UNSW-CSE-TR-0817, School of Computer Science and Engineering, University of New South Wales.

Doyle, J. & Thomason, R. H. (1999), 'Background to qualitative decision theory', *AI Magazine* **20**(2), 55–68.

Fishburn, P. (1999), 'Preference structures and their numerical representations', *Theor. Comput. Sci.* **217**(2), 359–383.

Kiefer, C., Bernstein, A. & Stocker, M. (2007), The fundamentals of iSPARQL – A virtual triple approach for similarity-based semantic web tasks, *in* 'ISWC '07'.

Kießling, W. (2002), Foundations of preferences in database systems, *in* 'VLDB', pp. 311–322.

Kießling, W. & Köstler, G. (2002), Preference SQL: design, implementation, experiences, *in* 'VLDB', pp. 990–1001.

Middleton, S. E., Shadbolt, N. R. & De Roure, D. C. (2004), 'Ontological user profiling in recommender systems', *ACM Trans. Inf. Syst.* **22**(1), 54–88.

Prud'hommeaux, E. & Seaborne, A. (2006), SPARQL query language for RDF, Technical report, W3C Candidate Recommendation.
URL: *http://www.w3.org/TR/rdf-sparql-query/*

Sarwar, B. M., Karypis, G., Konstan, J. A. & Reidl, J. (2001), Item-based collaborative filtering recommendation algorithms, *in* 'World Wide Web', pp. 285–295.

Schickel-Zuber, V. & Faltings, B. (2006), Inferring User's Preferences using Ontologies, *in* 'AAAI 2006', pp. 1413–1418.

Schickel-Zuber, V. & Faltings, B. (2007), Using hierarchical clustering for learning the ontologies used in recommendation systems, *in* 'KDD'07', pp. 599–608.

Seaborne, A. (2005), 'ARQ - A SPARQL processor for Jena. http://jena.sourceforge.net/arq'.

Siberski, W., Pan, J. Z. & Thaden, U. (2006), Querying the semantic web with preferences, *in* 'International Semantic Web Conference', pp. 612–624.

Wu, Z. & Palmer, M. (1994), Verb semantics and lexical selection, *in* '32nd Annual Meeting of the Association for Computational Linguistics', New Mexico State University, Las Cruces, New Mexico, pp. 133 –138.

An Argumentative Approach to Reasoning with Inconsistent Ontologies

Sergio Alejandro Gómez[1] **Carlos Iván Chesñevar**[1,2]
Guillermo Ricardo Simari[1]

[1] Artificial Intelligence Research and Development Laboratory,
Department of Computer Science and Engineering,
Universidad Nacional del Sur,
Av. Alem 1253 - (8000) Bahía Blanca, Argentina,
Email: {sag,cic,grs}@cs.uns.edu.ar
[2]CONICET (National Council of Scientific and Technical Research), Argentina

Abstract

Standard approaches to reasoning with Description Logics (DL) ontologies require them to be consistent. However, as ontologies are complex entities and sometimes built upon other imported ontologies, inconsistencies can arise. In this paper, we present a framework for reasoning with inconsistent DL ontologies. Our proposal involves expressing DL ontologies as Defeasible Logic Programs (DeLP). Given a query posed w.r.t. an inconsistent ontology, a dialectical analysis will be performed on a DeLP program obtained from such ontology where all arguments in favor and against the final answer of the query will be taken into account. We also present an application to ontology integration based on the global-as-view approach.

Keywords: Semantic Web, Description Logics, defeasible argumentation, Defeasible Logic Programming, inconsistent ontology handling, ontology integration

1 Introduction and Motivations

The *Semantic Web* (Berners-Lee et al. 2001) is a future vision of the web where stored information has exact meaning, thus enabling computers to understand and reason on the basis of such information. Assigning semantics to web resources is addressed by means of *ontology definitions* (Gruber 1993).

As proposed by the World Wide Web Consortium[1], ontology definitions are meant to be written in an *ontology description language* such as OWL (McGuiness & van Harmelen 2004), whose subset known as OWL-DL is based on Description Logics (DL) (Baader et al. 2003). Although ontology definitions expressed in DL can be processed with existing DL reasoners (*e.g.* Racer (Haarslev & Möller 2001)), such DL reasoners are incapable of dealing with *inconsistent* ontology definitions.

This situation is particularly important in the Semantic Web setting, where ontologies are complex entities prone to suffer inconsistencies (Huang et al. 2005). A particular source of inconsistency is related to the use of imported ontologies when the knowledge engineer has no authority to correct them. As these imported ontologies are usually developed independently, their combination could also result in inconsistencies. The problem of combining two or more different ontologies in order to obtain a unified, consistent ontology is known as *ontology integration* (Klein 2001). One kind of such integration is known as *global-as-view* integration (Calvanese et al. 2001), where a global ontology is used as a view of local ontologies that define the actual data.

There are two main ways to deal with inconsistency in ontologies (Huang et al. 2005): one is to diagnose and repair it when it is encountered; another is to avoid the inconsistency and to apply a non-standard inference relation to obtain meaningful answers. Although there are existing approaches for the former (*e.g.* identifying the minimally unsatisfiable sub-ontologies or calculating the maximally satisfiable sub-ontologies), in this work we propose to use *defeasible argumentation* (Chesñevar et al. 2000, Prakken & Vreeswijk 2002) to focus on the latter. Grosof et al. (2003) have determined that a subset of DL can be effectively translated into an equivalent subset of Horn logic. In particular, *DeLP* is an argumentative framework based on logic programming that is capable of dealing with possibly inconsistent knowledge bases (KB) codified as a set of Horn-like clauses called DeLP programs (García & Simari 2004). When presented with a query, DeLP performs a *dialectical* process in which all *arguments* in favor and against a conclusion are considered; arguments regarded as ultimately *undefeated* will be considered *warranted.*

In this article we propose a framework for representing possibly inconsistent DL ontologies. Reasoning in such ontologies will be carried out by means of a dialectical analysis. Our proposal involves mapping DL ontologies into a DeLP program. That is, given a DL ontology Σ_{DL}, provided Σ_{DL} satisfies certain restrictions, it will be translated into a DeLP program Σ_{DeLP}. Given a query ϕ, a dialectical process will be performed to determine if ϕ is warranted w.r.t. Σ_{DeLP}. Besides, we apply our proposal to perform global-as-view integration when the involved ontologies can be potentially inconsistent.

The rest of the article is structured as follows. Section 2 introduces the fundamentals of Description Logics. Section 3 briefly explains the Defeasible Logic Programming formalism. Section 4 explains how DL ontologies are going to be translated into DeLP. Section 5 introduces δ-ontologies, a particular kind of DL ontologies amenable for its treatment within DeLP. Section 6 presents an application to on-

This research was funded by Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 No. 13.096, PICT 2003 No. 15.043, PAV 2004 076), by CONICET (Argentina), by Project TIN2006-15662-C02-01 (MEC, Spain), Project 24/ZN10 (SGCyT, UNS).

[1]www.w3c.org

tology integration based on defeasible argumentation. Section 7 enumerates some properties of the proposed approach. Section 8 discusses related work. Finally Section 9 concludes.

2 Description Logics

Description Logics (DL) are a well-known family of knowledge representation formalisms (Baader et al. 2003). They are based on the notions of *concepts* (unary predicates, classes) and *roles* (binary relations), and are mainly characterized by constructors that allow complex concepts and roles to be built from atomic ones. The expressive power of a DL system is determined by the constructs available for building concept descriptions, and by the way these descriptions can be used in the *terminological* (*Tbox*) and *assertional* (*Abox*) components of the system.

We now describe the basic language for building DL expressions. Let C and D stand for concepts and R for a role name. Concept descriptions are built from concept names using the constructors conjunction ($C \sqcap D$), disjunction ($C \sqcup D$), negation ($\neg C$), existencial restriction ($\exists R.C$), and value restriction ($\forall R.C$). To define the semantics of concept descriptions, concepts are interpreted as subsets of a domain of interest, and roles as binary relations over this domain. An interpretation I consists of a non-empty set Δ^I (the domain of I) and a function $\cdot^I$ (the interpretation function of I) which maps every concept name A to a subset A^I of Δ^I, and every role name to R to a subset R^I of $\Delta^I \times \Delta^I$. The interpretation function is extended to arbitrary concept descriptions as follows: $(\neg C)^I = \Delta^I \backslash C^I$; $(C \sqcup D)^I = C^I \cup D^I$; $(C \sqcap D)^I = C^I \cap D^I$; $(\exists R.C)^I = \{x | \exists y \text{ s.t. } (x, y) \in R^I \text{ and } y \in C^I\}$, and $(\forall R.C)^I = \{x | \forall y, (x, y) \in R^I \text{ implies } y \in C^I\}$. Besides, the expressions $\top$ and $\bot$ are shorthands for $C \sqcup \neg C$ and $C \sqcap \neg C$, resp. Further extensions to the basic DL are possible including inverse and transitive roles noted as P^- and P^+, resp.

A DL *ontology* $\Sigma = (T, A)$ consists of two finite and mutually disjoint sets: the Tbox T which introduces the *terminology* and the Abox A which contains facts about particular objects in the application domain. Tbox statements have the form $C \sqsubseteq D$ (*inclusions*) and $C \equiv D$ (*equalities*), where C and D are (possibly complex) concept descriptions.

The semantics of Tbox statements is as follows. An interpretation I satisfies $C \sqsubseteq D$ iff $C^I \subseteq D^I$, I satisfies $C \equiv D$ iff $C^I = D^I$. Objects in the Abox are referred to by a finite number of *individual names* and these names may be used in two types of assertional statements: *concept assertions* of the type $a : C$ and *role assertions* of the type $\langle a, b \rangle : R$, where C is a concept description, R is a role name, and a and b are individual names. An interpretation I *satisfies* the assertion $a : C$ iff $a^I \in C^I$, and it *satisfies* $\langle a, b \rangle : R$ iff $(a^I, b^I) \in R^I$. An interpretation I is a *model* of a DL (Tbox or Abox) statement ϕ iff it satisfies the statement, and is a model of a DL ontology Σ iff it satisfies every statement in Σ. A DL ontology Σ *entails* a DL statement ϕ, written as $\Sigma \models \phi$, iff every model of Σ is a model of ϕ.

3 Defeasible Logic Programming

Defeasible Logic Programming (DeLP) (García & Simari 2004) provides a language for knowledge representation and reasoning that uses *defeasible argumentation* (Chesñevar et al. 2000, Prakken & Vreeswijk 2002, Simari & Loui 1992) to decide between contradictory conclusions through a *dialectical analysis.* Recent research has shown that DeLP provides a suitable framework for building real-world applications that deal with incomplete and potentially contradictory information.

In a defeasible logic program $\mathcal{P} = (\Pi, \Delta)$, a set Δ of defeasible rules $P \prec Q_1, \ldots, Q_n$, and a set Π of strict rules $P \leftarrow Q_1, \ldots, Q_n$ can be distinguished.

Definition 1 (Strict, defeasible and DeLP programs) *$\mathcal{L}_{DeLP} =_{df} \mathcal{L}_{DeLP_\Pi} \cup \mathcal{L}_{DeLP_\Delta}$ is the language of DeLP programs, where $\mathcal{L}_{DeLP_\Pi}$ is the language of DeLP programs formed by strict rules $B \leftarrow A_1, \ldots, A_n$ with $(n \geq 1)$ and facts B (i.e., rules where $n = 0$), and $\mathcal{L}_{DeLP_\Delta}$ is the language of DeLP programs formed only by defeasible rules $B \prec A_1, \ldots, A_n$ with $(n \geq 1)$.*

Literals can be positive or negative. The complement of a literal L (noted as $\overline{L}$) is p if $L = \sim p$ and $\sim p$ if $L = p$. Deriving literals in DeLP results in the construction of *arguments.* An argument $\mathcal{A}$ is a (possibly empty) set of ground defeasible rules that together with the set Π provides a logical proof for a given literal Q, satisfying the additional requirements of *non-contradiction* and *minimality.* Formally:

Definition 2 (Argument) *Given a DeLP program $\mathcal{P}$, an* argument *$\mathcal{A}$ for a query Q, denoted $\langle \mathcal{A}, Q \rangle$, is a subset of ground instances of defeasible rules in $\mathcal{P}$, such that: (1) there exists a* defeasible derivation *for Q from $\Pi \cup \mathcal{A}$; (2) $\Pi \cup \mathcal{A}$ is non-contradictory (i.e., $\Pi \cup \mathcal{A}$ does not entail two complementary literals P and $\sim P$), and, (3) there is no $A' \subseteq A$ such that there exists a defeasible derivation for Q from $\Pi \cup \mathcal{A}'$. An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a* sub-argument *of another argument $\langle \mathcal{A}_2, Q_2 \rangle$ if $\mathcal{A}_1 \subseteq \mathcal{A}_2$.*

The notion of defeasible derivation corresponds to the usual query-driven SLD derivation used in logic programming, performed by backward chaining on both strict and defeasible rules; in this context a negated literal $\sim P$ is treated just as a new predicate name *no_P*. Minimality imposes a kind of 'Occam's razor principle' (Simari & Loui 1992) on argument construction. The non-contradiction requirement forbids the use of (ground instances of) defeasible rules in an argument $\mathcal{A}$ whenever $\Pi \cup \mathcal{A}$ entails two complementary literals. The notion of contradiction is captured by the notion of counterargument.

Definition 3 (Counterargument. Defeat) *An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a* counterargument *for an argument $\langle \mathcal{A}_2, Q_2 \rangle$ iff there is a subargument $\langle \mathcal{A}, Q \rangle$ of $\langle \mathcal{A}_2, Q_2 \rangle$ such that the set $\Pi \cup \{Q_1, Q\}$ is contradictory. An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a* defeater *for an argument $\langle \mathcal{A}_2, Q_2 \rangle$ if $\langle \mathcal{A}_1, Q_1 \rangle$ counterargues $\langle \mathcal{A}_2, Q_2 \rangle$, and $\langle \mathcal{A}_1, Q_1 \rangle$ is preferred over $\langle \mathcal{A}_2, Q_2 \rangle$ w.r.t. a* preference criterion *$\preceq$ on conflicting arguments. Such criterion is defined as a partial order $\preceq \subseteq Args(\mathcal{P}) \times Args(\mathcal{P})$. The argument $\langle \mathcal{A}_1, Q_1 \rangle$ will be called a* proper defeater *for $\langle \mathcal{A}_2, Q_2 \rangle$ iff $\langle \mathcal{A}_1, Q_1 \rangle$ is strictly preferred over $\langle \mathcal{A}, Q \rangle$ w.r.t. $\preceq$; if $\langle \mathcal{A}_1, Q_1 \rangle$ and $\langle \mathcal{A}, Q \rangle$ are unrelated to each other will be called a* blocking defeater *for $\langle \mathcal{A}_2, Q_2 \rangle$.*

Generalized specificity (Simari & Loui 1992) is typically used as a syntax-based criterion among conflicting arguments. However, other alternative partial orders could also be valid.

In order to determine whether a given argument $\mathcal{A}$ is ultimately undefeated (or *warranted*), a dialectical process is recursively carried out, where defeaters for $\mathcal{A}$, defeaters for these defeaters, and so on, are taken into account. An *argumentation line* starting in an argument $\langle \mathcal{A}_0, Q_0 \rangle$ is a sequence $[\langle \mathcal{A}_0, Q_0 \rangle, \langle \mathcal{A}_1, Q_1 \rangle,$

$\langle \mathcal{A}_2, Q_2\rangle, \ldots, \langle \mathcal{A}_n, Q_n\rangle \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). Each $\langle \mathcal{A}_i, Q_i\rangle$ is a defeater for the previous argument $\langle \mathcal{A}_{i-1}, Q_{i-1}\rangle$ in the sequence, $i > 0$. In order to avoid *fallacious* reasoning, dialectics imposes additional constraints on such an argument exchange to be considered rationally *acceptable*. Given a DeLP program $\mathcal{P}$ and an initial argument $\langle \mathcal{A}_0, Q_0\rangle$, the set of all acceptable argumentation lines starting in $\langle \mathcal{A}_0, Q_0\rangle$ accounts for a whole dialectical analysis for $\langle \mathcal{A}_0, Q_0\rangle$ (*i.e.*, all possible dialogues about $\langle \mathcal{A}_0, Q_0\rangle$ between proponent and opponent), formalized as a *dialectical tree.*

Nodes in a dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0\rangle}$ can be marked as *undefeated* and *defeated* nodes (U-nodes and D-nodes, resp.). A dialectical tree will be marked as an AND-OR tree: all leaves in $\mathcal{T}_{\langle \mathcal{A}_0, Q_0\rangle}$ will be marked U-nodes (as they have no defeaters), and every inner node is to be marked as *D-node* iff it has at least one U-node as a child, and as *U-node* otherwise. An argument $\langle \mathcal{A}_0, Q_0\rangle$ is ultimately accepted as valid (or *warranted*) w.r.t. a DeLP program $\mathcal{P}$ iff the root of its associated dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0\rangle}$ is labeled as *U-node*.

Given a DeLP program $\mathcal{P}$, solving a query Q w.r.t. $\mathcal{P}$ accounts for determining whether Q is supported by (at least) one warranted argument. Different doxastic attitudes can be distinguished as follows: *Yes*, accounts for believing Q iff there is at least one warranted argument supporting Q on the basis of $\mathcal{P}$; *No*, accounts for believing $\sim Q$ iff there is at least one warranted argument supporting $\sim Q$ on the basis of $\mathcal{P}$; *Undecided*, neither Q nor $\sim Q$ are warranted w.r.t. $\mathcal{P}$, and *Unknown*, Q does not belong to the signature of $\mathcal{P}$.

4 Expressing DL Ontologies in DeLP

In the presence of inconsistent ontologies, traditional DL reasoners (such as RACER (Haarslev & Möller 2001)) issue an error message and stop further processing. Thus the burden of repairing the ontology (*i.e.*, making it consistent) is on the knowledge engineer. We are interested in coping with inconsistencies such that the task of dealing with them is automatically solved by the reasoning system. We propose using DeLP to perform such a task.

Grosof et al. (2003) show that the processing of ontologies can be improved by the use of techniques from the area of logic programming; they have identified a subset of DL languages that can be effectively mapped into a Horn-clause logics. Our proposal is in part based on such research by adapting it to the DeLP framework. We therefore propose translating a DL ontology $\Sigma = (T_S \cup T_D, A)$, with $T_S \cap T_D = \emptyset$, into a DeLP program $\mathcal{P} = (\Pi, \Delta)$ by means of a mapping $\mathcal{T}$ such that $\mathcal{P} = \mathcal{T}(\Sigma)$, where $\Pi = \mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A)$ and $\Delta = \mathcal{T}_\Delta(T_D)$. We will consider the Tbox as partitioned into two disjoint sets—a *strict terminology* T_S and a *defeasible terminology* T_D. Intuitively the set Π of strict rules in $\mathcal{P}$ will correspond to the Abox A joined with T_S in Σ, and the set Δ of defeasible rules will correspond to T_D in Σ.

In the rest of this section, we will explain how to achieve the translation of DL ontologies into DeLP programs. For clarity, strict rules of the form "$H \leftarrow B_1, \ldots, B_n$" will be sometimes written as "$H \leftarrow B_1 \wedge \ldots \wedge B_n$" and defeasible rules "$H \prec B_1, \ldots, B_n$" as "$H \prec B_1 \wedge \ldots \wedge B_n$". As noted by Grosof et al. (2003), for DL sentences to be mapped into Horn-logic rules, they must satisfy certain constraints. Conjunction and universal restrictions appearing in the right-hand side of inclusion axioms can be mapped to heads of rules (called $\mathcal{L}_h$-classes). In contrast, conjunction, disjunction and existential restriction can be mapped to rule bodies whenever they occur in the left-hand side of inclusion axioms (called $\mathcal{L}_b$-classes). As equality axioms "$C \equiv D$" are interpreted as two inclusion axioms "$C \sqsubseteq D$" and "$D \sqsubseteq C$", they must belong to the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$.

Definition 4 ($\mathcal{L}_h$ language. $\mathcal{L}_h$-classes. $\mathcal{L}_b$ language. $\mathcal{L}_b$-classes. $\mathcal{L}_{hb}$ language. $\mathcal{L}_{hb}$-classes (adapted from (Grosof et al. 2003))) *Let A be an atomic class name, C and D class expressions, and R a property. In the $\mathcal{L}_h$ language, $C \sqcap D$ is a class, and $\forall R.C$ is also a class. Class expressions in $\mathcal{L}_h$ are called* $\mathcal{L}_h$-classes. *In the $\mathcal{L}_b$ language, $C \sqcup D$ is a class, and $\exists R.C$ is a class too. Class expressions in $\mathcal{L}_b$ are called* $\mathcal{L}_b$-classes. *The $\mathcal{L}_{hb}$ language is defined as the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$. Class expressions in $\mathcal{L}_{hb}$ are called* $\mathcal{L}_{hb}$-classes.

We now define the mapping from DL to DeLP. Without losing generality, we assume that ontology definitions are normalized w.r.t. negation. That is, negations in class expressions are *shifted* inwards using De Morgan's rules and well-known relations between existential and value restrictions (Baader et al. 2003).

First we show how to map DL axioms into defeasible rules. As mentioned in Section 3, defeasible rules are meant to represent possibly inconsistent information. Thus DL axioms in defeasible terminologies are going to be interpreted as *default* class inclusions.

Definition 5 ($\mathcal{T}_\Delta$ mapping from DL sentences to DeLP defeasible rules) *Let A, C, D be concepts, X, Y variables, P, Q properties. The $\mathcal{T}_\Delta : 2^{\mathcal{L}_{DL}} \rightarrow 2^{\mathcal{L}_{DeLP_\Delta}}$ mapping is defined in Fig. 1. Besides, rules of the form "$(H_1 \wedge H') \prec B$" are rewritten as two rules "$H_1 \prec B$" and "$H_2 \prec B$", rules of the form "$H_1 \prec H_2 \prec B$" are rewritten as "$H_1 \prec B \wedge H_2$", and rules of the form "$H \prec (B_1 \vee B_2)$" are rewritten as two rules "$H \prec B_1$" and "$H_1 \prec B_2$".*

$$
\begin{array}{rcl}
\mathcal{T}_\Delta(\{C \sqsubseteq D\}) & =_{df} & \left\{ \; T_h(D,X) \prec T_b(C,X) \; \right\}, \\
 & & \text{if } C \text{ is an } \mathcal{L}_b\text{-class and } D \text{ an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Delta(\{C \equiv D\}) & =_{df} & \mathcal{T}_\Delta(\{C \sqsubseteq D\}) \cup \mathcal{T}_\Delta(\{D \sqsubseteq C\}), \\
 & & \text{if } C \text{ and } D \text{ are } \mathcal{L}_{hb}\text{-classes} \\
\mathcal{T}_\Delta(\{\top \sqsubseteq \forall P.D\}) & =_{df} & \left\{ \; T_h(D,Y) \prec P(X,Y) \; \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Delta(\{\top \sqsubseteq \forall P^-.D\}) & =_{df} & \left\{ \; T_h(D,X) \prec P(X,Y) \; \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Delta(\{P \sqsubseteq Q\}) & =_{df} & \left\{ \; Q(X,Y) \prec P(X,Y) \; \right\} \\
\mathcal{T}_\Delta(\{P \equiv Q\}) & =_{df} & \left\{ \begin{array}{l} Q(X,Y) \prec P(X,Y) \\ P(X,Y) \prec Q(X,Y) \end{array} \right\} \\
\mathcal{T}_\Delta(\{P \equiv Q^-\}) & =_{df} & \left\{ \begin{array}{l} Q(X,Y) \prec P(Y,X) \\ P(Y,X) \prec Q(X,Y) \end{array} \right\} \\
\mathcal{T}_\Delta(\{P^+ \sqsubseteq P\}) & =_{df} & \left\{ \; P(X,Z) \prec P(X,Y) \wedge P(Y,Z) \; \right\} \\
\mathcal{T}_\Delta(\{s_1, \ldots, s_n\}) & =_{df} & \bigcup_{i=1}^{n} \{\mathcal{T}_\Delta(\{s_i\})\}, \text{ if } n > 1 \\
\textbf{where:} & & \\
T_h(A,X) & =_{df} & A(X) \\
T_h((C \sqcap D),X) & =_{df} & T_h(C,X) \wedge T_h(D,X) \\
T_h((\forall R.C),X) & =_{df} & T_h(C,Y) \prec R(X,Y) \\
T_b(A,X) & =_{df} & A(X) \\
T_b((C \sqcap D),X) & =_{df} & T_b(C,X) \wedge T_b(D,X) \\
T_b((C \sqcup D),X) & =_{df} & T_b(C,X) \vee T_b(D,X) \\
T_b((\exists R.C),X) & =_{df} & R(X,Y) \wedge T_b(C,Y)
\end{array}
$$

Figure 1: Mapping from DL ontologies to DeLP defeasible rules

Example 1 *Consider the DL terminology $T_D = \{(b \sqsubseteq f), (c \sqsubseteq \neg f), (c \sqcap s \sqsubseteq f)\}$ that expresses both that birds fly and that chickens do not fly unless they* *are scared. The application of the $\mathcal{T}_\Delta$ mapping to T_D yields a set Δ of defeasible rules, where $\Delta = \{(f(X) \prec b(X)), (\sim f(X) \prec c(X)), (f(X) \prec c(X), s(X))\}$.*

Next we present a mapping from DL axioms to strict rules. We are going to assume that strict terminologies are consistent (see Section 5.3).

Definition 6 ($\mathcal{T}_\Pi^*$ mapping from DL sentences to DeLP strict rules) *Let A, C, D be concepts, X, Y variables, P, Q properties. The $\mathcal{T}_\Pi^* : 2^{\mathcal{L}_{DL}} \longrightarrow 2^{\mathcal{L}_{DeLP_\Pi}}$ mapping is defined in Fig. 2. Besides, rules of the form "$(H_1 \wedge H_2) \leftarrow B$" are rewritten as two rules "$H_1 \leftarrow B$" and "$H_2 \leftarrow B$", rules of the form "$H_1 \leftarrow H_2 \leftarrow B$" are rewritten as "$H_1 \leftarrow B \wedge H_2$", and rules of the form "$H \leftarrow (B_1 \vee B_2)$" are rewritten as two rules "$H \leftarrow B_1$" and "$H \leftarrow B_2$".*

$$
\begin{array}{rcl}
\mathcal{T}_\Pi^*(\{C \sqsubseteq D\}) & =_{df} & \left\{\ T_h(D,X) \leftarrow T_b(C,X)\ \right\}, \\
 & & \text{if } C \text{ is an } \mathcal{L}_b\text{-class and } D \text{ an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Pi^*(\{C \equiv D\}) & =_{df} & \mathcal{T}_\Pi^*(\{C \sqsubseteq D\}) \cup \mathcal{T}_\Pi^*(\{D \sqsubseteq C\}), \\
 & & \text{if } C \text{ and } D \text{ are } \mathcal{L}_{hb}\text{-classes} \\
\mathcal{T}_\Pi^*(\{\top \sqsubseteq \forall P.D\}) & =_{df} & \left\{\ T_h(D,Y) \leftarrow P(X,Y)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Pi^*(\{\top \sqsubseteq \forall P^-.D\}) & =_{df} & \left\{\ T_h(D,X) \leftarrow P(X,Y)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Pi^*(\{a : D\}) & =_{df} & \left\{\ T_h(D,a)\ \right\}, \\
 & & \text{if } D \text{ is an } \mathcal{L}_h\text{-class} \\
\mathcal{T}_\Pi^*(\{\langle a,b\rangle : P\}) & =_{df} & \left\{\ P(a,b)\ \right\} \\
\mathcal{T}_\Pi^*(\{P \sqsubseteq Q\}) & =_{df} & \left\{\ Q(X,Y) \leftarrow P(X,Y)\ \right\} \\
\mathcal{T}_\Pi^*(\{P \equiv Q\}) & =_{df} & \left\{ \begin{array}{l} Q(X,Y) \leftarrow P(X,Y) \\ P(X,Y) \leftarrow Q(X,Y) \end{array} \right\} \\
\mathcal{T}_\Pi^*(\{P \equiv Q^-\}) & =_{df} & \left\{ \begin{array}{l} Q(X,Y) \leftarrow P(Y,X) \\ P(Y,X) \leftarrow Q(X,Y) \end{array} \right\} \\
\mathcal{T}_\Pi^*(\{P^+ \sqsubseteq P\}) & =_{df} & \left\{\ P(X,Z) \leftarrow P(X,Y) \wedge P(Y,Z)\ \right\} \\
\mathcal{T}_\Pi^*(\{s_1,\ldots,s_n\}) & =_{df} & \bigcup_{i=1}^n \mathcal{T}_\Pi^*(\{s_i\}), \text{ if } n > 1 \\
\textbf{where:} & & \\
T_h(A,X) & =_{df} & A(X) \\
T_h((C \sqcap D),X) & =_{df} & T_h(C,X) \wedge T_h(D,X) \\
T_h((\forall R.C),X) & =_{df} & T_h(C,Y) \leftarrow R(X,Y) \\
T_b(A,X) & =_{df} & A(X) \\
T_b((C \sqcap D),X) & =_{df} & T_b(C,X) \wedge T_b(D,X) \\
T_b((C \sqcup D),X) & =_{df} & T_b(C,X) \vee T_b(D,X) \\
T_b((\exists R.C),X) & =_{df} & R(X,Y) \wedge T_b(C,Y)
\end{array}
$$

Figure 2: Mapping from DL ontologies to DeLP strict rules

As DeLP is based on SLD-derivation of literals, simple translation of DL sentences to DeLP strict rules does not allow to infer negative information by *modus tollens*. For instance, "$C \sqsubseteq D$" (all C's are D's) is translated as "$D(X) \leftarrow C(X)$", DeLP is not able to derive "$\sim D(a)$" from "$\sim C(a)$". Thus given "$C_1 \sqcap C_2 \sqcap \ldots \sqcap C_{n-1} \sqcap C_n \sqsubseteq D$", instead of only including the strict rule "$D(X) \leftarrow C_1(X), C_2(X), \ldots, C_{n-1}(X), C_n(X)$" in its translation, we propose including all of its *transposes.*

Definition 7 (Transposes of a strict rule) *Let $r = H \leftarrow B_1, B_2, B_3, \ldots, B_{n-1}, B_n$ be a DeLP strict rule. The* set of transposes of rule *r, noted as "Trans(r)", is defined as:*

$$
Trans(r) = \left\{ \begin{array}{l}
H \leftarrow B_1, B_2, \ldots, B_{n-1}, B_n \\
\overline{B_1} \leftarrow \overline{H}, B_2, B_3, \ldots, B_{n-1}, B_n \\
\overline{B_2} \leftarrow \overline{H}, B_1, B_3, \ldots, B_{n-1}, B_n \\
\overline{B_3} \leftarrow \overline{H}, B_1, B_2, \ldots, B_{n-1}, B_n \\
\ldots \\
\overline{B_{n-1}} \leftarrow \overline{H}, B_1, B_2, B_3 \ldots, B_n \\
\overline{B_n} \leftarrow \overline{H}, B_1, B_2, \ldots, B_{n-1}
\end{array} \right\}
$$

We define the mapping from DL ontologies into DeLP strict rules as $\mathcal{T}_\Pi(T) = Trans(\mathcal{T}_\Pi^*(T))$. Notice that we do not consider transposition of defeasible rules as this issue is controversial (Brewka et al. 1997, Caminada 2008).

5 DeLP-based Ontologies

As mentioned previously, traditional DL reasoners are not capable of inferring information in the presence of inconsistent ontologies. In this section, we present a framework where inconsistent definitions for concepts in an ontology are expressed are as a set of *defeasible* inclusion and equality axioms. These axioms are to be considered to be tentative. Thus, in the presence of inconsistency, to determine the epistemic status of a sentence about some individual's membership to a concept description, a dialectical analysis will be carried out considering all arguments in favor and against its membership.

5.1 Knowledge representation: δ-ontologies

An ontology is defined as a set of classes and a set of individuals belonging to such classes. We redefine the notion of DL ontology for making it suitable for our approach.

Definition 8 (δ-Ontology) *Let C be an $\mathcal{L}_b$-class, D an $\mathcal{L}_h$-class, A, B $\mathcal{L}_{hb}$-classes, P, Q properties, a, b individuals. Let T be a set of inclusion and equality sentences in $\mathcal{L}_{DL}$ of the form $C \sqsubseteq D$, $A \equiv B$, $\top \sqsubseteq \forall P.D$, $\top \sqsubseteq \forall P^-.D$, $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^-$, o $P^+ \sqsubseteq P$ such that T can be partitioned into two disjoint sets T_S and T_D. Let A be a set of assertions disjoint with T of the form $a : D$ or $\langle a, b\rangle : P$. A* δ-ontology *Σ is a tuple (T_S, T_D, A). The set T_S is called the* strict terminology *(or Sbox), T_D the* defeasible terminology *(or Dbox) and A the* assertional box *(or Abox).*

Example 2 *Let $\Sigma_2 = (T_S, T_D, A)$ be a δ-ontology, where:*[2] *$T_S = \{(c \sqcup p \sqsubseteq b), (p \sqsubseteq \neg f)\}$; $T_D = \{(b \sqsubseteq f), (c \sqsubseteq \neg f), (c \sqcap s \sqsubseteq f), (f \sqsubseteq nit)\}$, and $A = \{(ti : c), (tw : p), (ti : s)\}$. The Sbox T_S says both that chickens and penguins are birds, and that penguins do not fly. The Dbox T_D expresses that birds usually fly, chickens typically do not fly unless they are scared, and that flying animals normally nest in trees. The Abox A establishes that Tina is a chicken, Tweety is a penguin and Tina is scared.*

We will see how to assign semantics to δ-ontologies as DeLP programs. The Sbox will be interpreted as a set of strict rules, the Abox as set of facts and the Dbox as a set of defeasible rules.

5.2 Semantic interpretation of δ-ontologies as DeLP programs

The traditional approach to reasoning in DLs is based on a model-theoretic semantics. As DLs are a subset of first-order logic (FOL), entailment has an *explosive* effect in the presence of inconsistent ontologies. In this work, we propose an argumentative approach to reasoning with inconsistent ontologies. Thus a δ-ontology will be interpreted as a DeLP program. As required by the DeLP framework, we are assuming that under a traditional interpretation the set $T_S \cup A$ has a model (see Section 5.3).

Definition 9 (Interpretation of a δ-ontology) *Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology. The* interpretation *of Σ is a DeLP program $\mathcal{P} = (\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$.*

Example 3 (Continues Ex. 2) *Consider again the δ-ontology Σ_2. Σ_2 is interpreted as the DeLP*

[2]This example appears in (García & Simari 2004) in a different context.

program $\mathcal{P}_2 = (\Pi, \Delta)$*, where:* $\Pi = \{(b(X) \leftarrow c(X)), (\sim c(X) \leftarrow \sim b(X)), (b(X) \leftarrow p(X)), (\sim p(X) \leftarrow \sim b(X)), (\sim f(X) \leftarrow p(X)), (\sim p(X) \leftarrow f(X)), c(ti), p(tw), s(ti)\}$*, and* $\Delta = \{ (f(X) \prec b(X)), (\sim f(X) \prec c(X)), (f(X) \prec c(X), s(X)), (nit(X) \prec f(X))\}$.

5.3 Inference tasks in δ-ontologies

In the DL approach to reasoning with ontologies in the Semantic Web, once a knowledge engineer has designed the terminology and used the DL reasoning service for checking that all of the terminology's concepts are satisfiable, the Abox can be filled with assertions about individuals. In order to keep consistency within an argument as required by Def. 2, we must enforce some internal coherence between the Abox and the Tbox. Formally:

Definition 10 (Internal coherence in Aboxes. Consistency of Aboxes w.r.t. Sboxes) *Let* $\Sigma = (T_S, T_D, A)$ *be a δ-ontology. Let C be a class name, a, b individuals. The Abox A is* internally coherent *iff there are no pair of assertions $a : C$ and $a : \neg C$. The Abox A is* consistent w.r.t. *the terminology T_S iff it is not possible to derive two literals $C(a)$ and $\sim C(a)$ from $\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A)$.*

Example 4 *Let* $\Sigma_4 = (T_S, \emptyset, A)$ *be a δ-ontology such that* $T_S = \{(C \sqsubseteq D), (D \sqsubseteq \neg F)\}$ *and* $A = \{(a : C), (a : F)\}$. Σ_4 *is expressed as* $\mathcal{P}_4 = (\Pi, \emptyset)$*, where* $\Pi = \mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A) = \{C(a), F(a), (D(X) \leftarrow C(X)), (\sim C(X) \leftarrow \sim D(X)), (\sim F(X) \leftarrow D(X)), (\sim D(X) \leftarrow F(X))\}$*, from which is possible to strictly derive $F(a)$ and $\sim F(a)$. Therefore A is not coherent w.r.t. T_S.*

5.3.1 Instance checking

In the traditional DL setting, *instance checking* refers to determining whether the assertions in the Abox entail that a particular individual is an instance of a given concept description (Baader et al. 2003). We propose a set of definitions to capture this notion in the context of δ-ontologies.

Definition 11 (Potential, justified and strict membership of an individual to a class) *Let* $\Sigma = (T_S, T_D, A)$ *be a δ-ontology. Let C be a class name, a an individual, let* $\mathcal{P} = (\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$. *(a) The* individual a potentially belongs to class C *(noted as "C^a_p") iff there exists an argument $\langle \mathcal{A}, C(a)\rangle$ w.r.t. $\mathcal{P}$; (b) the* individual a justifiedly belongs to class C *(noted as "C^a_j") iff there exists a warranted argument $\langle \mathcal{A}, C(a)\rangle$ w.r.t. $\mathcal{P}$, and, (c) the* individual a strictly belongs to class C *(noted as "C^a_s") iff there exists an argument $\langle \emptyset, C(a)\rangle$ w.r.t. $\mathcal{P}$.*

Property 1 C^a_s *implies* C^a_j*, and* C^a_j *implies* C^a_p.

Proof: The former holds because in DeLP empty arguments (*i.e.*, literals derived exclusively from strict rules) have no defeaters and they are thus warranted. The latter trivially holds because warranted arguments are arguments.

We now extend the notion of membership to arbitrary concept expressions.

Definition 12 (Potential, justified and strict membership of an individual to a class (extended version)) *Let* $\Sigma = (T_S, T_D, A)$ *be a δ-ontology. Let C, D class names in Σ, a, b individuals in Σ, R an atomic property in Σ,* $\mathcal{P} = (\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$. *Let E be a class name not present in Σ. The* potential (justified resp.) membership *of a to a complex concept is defined as:*

- $\neg C$: $(\neg C)^a_p$ *($(\neg C)^a_j$, resp.) iff there exists an argument (warranted argument, resp.) $\langle \mathcal{A}, \sim C(a)\rangle$ w.r.t. $\mathcal{P}$.*
- $C \sqcap D$: *Let* $\mathcal{P}_2 = (\Pi, \Delta \cup \mathcal{T}_\Delta(C \sqcap D \sqsubseteq E))$. $(C \sqcap D)^a_p$ *($(C \sqcap D)^a_j$, resp.) iff there exists an argument (warranted argument, resp.) $\langle \mathcal{A}, E(a)\rangle$ w.r.t. $\mathcal{P}_2$.*
- $C \sqcup D$: *Let* $\mathcal{P}_2 = (\Pi, \Delta \cup \mathcal{T}_\Delta(C \sqcup D \sqsubseteq E))$. $(C \sqcup D)^a_p$ *($(C \sqcup D)^a_j$, resp.) iff there exists an argument (warranted argument, resp.) $\langle \mathcal{A}, E(a)\rangle$ w.r.t. $\mathcal{P}_2$.*
- $\exists R.C$: *Let* $\mathcal{P}_2 = (\Pi, \Delta \cup \mathcal{T}_\Delta(\exists R.C \sqsubseteq E))$. $(\exists R.C)^a_p$ *($(\exists R.C)^a_j$, resp.) iff there exists an argument (warranted argument, resp.) $\langle \mathcal{A}, E(a)\rangle$ w.r.t. $\mathcal{P}_2$.*

The strict membership *of a to a complex concept is defined as:*

- $\neg C$: $(\neg C)^a_s$ *iff there exists an argument $\langle \emptyset, \sim C(a)\rangle$ w.r.t. $\mathcal{P}$.*
- $C \sqcap D$: *Let* $\mathcal{P}_2 = (\Pi \cup \mathcal{T}_\Pi(C \sqcap D \sqsubseteq E), \Delta)$. $(C \sqcap D)^a_s$ *iff there exists an argument $\langle \emptyset, E(a)\rangle$ w.r.t. $\mathcal{P}_2$.*
- $C \sqcup D$: *Let* $\mathcal{P}_2 = (\Pi \cup \mathcal{T}_\Pi(C \sqcup D \sqsubseteq E), \Delta)$. $(C \sqcup D)^a_s$ *iff there exists an argument $\langle \emptyset, E(a)\rangle$ w.r.t. $\mathcal{P}_2$.*
- $\exists R.C$: *Let* $\mathcal{P}_2 = (\Pi \cup \mathcal{T}_\Pi(\exists R.C \sqsubseteq E), \Delta)$. $(\exists R.C)^a_s$ *iff there exists an argument $\langle \emptyset, E(a)\rangle$ w.r.t. $\mathcal{P}_2$.*

Property 2 *Let* $\Sigma = (T_S, T_D, A)$ *be a δ-ontology. It cannot be the case that individual a belongs justifiedly to concept C and $\neg C$ simultaneously.*

Proof: Suppose that both C^a_j and $(\neg C)^a_j$. Then it must be the case there exist two warranted arguments $\langle \mathcal{A}, C(a)\rangle$ and $\langle \mathcal{B}, \sim C(a)\rangle$ w.r.t. $(\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$. But this impossible as DeLP cannot warrant two complementary literals simultaneously (as proven by García & Simari (2004)).

Property 3 *Let* $\Sigma = (T_S, T_D, A)$ *be a δ-ontology. It cannot be the case that individual a belongs strictly to concept C and $\neg C$ simultaneously.*

Proof: If C^a_s and $(\neg C)^a_s$, then there must exist two arguments $\langle \emptyset, C(a)\rangle$ and $\langle \emptyset, \sim C(a)\rangle$ w.r.t. $(\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$. Then $\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A)$ is inconsistent (contradicting Def. 10).

5.3.2 Retrieval

When DL knowledge bases are considered, we would want to know all individuals that are instances of a certain concept. In the traditional DL setting, given an ontology (T, A) and a concept C, in the *retrieval problem* we are interested to know all of the individuals a such that $T \cup A \models a : C$ (Baader et al. 2003). We present naïve solutions to two related problems:

- *Open retrieval:* Given a δ-ontology $\Sigma = (T_S, T_D, A)$ and a class C, find all individuals which are instances of C. We solve this problem by finding all the individuals a such that there exists a warranted argument $\langle \mathcal{A}, C(a)\rangle$ w.r.t. DeLP program $(\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$.
- *Retrieval of all classes:* Given a δ-ontology $\Sigma = (T_S, T_D, A)$ and an individual a, find all named classes C such that a is an instance of C. We solve this problem by finding all the classes C such that there exists a warranted argument $\langle \mathcal{A}, C(a)\rangle$ w.r.t. DeLP program $(\mathcal{T}_\Pi(T_S) \cup \mathcal{T}_\Pi(A), \mathcal{T}_\Delta(T_D))$.

Property 4 *The running time of processes for "open retrieval" and for "retrieval of all classes" is finite.*

Proof: As a δ-ontology has a finite number of both named concepts and individual constants, the DeLP program obtained from it is finite. Cecchi et al. (2006) have shown that determining if there exists an argument for a literal is NP; besides, as the warrant procedure always builds a finite dialectical tree (García & Simari 2004), then proceses for "open retrieval" and for "retrieval of all classes" always terminate.

6 DeLP-based Integration of DL Ontologies

In this section, we introduce an application for ontology integration in the Semantic Web based on the framework presented above. *Ontology integration* is the problem of combining ontologies residing in different sources and to provide the user with an unified view of such ontologies (Calvanese et al. 2001). The problem of designing systems for ontology integration in the Semantic Web is particularly important because ontologies are to be developed independently from each other and, for this reason, they can be mutually inconsistent. One possible architecture for ontology integration systems is based on a global schema and a set of local sources. The local sources contain the actual data while the global schema provides a reconciled, unified view of the underlying sources. A basic service provided by ontology integration systems is that of answering queries posed in terms of the global schema.

Definition 13 (Ontology integration system) *An* ontology integration system *$\mathcal{I}$ is a triple $(\mathcal{G}, \mathcal{S}, \mathcal{M})$ where:*

- *$\mathcal{G}$ is a* global ontology *expressed as a δ-ontology over an alphabet $\mathcal{A}_\mathcal{G}$.*
- *$\mathcal{S}$ is a* set of n source ontologies *$\mathcal{S}_1, \dots, \mathcal{S}_n$ expressed as δ-ontologies over alphabets $\mathcal{A}_{\mathcal{S}_1}, \dots, \mathcal{A}_{\mathcal{S}_n}$, resp. Each alphabet $\mathcal{A}_{\mathcal{S}_i}$ includes a symbol for each element of the source $\mathcal{S}_i$, $i = 1, \dots, n$.*
- *$\mathcal{M}$ is a* set of n mappings *$\mathcal{M}_1, \dots, \mathcal{M}_n$ between $\mathcal{G}$ and $\mathcal{S}_1, \dots, \mathcal{S}_n$, resp. Each mapping $\mathcal{M}_i$ is constituted by a set of* assertions *of the form $q_{\mathcal{S}_i} \sqsubseteq q_\mathcal{G}$, where $q_\mathcal{G}$ y $q_{\mathcal{S}_i}$ are queries of the same arity defined over global ontology $\mathcal{G}$ and $\mathcal{S}_i$, $i = 1, \dots, n$, resp. Queries $q_\mathcal{G}$ are expressed over alphabet $\mathcal{A}_\mathcal{G}$ and queries $q_{\mathcal{S}_i}$ are expressed over alphabet $\mathcal{A}_{\mathcal{S}_i}$. The sets $\mathcal{M}_1, \dots, \mathcal{M}_n$ are called* bridge ontologies.

Next we show a case study in which several DL *local* ontologies and a DL *global* ontology are integrated. These ontologies will be interpreted as DeLP programs. Queries posed w.r.t. the global ontology are going to be interpreted as queries answered on the basis of such DeLP programs.

Example 5 *Consider the* global *δ-ontology $\mathcal{G} = (\emptyset, T_D^\mathcal{G}, \emptyset)$ presented in Fig. 3. The Dbox $T_D^\mathcal{G}$ expresses that computer geeks are usually not good in sports; expert swimmers are normally good at sports, and, if somebody is either capable of swimming both a race stroke and a rescue stroke or is a diver, then she is typically considered an expert swimmer.*

Defeasible terminology $T_D^\mathcal{G}$:

$geek \sqsubseteq \neg good$

$swimmer \sqsubseteq good$

$(\exists can\_swim.rescue\_stroke \sqcap \exists can\_swim.race\_stroke) \sqcup diver \sqsubseteq swimmer$

Figure 3: Global ontology $\mathcal{G} = (\emptyset, T_D^\mathcal{G}, \emptyset)$

Notice that the terminology $T_D^\mathcal{G}$ expresses a conflict w.r.t. concept "*good*". If we find that some individual in the Abox can be proven to be member of both concepts "*swimmer*" and "*geek*", then the Abox would be incoherent from a traditional DL point of view because that individual would belong both to "*good*" and to "$\neg$*good*, indicating that concept "*good*" should be empty and having one individual at the same time. We will show how this situation can be handled naturally in DeLP.

Example 6 (Continues Ex. 5) *In Fig. 4, we present two source ontologies: $\mathcal{S}_1$ about water activities, and $\mathcal{S}_2$ on computer programming. In source local ontology $\mathcal{S}_1 = (\emptyset, T_D^{\mathcal{S}_1}, A^{\mathcal{S}_1})$, Dbox $T_D^{\mathcal{S}_1}$ says that both free and scuba divers are divers, saturation divers are a particular class of scuba divers, and somebody capable of swimming some kind of stroke is usually a swimmer. Abox $A^{\mathcal{S}_1}$ establishes that John swims both crawl and side strokes, Paul is a saturation diver, and crawl and side are swimming strokes.*

In source local ontology $\mathcal{S}_2 = (T_S^{\mathcal{S}_2}, T_D^{\mathcal{S}_2}, A^{\mathcal{S}_2})$, Sbox $T_S^{\mathcal{S}_2}$ expresses that among programming languages, both logic programming and object-oriented languages can be found. Dbox $T_D^{\mathcal{S}_2}$ says that a programmer is usually somebody who can program in some programming language, and that someone who can read and write code in such a language can program unless she has failed the elementary programming course. Abox $A^{\mathcal{S}_2}$ establishes that Prolog is a logic programming language and that John can read and write Prolog code; that Java is an object-oriented language and that Mary can read and write Java code, and that Paul is capable of reading and writing Java code although he failed the elementary programming course.

Source ontology $\mathcal{S}_1 = (\emptyset, T_D^{\mathcal{S}_1}, A^{\mathcal{S}_1})$:

Defeasible terminology $T_D^{\mathcal{S}_1}$:

$free\_diver \sqcup scuba\_diver \sqsubseteq diver$

$saturation\_diver \sqsubseteq scuba\_diver$

$\exists swims.stroke \sqsubseteq swimmer$

Assertional box $A^{\mathcal{S}_1}$:

$crawl : stroke;$	$side : stroke$
$\langle john, crawl \rangle : swims;$	$\langle john, side \rangle : swims$
$paul : saturation\_diver$	

Source ontology $\mathcal{S}_2 = (T_S^{\mathcal{S}_2}, T_D^{\mathcal{S}_2}, A^{\mathcal{S}_2})$:

Strict terminology $T_S^{\mathcal{S}_2}$:

$lp\_lang \sqcup oop\_lang \sqsubseteq lang$

Defeasible terminology $T_D^{\mathcal{S}_2}$:

$\exists programs.lang \sqsubseteq programmer$

$\exists programs.lang \sqcap failed\_prog\_101 \sqsubseteq \neg programmer$

$reads \sqcap writes \sqsubseteq programs$

Assertional box $A^{\mathcal{S}_2}$:

$prolog : lp\_lang;$	$java : oop\_lang$
$\langle john, prolog \rangle : reads;$	$\langle john, prolog \rangle : writes$
$\langle mary, java \rangle : reads;$	$\langle mary, java \rangle : writes$
$\langle paul, java \rangle : reads;$	$\langle paul, java \rangle : writes$
$paul : failed\_prog\_101$	

Figure 4: Source ontologies $\mathcal{S}_1$ and $\mathcal{S}_2$

Notice how, in particular, source ontology $\mathcal{S}_2$ is inconsistent from a traditional point of view because the individual named Paul belongs at the same time to concepts "*programmer*" and "$\neg$*programmer*". Therefore, this ontology cannot be processed by traditional DL reasoners. We will show how can this be achieved in the framework of δ-ontologies.

As mentioned above, our goal is to answer queries about the membership of individuals to a certain concept of a global ontology using the data defined in source ontologies. The relationship of the global ontology with the local ontologies is achieved through *bridge ontologies.* A bridge ontology allows to map concepts and properties between two ontologies. Thus a concept in one ontology corresponds to a *view* of one or several concepts in some other ontology. In the examples we present, we consider bridge ontologies as *given*; for techniques on semi-automatic discovery of such mappings implemented as articulation rules, see (Mitra 2004). Moreover, we are going to assume *unique name assumption* w.r.t. references to individuals in Aboxes through our presentation.

Example 7 (Continues Ex. 6) *Consider again global ontology $\mathcal{G}$ and source ontologies $\mathcal{S}_1$ and $\mathcal{S}_2$. Definitions in $\mathcal{G}$ with those in $\mathcal{S}_1$ and $\mathcal{S}_2$ are articulated by bridge ontologies $\mathcal{M}_1$ and $\mathcal{M}_2$, resp. (see Fig. 5). For clarity, in bridge ontologies we qualify concept and property names with their defining ontology name.*

Bridge ontology $\mathcal{M}_1$ expresses that concept "swims" in $\mathcal{S}_1$ corresponds to concept "can_swim" in $\mathcal{G}$; concept "diver" in $\mathcal{S}_1$ refers to "diver" in $\mathcal{G}$; concept (anonimously defined by the one-of *construct) composed of individual "side" in $\mathcal{S}_1$ is mapped to the concept "rescue_stroke" in $\mathcal{G}$, and, the concept composed by individual "crawl" in $\mathcal{S}_1$ is mapped to "race_stroke" in $\mathcal{G}$. Bridge ontology $\mathcal{M}_2$ indicates that concept "programmer" in $\mathcal{S}_2$ corresponds to concept "geek" defined in $\mathcal{G}$.*

Bridge ontology $\mathcal{M}_1$ between $\mathcal{G}$ and $\mathcal{S}_1$:

$\mathcal{S}_1 : swims \sqsubseteq \mathcal{G} : can\_swim$
$\mathcal{S}_1 : diver \sqsubseteq \mathcal{G} : diver$
$\mathcal{S}_1 : \{side\} \sqsubseteq \mathcal{G} : rescue\_stroke$
$\mathcal{S}_1 : \{crawl\} \sqsubseteq \mathcal{G} : race\_stroke$

Bridge ontology $\mathcal{M}_2$ between $\mathcal{G}$ and $\mathcal{S}_2$:

$\mathcal{S}_2 : programmer \sqsubseteq \mathcal{G} : geek$

Figure 5: Bridge ontologies $\mathcal{M}_1$ and $\mathcal{M}_2$

As discussed above, in the *global-as-view* approach to ontology integration, queries are posed w.r.t. a global ontology which is used as a means to access data found in local source ontologies. Next we show how to extend the task of instance checking for individual membership to concepts defined in a global ontology in the context of an ontology integration system. We will show how an ontology integration system can be regarded as a DeLP program and queries to the ontology integration system can be interpreted as queries w.r.t. such DeLP program.

Definition 14 (Interpretation of an ontology integration system) *Let $\mathcal{I} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ be an ontology integration system such that $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ y $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$, where:*

- $\mathcal{G} = (T_S^{\mathcal{G}}, T_D^{\mathcal{G}}, A^{\mathcal{G}})$;
- $\mathcal{S}_i = (T_S^{\mathcal{S}_i}, T_D^{\mathcal{S}_i}, A_i^{\mathcal{S}_i})$, *with* $i = 1, \ldots, n$, *and,*
- $\mathcal{M}_i = (T_S^{\mathcal{M}_i}, T_D^{\mathcal{M}_i})$, *with* $i = 1, \ldots, n$.

The system $\mathcal{I}$ is interpreted as the DeLP program $\mathcal{I}_{DeLP} = (\Pi, \Delta)$:

$$\begin{aligned}\Pi &= \Big(\mathcal{T}_\Pi(T_S^{\mathcal{G}}) \Big) \cup \Big(\mathcal{T}_\Pi(A^{\mathcal{G}}) \Big) \cup \Big(\textstyle\bigcup_{i=1}^{n} \mathcal{T}_\Pi(T_S^{\mathcal{S}_i}) \Big) \cup \\ &\quad \Big(\textstyle\bigcup_{i=1}^{n} \mathcal{T}_\Pi(T_S^{\mathcal{M}_i}) \Big); \\ \Delta &= \Big(\mathcal{T}_\Delta(T_D^{\mathcal{G}}) \Big) \cup \Big(\textstyle\bigcup_{i=1}^{n} \mathcal{T}_\Delta(T_D^{\mathcal{S}_i}) \Big) \cup \\ &\quad \Big(\textstyle\bigcup_{i=1}^{n} \mathcal{T}_\Delta(T_D^{\mathcal{M}_i}) \Big).\end{aligned}$$

Example 8 (Continues Ex. 7) *The interpretation as DeLP programs of global ontology $\mathcal{G}$; sources $\mathcal{S}_1$ and $\mathcal{S}_2$, and bridges $\mathcal{M}_1$ and $\mathcal{M}_2$ are shown in Figs. 6, 7 and 8, resp. Global ontology $\mathcal{G}$ is interpreted as the DeLP program $\mathcal{P}_{\mathcal{G}} = (\emptyset, \Delta_{\mathcal{G}})$; source ontology $\mathcal{S}_1$, as $\mathcal{P}_1 = (\Pi_1, \Delta_1)$; source ontology $\mathcal{S}_2$, as $\mathcal{P}_2 = (\Pi_2, \Delta_2)$; bridge ontology $\mathcal{M}_1$, as the set of defeasible rules $\Delta_{\mathcal{M}_1}$, and, bridge ontology $\mathcal{M}_2$, as $\Delta_{\mathcal{M}_2}$.*

Thus, the interpretation of $\mathcal{I}$ is the DeLP program $\mathcal{P}_{\mathcal{I}} = (\Pi, \Delta)$ where $\Pi = \Pi_1 \cup \Pi_2$, and $\Delta = \Delta_{\mathcal{G}} \cup \Delta_1 \cup \Delta_2 \cup \Delta_{\mathcal{M}_1} \cup \Delta_{\mathcal{M}_2}$.

DeLP program $\mathcal{P}_{\mathcal{G}} = (\emptyset, \Delta_{\mathcal{G}})$ obtained from $\mathcal{G}$:

Defeasible rules $\Delta_{\mathcal{G}}$:
$\sim good(X) \prec geek(X)$
$good(X) \prec swimmer(X)$
$swimmer(X) \prec$
$\quad can\_swim(X, Y), rescue\_stroke(Y),$
$\quad can\_swim(X, Z), race\_stroke(Z)$
$swimmer(X) \prec diver(X)$

Figure 6: DeLP program $\mathcal{P}_{\mathcal{G}} = (\emptyset, \Delta_{\mathcal{G}})$ obtained from global ontology $\mathcal{G}$

DeLP program $\mathcal{P}_1 = (\Pi_1, \Delta_1)$ obtained from $\mathcal{S}_1$:

Facts Π_1:
$stroke(crawl)$; $\quad stroke(side)$
$swims(john, crawl)$; $\quad swims(john, side)$
$saturation\_diver(paul)$

Defeasible rules Δ_1:
$diver(X) \prec free\_diver(X)$
$diver(X) \prec scuba\_diver(X)$
$scuba\_diver(X) \prec saturation\_diver(X)$
$swimmer(X) \prec swims(X, Y), stroke(Y)$

DeLP program $\mathcal{P}_2 = (\Pi_2, \Delta_2)$ obtained from $\mathcal{S}_2$:

Facts and strict rules Π_2:
$lp\_lang(prolog)$; $\quad oop\_lang(java)$
$reads(john, prolog)$; $\quad writes(john, prolog)$
$reads(mary, java)$; $\quad writes(mary, java)$
$reads(paul, java)$; $\quad writes(paul, java)$
$failed\_prog\_101(paul)$
$lang(X) \leftarrow lp\_lang(X)$
$\sim lp\_lang(X) \leftarrow \sim lang(X)$
$lang(X) \leftarrow oop\_lang(X)$
$\sim oop\_lang(X) \leftarrow \sim lang(X)$

Defeasible rules Δ_2:
$programmer(X) \prec programs(X, Y), lang(Y)$
$\sim programmer(X) \prec$
$\quad programs(X, Y), lang(Y), failed\_prog\_101(X)$
$programs(X, Y) \prec reads(X, Y), writes(X, Y)$

Figure 7: DeLP programs $\mathcal{P}_1$ and $\mathcal{P}_2$ obtained from source ontologies $\mathcal{S}_1$ and $\mathcal{S}_2$, resp.

Possible inferences in the integrated ontology $\mathcal{I}_{DeLP}$ are modeled by means of a dialectical analysis in the DeLP program that is obtained when each DL sentence of the ontology is mapped into DeLP clauses. Thus warranted arguments will be the valid consequences that will be obtained from the original ontology, provided the strict information in $\mathcal{I}_{DeLP}$ is consistent. Formally:

Definition 15 (Potential, justified and strict membership of individuals to concepts in ontology integration systems) *Let $\mathcal{I} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ be an ontology integration system. Let a be an individual name, and c a concept name defined in $\mathcal{G}$. Individual a* belongs potentially *to concept C iff there exists an argument $\mathcal{A}$ for the literal $C(a)$ w.r.t. DeLP program $\mathcal{I}_{DeLP}$. Individual a* belongs justifiedly *to concept C iff there exists a warranted argument $\mathcal{A}$ for the literal*

Bridge rules $\Delta_{\mathcal{M}_1}$ between ontologies $\mathcal{G}$ and $\mathcal{S}_1$:

$\mathcal{G} : can\_swim(X, Y) \prec \mathcal{S}_1 : swims(X, Y)$
$\mathcal{G} : diver(X) \prec \mathcal{S}_1 : diver(X)$
$\mathcal{G} : rescue\_stroke(X) \prec \mathcal{S}_1 : stroke(X), X = side$
$\mathcal{G} : race\_stroke(X) \prec \mathcal{S}_1 : stroke(X), X = crawl$

Bridge rules $\Delta_{\mathcal{M}_2}$ between ontologies $\mathcal{G}$ and $\mathcal{S}_2$:

$\mathcal{G} : geek(X) \prec \mathcal{S}_2 : programmer(X)$

Figure 8: Bridge ontologies expressed as defeasible rules

$C(a)$ w.r.t. DeLP program $\mathcal{I}_{DeLP}$. Individual a belongs strictly *to concept C iff there exists an empty argument for the literal $C(a)$ w.r.t. DeLP program $\mathcal{I}_{DeLP}$.*

Next we will show some of the arguments that can be built from the integrated ontology system. In the rest of the presentation, we are assuming generalized specificity (Simari & Loui 1992) as the criterion for argument comparison.

Example 9 (Continues Ex. 8) *Consider again DeLP program $\mathcal{I}_{DeLP}$, we are interested in determining the justified membership of individuals John, Mary and Paul to concepts "good" and/or "$\neg$good". According to Def. 15, it is necessary to determine if there exist warranted arguments for literals good(john), good(mary) and good(paul), resp. Notice that answers to queries cannot be ambiguous as it is not possible to warrant complementary literals in DeLP at the same time (see Section 7). We will see that as John is both a geek and a swimmer, it will not be possible to determine if he is or not good at sports. In spite of this result, we will also see that as Mary is a Java programmer, she will not be regarded as good at sports. In the case of Paul, as he is a diver and, although he programs in Java but failed the elementary programming course, he will not be considered a programmer and thus, he will be regarded as good at sports.*

First, we will consider the dialectical analysis for the query "good(john)". There are reasons to assert that John belongs potentially to concept "good". Formally, there exists an argument $\langle \mathcal{A}_1, good(john)\rangle$ where:

$$\mathcal{A}_1 = \left\{ \begin{array}{l} (good(john) \mathrel{-\!\!\prec} swimmer(john)),\\ (swimmer(john) \mathrel{-\!\!\prec}\\ \quad can\_swim(john, side), rescue\_stroke(side),\\ \quad can\_swim(john, crawl), race\_stroke(crawl)),\\ (\mathcal{G} : race\_stroke(crawl) \mathrel{-\!\!\prec}\\ \quad \mathcal{S}_1 : stroke(crawl), crawl = crawl),\\ (\mathcal{G} : can\_swim(john, crawl) \mathrel{-\!\!\prec}\\ \quad \mathcal{S}_1 : swims(john, crawl)),\\ (\mathcal{G} : rescue\_stroke(side) \mathrel{-\!\!\prec}\\ \quad \mathcal{S}_1 : stroke(side), side = side)\\ (\mathcal{G} : can\_swim(john, side) \mathrel{-\!\!\prec}\\ \quad \mathcal{S}_1 : swims(john, side)) \end{array} \right\}$$

However, John belongs potentially to concept "$\neg$good" because he is a computer geek. Formally, there is an argument $\langle \mathcal{A}_2, \sim good(john)\rangle$ that defeats argument $\mathcal{A}_1$, where:

$$\mathcal{A}_2 = \left\{ \begin{array}{l} (\sim good(john) \mathrel{-\!\!\prec} geek(john)),\\ (\mathcal{G} : geek(john) \mathrel{-\!\!\prec} \mathcal{S}_2 : programmer(john)),\\ (programmer(john) \mathrel{-\!\!\prec}\\ \quad programs(john, prolog), lang(prolog)),\\ (programs(john, prolog) \mathrel{-\!\!\prec}\\ \quad reads(john, prolog), writes(john, prolog)) \end{array} \right\}.$$

Thus, in the dialectical tree for the query "good(john)", defeated argument $\mathcal{A}_1$ appears labeled as a D-node while victorious argument $\mathcal{A}_2$ appears marked as a U-node. (see Fig. 9.(a)). On the other hand, when we consider the membership of John to concept "$\neg$good", we discover that argument $\mathcal{A}_2$ supporting this conclusion is defeated by argument $\mathcal{A}_1$ (see Fig. 9.(b)). Therefore, the answer to query "good(john)" is Undecided.

Second, we consider the dialectical analysis for determining if Mary belongs to concept "good". Mary belongs justifiedly to concept "$\neg$good" as the answer to query "good(mary)" is No *because there is a warranted argument $\langle \mathcal{B}, \sim good(mary)\rangle$ (see Fig. 9.(c)), where:*

$$\mathcal{B} = \left\{ \begin{array}{l} (\sim good(mary) \mathrel{-\!\!\prec} geek(mary)),\\ (\mathcal{G} : geek(mary) \mathrel{-\!\!\prec} \mathcal{S}_2 : programmer(mary)),\\ (programmer(mary) \mathrel{-\!\!\prec}\\ \quad programs(mary, java), lang(java)),\\ (lang(java) \mathrel{-\!\!\prec} oop\_lang(java)),\\ (programs(mary, java) \mathrel{-\!\!\prec}\\ \quad reads(mary, java), writes(mary, java)) \end{array} \right\}$$

Third, we will see why Paul belongs justifiedly to concept "good". Let us consider the dialectical tree for the literal "good(paul)". There is an argument $\langle \mathcal{C}_1, good(paul)\rangle$, based on the defeasible information that expresses that Paul is an expert swimmer (because he is a saturation diver):

$$\mathcal{C}_1 = \left\{ \begin{array}{l} (good(paul) \mathrel{-\!\!\prec} swimmer(paul)),\\ (swimmer(paul) \mathrel{-\!\!\prec} \mathcal{G} : diver(paul)),\\ (\mathcal{G} : diver(paul) \mathrel{-\!\!\prec} \mathcal{S}_1 : diver(paul)),\\ (\mathcal{S}_1 : diver(paul) \mathrel{-\!\!\prec} scuba\_diver(paul)),\\ (scuba\_diver(paul) \mathrel{-\!\!\prec} saturation\_diver(paul)) \end{array} \right\}$$

But argument $\mathcal{C}_1$ is attacked by an argument $\langle \mathcal{C}_2, \sim good(paul)\rangle$, where:

$$\mathcal{C}_2 = \left\{ \begin{array}{l} (\sim good(paul) \mathrel{-\!\!\prec} geek(paul)),\\ (\mathcal{G} : geek(paul) \mathrel{-\!\!\prec} \mathcal{S}_2 : programmer(paul)),\\ (programmer(paul) \mathrel{-\!\!\prec}\\ \quad programs(paul, java), lang(java)),\\ (programs(paul, java) \mathrel{-\!\!\prec}\\ \quad reads(paul, java), writes(paul, java)) \end{array} \right\}$$

Nevertheless, Paul also belongs potentially to concept "$\mathcal{S}_2$: $\neg$programmer" (because he failed the elementary programming course), as an argument $\langle \mathcal{C}_3, \mathcal{S}_2 : \sim programmer(paul)\rangle$ can be found, where:

$$\mathcal{C}_3 = \left\{ \begin{array}{l} (\sim programmer(paul) \mathrel{-\!\!\prec}\\ \quad programs(paul, java), lang(java),\\ \quad failed\_prog\_101(paul)),\\ (programs(paul, java) \mathrel{-\!\!\prec}\\ \quad reads(paul, java), writes(paul, java)) \end{array} \right\}$$

Thus, the dialectical tree for the query "good(paul)" has three nodes (see Fig. 9.(d)). Respect the query "$\sim$good(paul)" for determining if Paul belongs justifiedly to concept "$\neg$good", argument $\mathcal{C}_2$ is defeated by argument $\mathcal{C}_1$ (see Fig. 9.(e)). Therefore, the answer to the query "good(paul)" is Yes, *and we conclude that Paul belongs justifiedly to concept "good".*

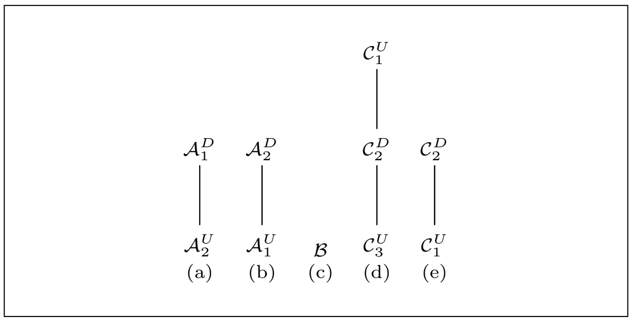


Figure 9: Dialectical trees for queries *good(john)*, *good(mary)* and *good(paul)*

7 Evaluation of the Proposal

In order to evaluate our approach, we propose using the framework presented by Huang et al. (2005) for reasoning with inconsistent ontologies with a non-standard inference relation. With classical reasoning, a query ϕ given an ontology Σ can be expressed as an evaluation of the consequence relation $\Sigma \models \phi$; there are two answers to a query: either "yes" ($\Sigma \models \phi$) or "no" ($\Sigma \not\models \phi$). For reasoning with inconsistent ontologies with a non-standard inference relation, Huang et al. (2005) propose using an alternative classification to distinguish answers to queries:

Definition 16 (Epistemic status of an answer (Huang et al. 2005)) *Given an ontology Σ and a query ϕ, the answer to ϕ will have one of the four epistemic states:*

1. *Over-determined:* $\Sigma \mid\!\approx \phi$ *and* $\Sigma \mid\!\approx \neg\phi$*;*
2. *Accepted:* $\Sigma \mid\!\approx \phi$ *and* $\Sigma \not\mid\!\approx \neg\phi$*;*
3. *Rejected:* $\Sigma \not\mid\!\approx \phi$ *and* $\Sigma \mid\!\approx \neg\phi$*;*
4. *Undetermined:* $\Sigma \not\mid\!\approx \phi$ *and* $\Sigma \not\mid\!\approx \neg\phi$*.*

If we regard the relation $\mid\!\approx$ as "justified membership" of instances to concepts (see Def. 11), $\Sigma \mid\!\approx \phi$ corresponds to a YES answer to query ϕ w.r.t. program $\mathcal{T}(\Sigma)$.

Property 5 *Let $\mid\!\approx$ be the "justified membership" of instances to concepts relationship. Let Σ be a δ-ontology. The answer to a query ϕ is never over-determined.*

Proof: Suppose by the contrary that the answer to ϕ is over-determined. Then it must be the case that there exist two warranted arguments $\langle \mathcal{A}, \phi \rangle$ and $\langle \mathcal{A}, \sim\!\phi \rangle$ w.r.t. $\mathcal{T}(\Sigma)$. This cannot be the case in DeLP as shown by García & Simari (2004).

Notice that, as required by traditional DL reasoning, DeLP does not adopt the closed-world assumption. That is, not being able to prove Q in DeLP does not imply that $\sim\!Q$ will be assumed. On the contrary, such an answer will be the result of a dialectical analysis that will take into account all of the reasons for Q and $\sim\!Q$.

Definition 17 (Soundness (Huang et al. 2005)) *An inconsistency reasoner $\mid\!\approx$ is* sound *if the formulas that follow from an inconsistent theory Σ follow from a consistent subtheory of Σ using classical reasoning.*

Property 6 $\mid\!\approx$ *is a sound inconsistency reasoner.*

Proof: Let $\Sigma = (T_S, T_D, A)$ be a δ-ontology. If $\Sigma \mid\!\approx \phi$ then there exists a warranted argument $\langle \mathcal{A}, \phi \rangle$ w.r.t. $\mathcal{T}(\Sigma) = (\Pi_S \cup \Pi_A, \Delta)$, where $\Pi_S = \mathcal{T}_\Pi(T_S)$, $\Pi_A = \mathcal{T}_\Pi(A)$ and $\Delta = \mathcal{T}_\Delta(T_D)$. The set $\Pi_S \cup \Pi_A \cup \mathcal{A}$ (see Def. 2) is consistent and as $\mathcal{T}$ is a transformation that preserves semantics (Grosof et al. 2003), there must exists a subset $\Sigma' \subseteq \Sigma$ such that $\mathcal{T}(\Sigma') = \Pi_S \cup \Pi_A \cup \mathcal{A}$.

Definition 18 (Consistency (Huang et al. 2005)) *An inconsistency reasoner $\mid\!\approx$ is* consistent *iff $\Sigma \mid\!\approx \phi \Rightarrow \Sigma \not\mid\!\approx \neg\phi$.*

Property 7 $\mid\!\approx$ *is a consistent inconsistency reasoner.*

Proof: Corollary of Prop. 5.

Definition 19 (Meaningfulness (Huang et al. 2005)) *An answer given by an inconsistency reasoner is meaningful iff it is consistent and sound. An inconsistency reasoner is said to be meaningful iff all of its answers are meaningful.*

Property 8 $\mid\!\approx$ *is a meaningful inconsistency reasoner.*

Proof: Trivial from Props. 6 and 7.

Implementation issues

As mentioned in Section 4, we base our translation function from DL to DeLP on the work reported by Grosof et al. (2003). In this respect, Volz (2004) shows that the fragment of DL expressible in logic programming (referred to as *DLP*) is sufficient to express most available Web ontologies. Volz has analyzed the largest currently available collection of Web ontologies and checked which fragment of those ontologies can be expressed in DLP; he claims both that DLP languages suffice to express 77%–87% of the analyzed ontologies and can express 93%–99% of the individual axioms in the analyzed ontologies.

As performing defeasible argumentation is a computationally complex task, an abstract machine called JAM (Justification Abstract Machine) has been specially developed for an efficient implementation of DeLP (García & Simari 2004). JAM provides an argument-based extension of the traditional WAM (Warren's Abstract Machine) for Prolog. A full-fledged implementation of DeLP is available online[3], including facilities for visualizing arguments and dialectical trees.

8 Related Work

Grosof et al. (2003) show how to interoperate, semantically and inferentially, between the leading Semantic Web approaches to rules (RuleML Logic Programs) and ontologies (OWL DL) by analyzing their expressive intersection. They define a new intermediate knowledge representation called Description Logic Programs (DLP), and the closely related Description Horn Logic (DHL) which is an expressive fragment of FOL. They show how to perform the translation of premises and inferences from the DLP fragment of DL to logic programming. Part of our approach is based on Grosof's work as the algorithm for translating DL ontologies into DeLP is based on it. However, as Grosof et al. (2003) use standard Prolog rules, they are not able to deal with inconsistent DL knowledge bases as our proposal does.

Heymans & Vermeir (2002) extend the DL $\mathcal{SHOQ}(D)$ with a preference order on the axioms. With this strict partial order certain axioms can be overruled, if defeated with more preferred ones. They also impose a preferred model semantics, introducing nonmonotonicity into $\mathcal{SHOQ}(D)$. Similarly to Heymans & Vermeir (2002) we allow to perform inferences inferences from inconsistent ontologies by considering subsets (arguments) of the original ontology. Heymans & Vermeir (2002) also impose a hard-coded comparison criterion on DL axioms. In our work, the system, and not the programmer, decides which DL axioms are to be preferred as we use specificity as argument comparison criterion. We think that our approach can be considered more declarative in this respect. In particular the comparison criterion in DeLP is modular, so that rule comparison could also be adopted (García & Simari 2004).

Eiter et al. (2004) propose a combination of logic programming under the answer set semantics with the DLs $\mathcal{SHIF}(D)$ and $\mathcal{SHOIN}(D)$. This combination allows for building rules on top of ontologies. In contrast to our approach, they keep separated rules and ontologies and handle exceptions by codifying them explicitly in programs under answer set semantics.

Huang et al. (2005) use paraconsistent logics to reason with inconsistent ontologies. They use a selection function to determine which consistent subsets of an inconsistent ontology should be considered in the reasoning process. In our approach given an inconsistent ontology Σ, we consider the set of warranted arguments from $\mathcal{T}(\Sigma)$ as the valid consequences.

Williams & Hunter (2007) use argumentation to reason with possibly inconsistent rules on top of DL ontologies. In contrast, we translate possible inconsistent DL ontologies to DeLP to reason with them within DeLP. Laera et al. (2006) propose an approach for supporting the creation and exchange of different arguments, that support or reject possible correspondences between ontologies in the context of a

[3] See `http://lidia.cs.uns.edu.ar/DeLP`

multi-agent system. In our work we assume correspondences between ontologies as given.

Antoniou & Bikakis (2007) propose a rule-based approach to defeasible reasoning based on a translation to logic programming with declarative semantics that can reason with rules, RDF(S) and parts of OWL ontologies. In contrast with our approach, argumentation is not used explicitly for detecting inconsistent ontologies. Instead, they translate OWL statements to Prolog to reason with Defeasible Logic.

9 Conclusions and Future Work

We have presented a framework for reasoning with inconsistent Description Logics (DL) ontologies. Our proposal involves expressing DL ontologies as a Defeasible Logic Program (DeLP) by means of a translation function $\mathcal{T}$. Given a query ϕ posed w.r.t. an inconsistent ontology Σ, a dialectical analysis is performed on a DeLP program $\mathcal{T}(\Sigma)$ where all arguments in favor and against ϕ's acceptance are taken into account. We have also presented an application to ontology integration based on the global-as-view approach to ontology integration where queries respect a global ontology are posed while data is extracted from local ontologies that could be inconsistent.

Several issues need to be solved and part of our efforts are focused on that matter. For instance, as DeLP does not support disjunctions in the head of rules, our approach is not able to deal with DL axioms that require the construction of such rules, a possible extension to this work could be in that direction. Other issue that needs to be addressed is given by the mapping of DL equality axioms into DeLP rules. Currently a DL axiom of the form "$C \equiv D$" generates two rules of the form "$C(X) \prec D(X)$" and "$D(X) \prec C(X)$". This situation could clearly produce loops during argument construction when solving queries in actual DeLP programs. Nevertheless the examples considered in this work model an important part of ontologies where this situation does not happen, a possible solution involves separating equality axioms from simple inclusion axioms and keeping track of their instantiations into ground rules to avoid such looping situations.

References

Antoniou, G. & Bikakis, A. (2007), 'DR-Prolog: A System for Defeasible Reasoning with Rules and Ontologies on the Semantic Web', *IEEE Trans. on Knowledge and Data Eng.* **19**(2), 233–245.

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

Berners-Lee, T., Hendler, J. & Lassila, O. (2001), 'The Semantic Web', *Scientific American* **284**(5), 34–43.

Brewka, G., Dix, J. & Konolige, K. (1997), *Non monotonic reasoning. An overview*, CSLI Publications, Stanford, USA.

Calvanese, D., Giacomo, G. D. & Lenzerini, M. (2001), A Framework for Ontology Integration, *in* 'Proceedings of the 1st Semantic Web Working Symposium (SWWS 2001)', pp. 303–316.

Caminada, M. (2008), 'On the Issue of Contraposition of Defeasible Rules', *COMMA 2008 (to appear)* .

Cecchi, L. A., Fillottrani, P. R. & Simari, G. R. (2006), On Complexity of DeLP through Game Semantics, *in* J. Dix & A. Hunter, eds, '11th. Intl. Workshop on Nonmonotonic Reasoning', pp. 386–394.

Chesñevar, C. I., Maguitman, A. & Loui, R. (2000), 'Logical Models of Argument', *ACM Computing Surveys* **32**(4), 337–383.

Eiter, T., Lukasiewicz, T., Schindlauer, R. & Tompits, H. (2004), 'Combining Answer Set Programming with Description Logics for the Semantic Web', *KR 2004* pp. 141–151.

García, A. J. & Simari, G. R. (2004), 'Defeasible Logic Programming: An Argumentative Approach', *Theory and Practice of Logic Programming* **4**(1), 95–138.

Grosof, B. N., Horrocks, I., Volz, R. & Decker, S. (2003), 'Description Logic Programs: Combining Logic Programs with Description Logics', *WWW2003, Budapest, Hungary* .

Gruber, T. R. (1993), 'A translation approach to portable ontologies', *Knowledge Acquisition* **5**(2), 199–220.

Haarslev, V. & Möller, R. (2001), RACER System Description, Technical report, University of Hamburg, Computer Science Department.

Heymans, S. & Vermeir, D. (2002), A Defeasible Ontology Language, *in* 'CoopIS/DOA/ODBASE', pp. 1033–1046.

Huang, Z., van Harmelen, F. & ten Teije, A. (2005), Reasoning with inconsistent ontologies, *in* 'Proc. of the Nineteenth Intl. Joint Conference on Artificial Intelligence (IJCAI'05)', pp. 454–459.

Klein, M. (2001), Combining and relating ontologies: an analysis of problems and solutions, *in* A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt & M. Uschold, eds, 'Workshop on Ontologies and Information Sharing, IJCAI'01', Seattle, USA.

Laera, L., Tamma, V., Euzenat, J., Bench-Capon, T. & Payne, T. (2006), Reaching agreement over ontology alignments, *in* 'Proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA'.

McGuiness, D. L. & van Harmelen, F. (2004), 'OWL Web Ontology Language Overview'.

Mitra, P. (2004), An Algebraic Framework for the Interoperation of Ontologies, PhD thesis, Dept. of Electrical Eng., Stanford Univ.

Prakken, H. & Vreeswijk, G. (2002), Logics for Defeasible Argumentation, *in* D. Gabbay & F. Guenthner, eds, 'Handbook of Philosophical Logic', Kluwer Academic Publisher, pp. 219–318.

Simari, G. R. & Loui, R. P. (1992), 'A Mathematical Treatment of Defeasible Reasoning and its Implementation', *Artificial Intelligence* **53**, 125–157.

Volz, R. (2004), Web Ontology Reasoning with Logic Databases, PhD thesis, Universität Fridericiana zu Karlsruhe.

Williams, M. & Hunter, A. (2007), 'Harnessing ontologies for argument-based decision-making in breast cancer', *Proc. of the Intl. Conf. on Tools with AI (ICTAI'07)* pp. 254–261.

Towards Distributed Tableaux Reasoning Procedure for DDL with Increased Subsumption Propagation between Remote Ontologies

Martin Homola[1] **Luciano Serafini**[2]

[1]Faculty of Mathematics, Physics and Informatics,
Comenius University, Mlynska dolina, 842 48 Bratislava, Slovakia,
Email: `homola@fmph.uniba.sk`

[2] FBK-IRST, Via Sommarive 18, 38050 Povo, Trento, Italy
Email: `serafini@fbk.eu`

Abstract

Distributed Description Logics (DDL) enable reasoning with multiple ontologies interconnected by directional semantic mapping. In DDL semantic mapping is indicated by so called bridge rules. There are two kinds: into- and onto-bridge rules. Mappings between concepts allow subsumption to propagate from one ontology to another. However, in some cases, especially when more than two local ontologies are involved, subsumption does not always propagate as we would expect. In a recent study, an adjusted semantics has been introduced that is able to cope with more complex scenarios. In particular, subsumption propagates along chains of several bridge rules under this new semantics. This study makes use of so called compositional consistency requirement that has been employed before in Package-based description logics. While the results concerning subsumption propagation under the adjusted semantics are encouraging, we show in this paper that this semantics also has drawbacks. In certain situations it violates the directionality principle. We propose a weaker version of the semantics in this paper that is able to cope with chains of onto-bridge rules but it is not able to deal with chains of into-bridge rules. Furthermore we provide a sound and complete tableaux reasoning algorithm for this semantics.

1 Introduction and Motivation

Distributed Description Logics (DDL) have been introduced by Borgida and Serafini in (Borgida & Serafini 2003) and later developed in (Serafini & Tamilin 2004, Serafini et al. 2005). DDL are intended especially to enable reasoning over systems of multiple ontologies connected by directional semantic mapping, built upon the formal, logical and well established framework of Description Logics (DL) (Baader et al. 2003). DDL capture the idea of importing and reusing concepts between several ontologies. This idea combines well with the basic assumption of the Semantic Web that no central ontology but rather many ontologies with redundant knowledge will exist (Berners-Lee et al. 2001).

In DDL special syntactic constructs, dubbed bridge rules, are introduced for encoding semantic mapping between ontologies in a formal fashion. With bridge rules one is able to assert that some concept, say C, local to ontology $\mathcal{T}_1$, is mapped to an independent ontology $\mathcal{T}_2$ as a subconcept/superconcept of some $\mathcal{T}_2$-local concept, say D. Moreover, bridge rules are directed, and hence if there is a bridge rule with direction from $\mathcal{T}_1$ to $\mathcal{T}_2$, then $\mathcal{T}_2$ reuses knowledge from $\mathcal{T}_1$ but not necessarily the other way around. Semantic mapping encoded with bridge rules enables for knowledge reuse. In particular, we talk about subsumption propagation from one ontology to another. Consider the example depicted in Fig. 1. Let the ontology on the right be our local back-yard ontology. In this ontology we maintain knowledge about all animals in our backyard. Instead of complex modeling of all relations between our animals we reuse a far more complex ontology that contains classification of all animal species. We map MyCat to genus Felis, and we also create the concept DangerousAnimal and map it to Felidae, the family of all cats, since we also keep a hamster and we know that all cats hunt hamsters for food. Thanks to the semantics of DDL we are now able to derive that MyCat is a subconcept of DangerousAnimal, even if this relation is not derived in the original backyard ontology locally. We say that the subsumption between Felis and Felidae has propagated to the backyard ontology thanks to the mapping.

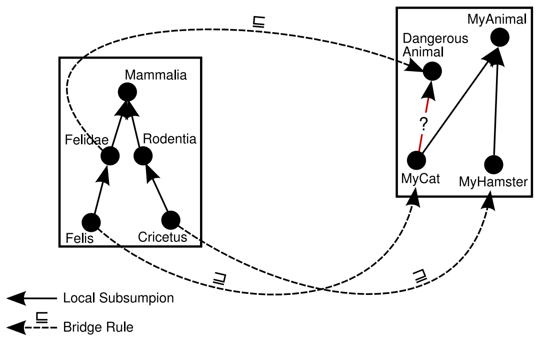


Figure 1: A DDL knowledge base with bridge rules. In the ontology on the right a query is marked up with "?"; we query whether MyCat is a subconcept of DangerousAnimal. This query is answered "yes" under the original DDL semantics.

Please note also that the direction of the mapping is from the species ontology to the backyard ontology, as we only intend to reuse the knowledge of the species ontology within the backyard ontology and not the other way around. The mapping depicted on Fig. 1 does not affect the knowledge within the species ontology in any way.

Subsumption propagation, as seen above, has been described as desired and one of the main features of DDL and it has been studied (cf. (Borgida & Serafini 2003, Serafini & Tamilin 2004, Serafini et al. 2005, Homola 2007, 2008)). In (Homola 2008) we have argued that the original semantics of DDL behaves counter-intuitively in certain cases and we have provided an adjustment to the semantics in order to cope with this issue. For illustration consider Fig. 2 which extends the previously discussed example depicted in Fig. 1. In this case, the situation is more complex. We no longer map between concepts Felidae and DangerousAnimal, but instead these two concepts are connected indirectly via the concept Carnivore from yet another ontology. This third ontology deals with animal behaviour in contrast with the species ontology that merely provides classification. In case that there are such ontologies available, we suggest that this way of modelling is even more natural. As in the above example, again we would expect to derive that MyCat is a subconcept of DangerousAnimal within the backyard ontology. This relation is not derived under the original DDL semantics however.

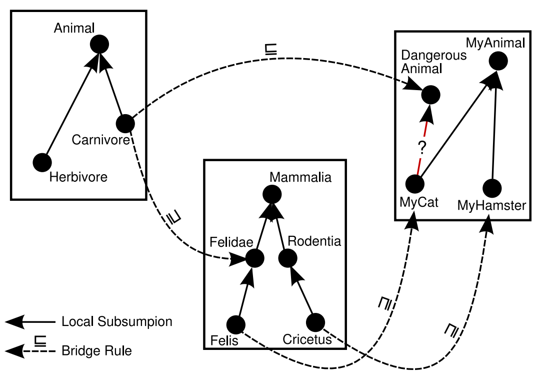


Figure 2: An example of complex concept mapping between three ontologies. In this case the query whether MyCat is a subconcept of DangerousAnimal is not true under the original DDL semantics.

In the example from Fig. 2 there is a so called chain of bridge rules. The onto-bridge rule between concepts Carnivore and Felidae and the other one between Felis and MyCat form a chain, as Felis is a subconcept of Felidae in the classification ontology. Similar chains are also possible with into-bridge rules. The adjusted semantics of (Homola 2008) allows subsumption propagation along such chains of bridge rules and hence the subsumption between MyCat and DangerousAnimal is entailed under this semantics.

In this paper we take steps forward in order to develop a distributed tableaux reasoning algorithm that would decide satisfiability of concepts and subsumption with respect to the adjusted semantics. The algorithm that is introduced in this paper handles chaining onto-bridge rules correctly, but it is unable to cope with chaining into-bridge rules. We provide precise characterization of the semantics that the algorithm actually implements. The algorithm works with acyclic DDL knowledge bases build on top of $\mathcal{ALC}$ as the local language. As in the case of the tableaux algorithm that is known for the original semantics (Serafini et al. 2005), the algorithm is truly distributed and it permits the scenario where every local ontology is governed by an autonomous reasoning service and these services communicate by passing queries. The local reasoner that has started the computation collects all the answers and makes the decision at the end. Besides the fact that each algorithm works under a different semantics, the main distinguishing feature of the newly introduced algorithm is that communication between local reasoners is divided into multiple messages. We believe that such behaviour may serve as a base for more fine-grained optimization in the future.

2 Distributed Description Logics

Distributed Description Logics have been introduced in (Borgida & Serafini 2003), as a formal framework for reasoning with multiple local ontologies interconnected with semantic mapping. Faithful to their namesake, DDL rely on Description Logics (Baader et al. 2003) as for the representation language of the local knowledge bases. DL provide a well established ontological representation language with formal semantics and with reasoning tasks and associated computational issues well understood. In the following we briefly introduce the $\mathcal{ALC}$ DL in oder to be able to build on it later in the paper. DDL, however, are also built over more expressive DL up to $\mathcal{SHIQ}$ (Horrocks et al. 1999).

A basic form of DL knowledge base is a TBox. We will denote TBoxes with $\mathcal{T}_i$ because we deal with several ontologies. Assume that i is an index from some index set. Each TBox is a set of subsumption axioms called general concept inclusions (GCIs), each of the form $i : C \sqsubseteq D$, where C and D are concepts. Concepts are either atomic or complex. Atomic concepts have no further structure. Complex concepts are composed from atomic concepts and roles using some of the DL concept constructors. For list of constructors and their meaning see Fig. 3. The semantics of DL is model-theoretic. Each ontology $\mathcal{T}_i$ is assigned an interpretation $\mathcal{I}_i$ consisting of a domain $\Delta^{\mathcal{I}_i}$ and an interpretation function $\cdot^{\mathcal{I}_i}$ which interprets each concept C by $C^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i}$. Each role R is interpreted by $R^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$. Complex concepts must satisfy further semantic constraints, see Fig. 3. There are also two special concepts $\bot$ and $\top$. The bottom concept ($\bot$) is always interpreted by $\bot^{\mathcal{I}_i} = \emptyset$, the top concept is always interpreted by $\top^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i}$, but they are formally defined as syntactic sugar: $\bot \equiv E \sqcap \neg E$ and $\top \equiv E \sqcup \neg E$, for some new concept name E. A concept C is in negation normal form (NNF) if negation ($\neg$) only appears in C directly in front of atomic concepts. For each concept, an equivalent concept that is in NNF exists (Baader et al. 2003), we denote NNF of C by $\mathrm{nnf}(C)$. Finally, a GCI axiom $i : C \sqsubseteq D$, is satisfied by $\mathcal{I}_i$ whenever $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. An interpretation $\mathcal{I}_i$ is a model of $\mathcal{T}_i$ if it satisfies all GCIs in $\mathcal{T}_i$. And, subsumption $i : C \sqsubseteq D$ is entailed by $\mathcal{T}_i$ if $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$ holds in all models of $\mathcal{T}_i$. Satisfiability of concepts and subsumption entailment are standard decision problems that are investigated in each DL. For more detailed introduction to DL please refer to (Baader et al. 2003, Horrocks et al. 1999).

E	$E^{\mathcal{I}_i}$
$\neg C$	$\Delta^{\mathcal{I}_i} \setminus C^{\mathcal{I}_i}$
$C \sqcap D$	$C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}$
$C \sqcup D$	$C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i}$
$\forall R.C$	$\{i \in \Delta^{\mathcal{I}_i} \mid (\forall j \in \Delta^{\mathcal{I}_i})\ (i,j) \in R^{\mathcal{I}_i} \implies j \in C^{\mathcal{I}_i}\}$
$\exists R.C$	$\{i \in \Delta^{\mathcal{I}_i} \mid (\exists j \in \Delta^{\mathcal{I}_i})\ (i,j) \in R^{\mathcal{I}_i} \wedge j \in C^{\mathcal{I}_i}\}$

Figure 3: Complex concepts in $\mathcal{ALC}$ and their semantics.

A DDL knowledge base, commonly dubbed distributed TBox $\mathfrak{T}$, includes a set of local TBoxes each over its own DL language and a set of bridge rules $\mathfrak{B}$ that provides mappings between local TBoxes. The local languages are commonly required to be under $\mathcal{SHIQ}$ (Horrocks et al. 1999) as it is not trivial to extend DDL with nominals (Serafini et al. 2005). In this work, we use the formal definition as follows.

Definition 1. *Assume a DL language $\mathcal{L}$ – a sub-language of $\mathcal{SHIQ}$, a non-empty index set I, a set of concept names $\mathrm{N_C} = \bigcup_{i\in I} \mathrm{N_C}_i$ and a set of role names $\mathrm{N_R} = \bigcup_{i\in I} \mathrm{N_R}_i$. A Distributed TBox over $\mathcal{L}$ is a pair $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \mathfrak{B}\rangle$ such that:*

1. *Each local TBox $\mathcal{T}_i$ is a collection of general concept inclusions (GCIs) over $\mathrm{N_C}_i$ and $\mathrm{N_R}_i$ in the local language of $\mathcal{T}_i$, a sub-language of $\mathcal{L}$. Each GCI is an axiom of the form*
$$i : C \sqsubseteq D \ .$$
2. *The set of bridge rules $\mathfrak{B}$ divides into sets of bridge rules $\mathfrak{B} = \bigcup_{i,j\in I, i\neq j} \mathfrak{B}_{ij}$. Each $\mathfrak{B}_{ij}$ is a collection of bridge rules in direction from $\mathcal{T}_i$ to $\mathcal{T}_j$ which are of two forms,* into-*bridge rules and* onto-*bridge rules (in the respective order):*
$$i : A \overset{\sqsubseteq}{\rightarrow} j : G \ , \qquad i : B \overset{\sqsupseteq}{\rightarrow} j : H \ .$$

As we have already mentioned in the introduction, the direction of bridge rules matters and hence $\mathfrak{B}_{ij}$ and $\mathfrak{B}_{ji}$ are possibly and expectedly distinct. The bridge graph $\mathrm{G}_\mathfrak{T} = \langle V, E\rangle$ of a distributed TBox $\mathfrak{T}$ is defined as follows: $V = I$ and $\langle i, j\rangle \in E$ if $\mathfrak{B}_{ij} \neq \emptyset$. We say that $\mathfrak{T}$ is acyclic if $\mathrm{G}_\mathfrak{T}$ is acyclic.

Given a TBox $\mathcal{T}$, a hole is an interpretation $\mathcal{I}^\epsilon = \langle\emptyset, \cdot^\epsilon\rangle$ with empty domain. Holes are used for fighting propagation of inconsistency. We use the most recent definition for holes, introduced in (Serafini et al. 2005). A distributed interpretation $\mathfrak{I} = \langle\{\mathcal{I}_i\}_{i\in I}, \{r_{ij}\}_{i\in I, i\neq j}\rangle$ of a distributed TBox $\mathfrak{T}$ consists of a set of local interpretations $\{\mathcal{I}_i\}_{i\in I}$ and a set of domain relations $\{r_{ij}\}_{i\in I, i\neq j}$. .or each $i \in I$, either $\mathcal{I}_i = (\Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i})$ is an interpretation of local TBox $\mathcal{T}_i$ or $\mathcal{I}_i = \mathcal{I}^\epsilon$ is a hole. Each domain relation r_{ij} is a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. We denote by $r_{ij}(d)$ the set $\{d' \mid \langle d, d'\rangle \in r_{ij}\}$ and by $r_{ij}(D)$ the set $\bigcup_{d\in D} r_{ij}(d)$.

Definition 2. *For every i and j, a distributed interpretation $\mathfrak{I}$ satisfies the elements of a distributed TBox $\mathfrak{T}$ (denoted by $\mathfrak{I} \models_\epsilon \cdot$) according to the following clauses:*

1. *$\mathfrak{I} \models_\epsilon i : C \sqsubseteq D$ if $\mathcal{I}_i \models C \sqsubseteq D$.*
2. *$\mathfrak{I} \models_\epsilon \mathcal{T}_i$ if $\mathfrak{I} \models_\epsilon i : C \sqsubseteq D$ for each $C \sqsubseteq D \in \mathcal{T}_i$.*
3. *$\mathfrak{I} \models_\epsilon i : C \overset{\sqsubseteq}{\rightarrow} j : G$ if $r_{ij}(C^{\mathcal{I}_i}) \subseteq G^{\mathcal{I}_j}$.*
4. *$\mathfrak{I} \models_\epsilon i : C \overset{\sqsupseteq}{\rightarrow} j : G$ if $r_{ij}(C^{\mathcal{I}_i}) \supseteq G^{\mathcal{I}_j}$.*
5. *$\mathfrak{I} \models_\epsilon \mathfrak{B}$ if $\mathfrak{I}$ satisfies all bridge rules in $\mathfrak{B}$.*
6. *$\mathfrak{I} \models_\epsilon \mathfrak{T}$ if $\mathfrak{I} \models_\epsilon \mathfrak{B}$ and $\mathfrak{I} \models_\epsilon \mathcal{T}_i$ for each i.*

If $\mathfrak{I} \models_\epsilon \mathfrak{T}$ then we also say that $\mathfrak{I}$ is a (distributed) model of $\mathfrak{T}$.

The two standard decision problems in DL, satisfiability of concepts and entailment of subsumption, play prominent rôle also in context of DDL. Formally, the decision problems are defined as follows.

Definition 3. *Given a distributed TBox $\mathfrak{T}$, an i-local concept C is satisfiable with respect to $\mathfrak{T}$ if there exists a distributed model $\mathfrak{I}$ of $\mathfrak{T}$ such that $C^{\mathcal{I}_i} \neq \emptyset$.*

Definition 4. *Given a distributed TBox $\mathfrak{T}$ and two i-local concepts C and D, it is said that C is subsumed by D with respect to $\mathfrak{T}$ if $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$ in every distributed model $\mathfrak{I}$ of $\mathfrak{T}$. We also sometimes say that the subsumption formula $i : C \sqsubseteq D$ is entailed by $\mathfrak{T}$ and denote this by $\mathfrak{T} \models_\epsilon i : C \sqsubseteq D$.*

It is a well known result that in most DL subsumption and unsatisfiability are inter-reducible. It follows rather straight forward, that this reduction is also valid for DDL, as follows in Theorem 1.

Theorem 1. *Assume a distributed TBox $\mathfrak{T}$ and two i-local concepts C and D. $\mathfrak{T} \models_\epsilon i : C \sqsubseteq D$ if and only if the concept $\neg C \sqcap D$ is unsatisfiable with respect to $\mathfrak{T}$. Also, C is satisfiable with respect to $\mathfrak{T}$ if and only if the subsumption formula $i : C \sqsubseteq \bot$ is not entailed by $\mathfrak{T}$.*

Below in this paper we present a distributed tableaux reasoning algorithm that decides satisfiability of concepts with respect to a distributed TBox for the adjusted semantics for DDL of (Homola 2008). Thanks to this reduction the algorithm provides a decision procedure for both satisfiability and subsumption entailment.

Among the properties of DDL we find the characterization of subsumption propagation, which formally describes the mechanism of knowledge reuse of DDL. Theorem 2 (Serafini & Tamilin 2004) constitutes the most basic version of this property: thanks to a pair of one into-bridge rule and one onto-bridge rule local subsumption relationship is propagated from the source ontology of these bridge rules to the target ontology.

Theorem 2 (Simple subsumption propagation)**.** *In each distributed TBox $\mathfrak{T}$ with two bridge rules $i : C \overset{\sqsupseteq}{\rightarrow} j : G \in \mathfrak{B}$ and $i : D \overset{\sqsubseteq}{\rightarrow} j : H \in \mathfrak{B}$ the following holds:*
$$\mathfrak{T} \models_\epsilon i : C \sqsubseteq D \implies \mathfrak{T} \models_\epsilon j : G \sqsubseteq H \ .$$

Study of subsumption propagation in more complex cases appears in the literature (Serafini et al. 2005, Homola 2007). These cases are not as much interesting for this study, since they do not extend the case captured by Theorem 2 in that aspect that only two local ontologies that are directly connected with bridge rules are studied. In this work we concentrate on indirectly connected ontologies. Various other properties of DDL have showed that DDL constitute a monotonic logic, effect of bridge rules is directional, and that if some of the local ontologies is inconsistent, it does not necessarily pollute the whole distributed system. The reader is kindly redirected to (Borgida & Serafini 2003, Serafini & Tamilin 2004, Serafini et al. 2005, Homola 2007) for all details and discussion.

3 Towards Improved Subsumption Propagation in DDL

In our recent paper (Homola 2008) we have adjusted the original semantics of DDL in order to improve subsumption propagation between remote ontologies, in cases when local ontologies are only connected indirectly, as in the example from Fig. 2. The adjustment exploits the *compositional consistency* requirement, that is known from Package-based Description Logics (P-DL) (Bao et al. 2006). In a nutshell, this requirement only allows transitive domain relations in distributed interpretations. Formal definition follows in Definition 5.

Definition 5. *Given a distributed interpretation $\mathfrak{I}$ with domain relation r, we say that r (and also $\mathfrak{I}$) satisfies compositional consistency if for each $i, j, k \in I$ and for each $x \in \Delta^{\mathcal{I}_i}$ with $r_{ij}(x) = D$ we have $r_{jk}(D) = r_{ik}(x)$.*

Now the adjusted semantics is simply obtained from the original DDL semantics by allowing only distributed interpretations that satisfy compositional consistency. In accordance we often use the wording "DDL under compositional consistency" when referring to this semantics. The adjusted semantics actually extends the original one, in the sense that if some subsumption formula Φ is entailed by a distributed TBox $\mathfrak{T}$ in the original semantics, then it is also entailed by $\mathfrak{T}$ under compositional consistency. The only difference is that in the adjusted semantics possibly some more subsumption formulae are entailed in addition. This is formally stated in the following theorem (see (Homola 2008) for a proof).

Theorem 3. *Given a distributed TBox $\mathfrak{T}$ and a subsumption formula Φ, if $\mathfrak{T} \models_\epsilon \Phi$ according to the original semantics, then $\mathfrak{T} \models_\epsilon \Phi$ also holds in DDL under compositional consistency.*

To demonstrate the mechanism of improved subsumption propagation within the adjusted semantics, let us revisit the example from Fig. 2 formally.

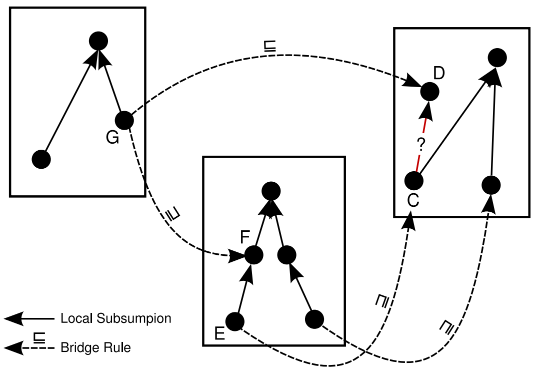


Figure 4: Renaming of concepts from Fig. 2 employed in Example 1. The three local ontologies are referred to as $\mathcal{T}_b$ – behaviour ontology, $\mathcal{T}_c$ – classification ontology, and $\mathcal{T}_y$ – backyard ontology.

Example 1. *Recall the example depicted in Fig. 2. Let's simplify the notation by renaming the concepts as follows: $C :=$ MyCat, $D :=$ DangerousAnimal, $E :=$ Felis, $F :=$ Felidae, $G :=$ Carnivore. Remaining concepts are of no interest (see Fig. 4). Assume the index set $I = \{b, c, y\}$, where b represents the behaviour ontology (on the left), c represents the classification ontology (in the middle) and y represents the backyard ontology (on the right). There are various axioms in $\mathfrak{T} = \langle\{\mathcal{T}_b, \mathcal{T}_c, \mathcal{T}_y, \}, \mathfrak{B}\rangle$ but of the GCIs only $c : E \sqsubseteq F$, actually matters to us, and there are three bridge rules in $\mathfrak{B}$:*

$$b : G \xrightarrow{\sqsupseteq} c : F \ , \qquad c : E \xrightarrow{\sqsupseteq} y : C \ ,$$
$$b : G \xrightarrow{\sqsubseteq} y : D \ .$$

We query whether it holds that $\mathfrak{T} \models_\epsilon y : C \sqsubseteq D$. Assume a distributed interpretation $\mathfrak{I}$. Let us first assume that $\mathfrak{I}$ contains no hole. From Definition 2 we get $r_{cy}(E^{\mathcal{I}_c}) \supseteq C^{\mathcal{I}_y}$. We also have $r_{bc}(G^{\mathcal{I}_b}) \supseteq F^{\mathcal{I}_c}$, but since $r_{cy}(\cdot)$ is a mapping we get that $r_{cy}(r_{bc}(G^{\mathcal{I}_b})) \supseteq r_{cy}(F^{\mathcal{I}_c})$. And from compositional consistency we get $r_{cy}(r_{bc}(G^{\mathcal{I}_b})) = r_{by}(G^{\mathcal{I}_b})$ and so $r_{by}(G^{\mathcal{I}_b}) \supseteq r_{cy}(F^{\mathcal{I}_c})$. From the GCI $c : E \sqsubseteq F$ we get $F^{\mathcal{I}_c} \supseteq E^{\mathcal{I}_c}$ and from properties of mapping we again get $r_{cy}(F^{\mathcal{I}_c}) \supseteq r_{cy}(E^{\mathcal{I}_c})$. Putting this all together we derive:

$$r_{by}(G^{\mathcal{I}_b}) \supseteq r_{cy}(F^{\mathcal{I}_c}) \supseteq r_{cy}(E^{\mathcal{I}_c}) \supseteq C^{\mathcal{I}_y} \ ,$$

and that amounts to $r_{by}(G^{\mathcal{I}_b}) \supseteq C^{\mathcal{I}_y}$. On the other hand, from the into-bridge rule between $\mathcal{T}_b$ and $\mathcal{T}_y$ we derive $r_{by}(G^{\mathcal{I}_b}) \subseteq D^{\mathcal{I}_y}$. And so we finally get $C^{\mathcal{I}_y} \subseteq D^{\mathcal{I}_y}$.

For the case with holes assume for instance that $\mathcal{I}_b = \mathcal{I}^\epsilon$. In that case $G^{\mathcal{I}_b} = \emptyset$. Thanks to the constraint generated by bridge rules and the GCI we easily derive that also $F^{\mathcal{I}_c} = \emptyset$, $E^{\mathcal{I}_c} = \emptyset$, and also $C^{\mathcal{I}_y} = \emptyset$. In such a case, however, $C^{\mathcal{I}_y} \subseteq D^{\mathcal{I}_y}$ holds trivially. If we substitute other local interpretations for holes, we get the very same result similarly.

Summing up, in every model of $\mathfrak{T}$ we have $C^{\mathcal{I}_y} \subseteq D^{\mathcal{I}_y}$ and hence $\mathfrak{T} \models_\epsilon y : C \sqsubseteq D$. Recall that we have actually used the compositional consistency requirement in our argumentation. Without it we would not be able to establish the result: we would not be able to prove that $r_{by}(G^{\mathcal{I}_b}) \supseteq r_{cy}(F^{\mathcal{I}_c})$. In fact, the original DDL semantics allows models that violate this inclusion and hence $\mathfrak{T} \models_\epsilon y : C \sqsubseteq D$ does not hold under the original semantics.

Theorem 4 below provides a more general characterization of cases when subsumption propagates to remote ontologies. This characterization generalizes the setting from Figs. 2-4. For a proof, please refer to (Homola 2008).

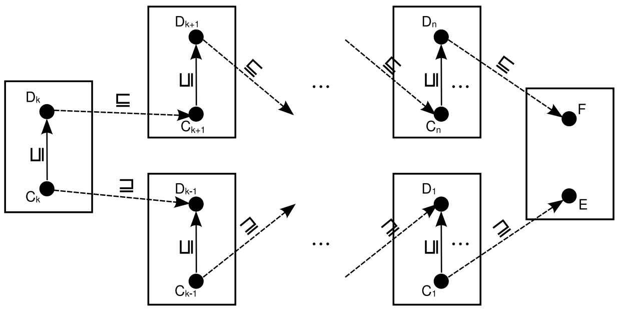


Figure 5: Depiction of the distributed TBox from Theorem 4. Local subsumption is indicated by solid arrows and bridge rules by dashed arrows.

Theorem 4. *Given a distributed TBox, as illustrated in Fig. 5, with index set I and set of bridge rules $\mathfrak{B}$, that features $n + 1$ local TBoxes $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_n$ with concepts $E, F \in \mathcal{T}_0$, and $C_i, D_i \in \mathcal{T}_i$, for $1 \leq i \leq n$, and k with $1 \leq k \leq n$ such that:*

1. *$\mathfrak{T} \models_\epsilon i : C_i \sqsubseteq D_i$, for $1 \leq i \leq n$,*
2. *$i + 1 : C_{i+1} \xrightarrow{\sqsupseteq} i : D_i \in \mathfrak{B}$, for $1 \leq i < k$,*
3. *$i : D_i \xrightarrow{\sqsubseteq} i + 1 : C_{i+1} \in \mathfrak{B}$, for $k \leq i < n$,*
4. *$1 : C_1 \xrightarrow{\sqsupseteq} 0 : E \in \mathfrak{B}$ and $n : D_n \xrightarrow{\sqsubseteq} 0 : F \in \mathfrak{B}$.*

In DDL under compositional consistency it follows that $\mathfrak{T} \models_\epsilon 0 : E \sqsubseteq F$.

In a nutshell, Theorem 4 basically says that the effect of bridge rules is now transitive, and hence subsumption propagates even between remote ontologies within the system. Unfortunately, current notation used with DDL does not allow us to formally state this in a more elegant and easier to read fashion.

4 DDL under the Transitivity Condition

Contrary to the claims of (Homola 2008), the adjusted semantics suffers from some drawbacks. The example below demonstrates that the directionality property is violated in this new setting.

Example 2. *Consider a distributed TBox $\mathfrak{T}$ with three local TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ and two bridge rules:*

$$1 : A \xrightarrow{\sqsupseteq} 2 : \top \ , \qquad 1 : A \xrightarrow{\sqsubseteq} 3 : \bot \ .$$

Assuming that $\mathcal{T}_2$ is consistent we have $\top^{\mathcal{I}_2} \neq \emptyset$ and hence there must be some $y \in \top^{\mathcal{I}_2}$. In such a case the first bridge rule requires that there also is $x \in A^{\mathcal{I}_1}$ with $y \in r_{12}(x)$. However, the compositional consistency condition requires that $r_{32}(r_{13}(x)) = r_{12}(x)$ and hence, there must also be some $z \in \Delta^{\mathcal{I}_3}$ such that $z \in r_{13}(x)$. But for each such z the second bridge rule requires that $z \in \bot^{\mathcal{I}_3}$, which violates the condition that $\bot^{\mathcal{I}_3} = \emptyset$. Hence inconsistency is induced within $\mathcal{T}_2$.

This violates directionality because, as there is no directed path from 2 to 3 in $\mathrm{G}_{\mathfrak{T}}$, there should be no change in the semantics of $\mathcal{T}_3$ if we remove $\mathcal{T}_2$ from $\mathfrak{T}$. But indeed, if we remove $\mathcal{T}_2$, then $\mathcal{T}_3$ is no longer inconsistent.

From the example we conclude that the compositional consistency requirement is perhaps too strong. It implies all the constraints that are necessary in order to increase subsumption propagation in the amount we have described as desired but it also has some more consequences that are possibly destructive as we have seen in Example 2. In the following we study a relaxed version of the compositional consistency condition, which basically is just transitivity of the domain relation in distributed models. We will show that subsumption propagation is more limited under this weaker condition, but on the other hand we will be able to introduce a distributed tableaux reasoning algorithm that decides satisfiability for this semantics. Formally, the weaker condition is defined as follows.

Definition 6. *Given a distributed interpretation $\mathfrak{I}$ with domain relation r, we say that r (and also $\mathfrak{I}$) satisfies the transitivity condition if for each $i, j, k \in I$ and for each $x \in \Delta^{\mathcal{I}_i}$ with $r_{ij}(x) = D$ we have $r_{jk}(D) \subseteq r_{ik}(x)$.*

Apparently, a distributed interpretation satisfies the transitivity condition if and only if its domain relation is transitive. The semantics obtained by allowing only distributed interpretations that satisfy the transitivity condition is hence called DDL under transitivity or DDL with transitive domain relation. The first observation for the new semantics is that the proposition of Theorem 4 does not hold any more if we relax from compositional consistency to transitivity. We demonstrate this problem by an example.

Example 3. *Consider a distributed TBox $\mathfrak{T}$ with three local TBoxes $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, with local concepts C_1, D_1 in $\mathcal{T}_1$, E_2 in $\mathcal{T}_2$ and F_3 in $\mathcal{T}_3$, and with bridge rules:*

$$2 : E_2 \xrightarrow{\sqsubseteq} 3 : F_3 \ , \qquad 3 : F_3 \xrightarrow{\sqsubseteq} 1 : D_1 \ ,$$

$$2 : E_2 \xrightarrow{\sqsupseteq} 1 : C_1 \ .$$

This distributed TBox is clearly a very simple instance of the setting assumed by Theorem 4. According to this theorem, we should be able to derive $\mathfrak{T} \models_{\mathrm{d}} 1 : C_1 \sqsubseteq D_1$. This is not true, however, as a distributed model of $\mathfrak{T}$ exists in which ${C_1}^{\mathcal{I}_1} \not\subseteq {D_1}^{\mathcal{I}_1}$. This distributed model $\mathfrak{I}$ has three local domains $\Delta^{\mathcal{I}_1} = \{c_1, d_1\}$, $\Delta^{\mathcal{I}_2} = \{e_2\}$, and $\Delta^{\mathcal{I}_3} = \{f_3\}$. Local concepts are interpreted as follows: ${C_1}^{\mathcal{I}_1} = \{c_1, d_1\}$, ${D_1}^{\mathcal{I}_1} = \{d_1\}$, ${E_2}^{\mathcal{I}_2} = \{e_2\}$, and ${F_3}^{\mathcal{I}_3} = \{f_3\}$. Finally, the domain relation is the following: $r_{21} = \{\langle e_2, c_1\rangle, \langle e_2, d_1\rangle\}$, $r_{23} = \{\langle e_2, f_3\rangle\}$, $r_{31} = \{\langle f_3, d_1\rangle\}$, $r_{12} = r_{32} = r_{13} = \emptyset$. It is easily verified that r is transitive, hence the transitivity condition is satisfied, and hence Theorem 4 does not stand if compositional consistency is relaxed to transitivity. At the same time, this example does not invalidate Theorem 4 under compositional consistency, as the stronger condition is not satisfied by $\mathfrak{I}$ because $r_{21}(e_2) = \{c_1, d_1\} \not\subseteq \{d_1\} = r_{31}(r_{23}(e_2))$.

So we see that DDL with transitive domain relation has problems with "chaining" into-bridge rules. Luckily enough the problem does not repeat with chaining onto-bridge rules – if we reinspect Example 1 we will see that transitivity is enough to guarantee the two chaining onto-bridge rules in this example to propagate the subsumption relationship as expected. This observation holds in general an is formally established by Theorem 5 below. This theorem is a weaker version of Theorem 4 and provides characterization of the semantics of DDL under the transitivity condition.

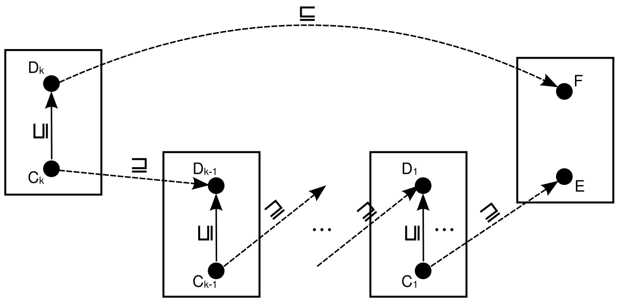


Figure 6: Depiction of the distributed TBox from Theorem 5. Local subsumption is indicated by solid arrows and bridge rules by dashed arrows.

Theorem 5. *Given a distributed TBox, as illustrated in Fig. 6, with index set I and set of bridge rules $\mathfrak{B}$, that features $k+1$ local TBoxes $\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_k$ with concepts $E, F \in \mathcal{T}_0$, and $C_i, D_i \in \mathcal{T}_i$, for $1 \leq i \leq k$, such that:*

1. *$\mathfrak{T} \models_\epsilon i : C_i \sqsubseteq D_i$, for $1 \leq i \leq k$,*
2. *$i+1 : C_{i+1} \xrightarrow{\sqsupseteq} i : D_i \in \mathfrak{B}$, for $1 \leq i < k$,*
3. *$1 : C_1 \xrightarrow{\sqsupseteq} 0 : E \in \mathfrak{B}$ and $k : D_k \xrightarrow{\sqsubseteq} 0 : F \in \mathfrak{B}$.*

In DDL under the transitivity condition it follows that $\mathfrak{T} \models_\epsilon 0 : E \sqsubseteq F$.

Proof. The theorem is proved by the following chain of inclusions:

$$E^{\mathcal{I}_0} \overset{1}{\subseteq} r_{k0}\big({C_k}^{\mathcal{I}_k}\big) \overset{2}{\subseteq} r_{k0}\big({D_k}^{\mathcal{I}_k}\big) \overset{3}{\subseteq} F^{\mathcal{I}_0} \ .$$

Inclusion 3 is a direct consequence of the into-bridge rule $k : D_k \xrightarrow{\sqsubseteq} 0 : F \in \mathfrak{B}$. As for Inclusion 2, $\mathfrak{T} \models_{\mathrm{d}} k : C_k \sqsubseteq D_k$ implies ${C_k}^{\mathcal{I}_k} \subseteq {D_k}^{\mathcal{I}_k}$ and since $r_{k0}(\cdot)$ we also have $r_{k0}\big({C_k}^{\mathcal{I}_k}\big) \subseteq r_{k0}\big({D_k}^{\mathcal{I}_k}\big)$. Inclusion 1 divides into:

$$E^{\mathcal{I}_0} \subseteq r_{10}\big(r_{21}(\cdots r_{kk-1}({C_k}^{\mathcal{I}_k})\cdots)\big) \subseteq r_{k0}\big({C_k}^{\mathcal{I}_k}\big) \ .$$

Both these inclusions ought to be proved by mathematical induction. The first inclusion is proved by induction on k, base case is $E^{\mathcal{I}_0} \subseteq r_{10}\big({C_1}^{\mathcal{I}_1}\big)$. This is a direct consequence of the onto-bridge rule $1 : C_1 \xrightarrow{\sqsupseteq} 0 : E \in \mathfrak{B}$. In the induction step, by the induction hypothesis we have that $E^{\mathcal{I}_0} \subseteq r_{10}\big(r_{21}\big(\cdots r_{k-1k-2}\big({C_{k-1}}^{\mathcal{I}_{k-1}}\big)\cdots\big)\big)$. From the assumptions of the theorem we derive:

$$ {C_{k-1}}^{\mathcal{I}_{k-1}} \subseteq {D_{k-1}}^{\mathcal{I}_{k-1}} \subseteq r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big) \ . $$

The composition $r_{10}(r_{21}(\cdots r_{k-1k-2}(\cdot)\cdots))$ is still a mapping, and so $r_{10}\big(r_{21}\big(\cdots r_{k-1k-2}\big({C_{k-1}}^{\mathcal{I}_{k-1}}\big)\cdots\big)\big)$ is a subset of $r_{10}\big(r_{21}\big(\cdots r_{k-1k-2}\big(r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big)\big)\cdots\big)\big)$. Together with the induction hypothesis this amounts to the inclusion we wanted to prove.

As for the second inclusion, we shall prove a slightly more general proposition:

$$ r_{n+1n}\big(r_{n+2n+1}\big(\cdots r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{kn}\big({C_k}^{\mathcal{I}_k}\big) \ , $$

where $0 \leq n \leq k-2$. The inclusion we ought to prove is then derived by setting $n = 0$. The proposition is proved by mathematical induction on $k - n$. As for the base case consider $k - n = 2$, and so $n = k - 2$. That means we have to show $r_{k-1k-2}\big(r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big)\big) \subseteq r_{kk-2}\big({C_k}^{\mathcal{I}_k}\big)$, which indeed holds because $\mathfrak{I}$ satisfies the transitivity condition. The induction step is for $k - n = j > 2$, and so $n = k - j$. From the induction hypothesis we get $r_{n+2n+1}\big(r_{n+3n+2}\big(\cdots r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{kn+1}\big({C_k}^{\mathcal{I}_k}\big)$. And since $r_{n+1n}(\cdot)$ is a mapping, we also get $r_{n+1n}\big(r_{n+2n+1}\big(\cdots r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{n+1n}\big(r_{kn+1}\big({C_k}^{\mathcal{I}_k}\big)\big)$. Since $\mathfrak{I}$ satisfies the transitivity condition, we get $r_{n+1n}\big(r_{kn+1}\big({C_k}^{\mathcal{I}_k}\big)\big) \subseteq r_{kn}\big({C_k}^{\mathcal{I}_k}\big)$. By combining the last two inclusions we directly get:

$$ r_{n+1n}\big(r_{n+2n+1}\big(\cdots r_{kk-1}\big({C_k}^{\mathcal{I}_k}\big)\cdots\big)\big) \subseteq r_{kn}\big({C_k}^{\mathcal{I}_k}\big) \ . $$

By setting $n = 0$ we derive the inclusion we wanted to prove, and hence the Theorem. □

And thus we are able to conclude this section by comparing the three semantics of DDL with respect to the amount of subsumption propagation they allow along chaining bridge rules. The original semantics of DDL in general does not guarantee subsumption propagation between remote ontologies along chaining bridge rules. A limited case when subsumption propagates between remote ontologies under the original semantics is described in (Homola 2008). Semantics of DDL with transitive domain relation ensures subsumption propagation along chaining onto-bridge rules in scenarios that instantiate Theorem 5. Finally, in DDL under computational consistency subsumption propagates along chaining onto-bridge rules and along chaining into-bridge rules in addition, as established by Theorem 4.

5 Distributed Tableaux Algorithm for DDL with Transitive Domain Relation

In this section, we introduce a distributed tableaux algorithm for deciding satisfiability of concepts with respect to an acyclic distributed TBox for DDL over $\mathcal{ALC}$ under the transitivity requirement. As in the algorithm of (Serafini et al. 2005), also in our approach, local reasoners are run independently. We keep precise track of the domain relation r however, and the communication between local reasoners is divided into multiple queries. Hence while the original algorithm can be seen as working with autonomous local tableaux by passing queries, our algorithm can be seen as working with a truly distributed tableau.

Definition 7. *Assume a distributed TBox $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \mathfrak{B}\rangle$ over $\mathcal{ALC}$ with index set I, concept names $\mathrm{N_C} = \bigcup_{i\in I} \mathrm{N_C}_i$ and role names $\mathrm{N_R} = \bigcup_{i\in I} \mathrm{N_R}_i$. Let CC_i be the set of all (atomic and complex) concepts over $\mathrm{N_C}_i$ and $\mathrm{N_R}_i$ in negation normal form. A distributed completion tree $T = \{T_i\}_{i\in I}$ is a set of labeled trees $T_i = \langle V_i, E_i, \mathcal{L}_i, r_i\rangle$, such that for each $i \in I$:*

1. *the members of $\{V_i\}_{i\in I}$ are mutually disjoint;*
2. *the members of $\{E_i\}_{i\in I}$ are mutually disjoint;*
3. *the labeling function $\mathcal{L}_i$ labels each node $x \in V_i$ with $\mathcal{L}(x) \subseteq 2^{\mathrm{CC}_i}$ and each edge $\langle x, y\rangle \in E_i$ with $\mathcal{L}(\langle x, y\rangle) \in \mathrm{N_R}_i$;*
4. *the labeling function r_i labels each node $x \in V_i$ with a set of references to its r-images $r_i(x) \subseteq \{j : y \mid j \in I \wedge y \in V_j\}$.*

During the run of the tableaux algorithm, tableaux expansion rules are applied on the completion tree and the tree is expanded by each rule application. If no more rules are applicable any more, we say that the completion tree is *complete*. There is a *clash* in the completion tree T if for some $x \in V_i$ and for some $C \in \mathrm{N_C}_i$ we have $\{C, \neg C\} \subseteq \mathcal{L}_i(x)$. If there is no clash in T then we say that T is *clash-free*. In order to assure termination we use standard subset blocking that is common for $\mathcal{ALC}$. Given a distributed completion tree $T = \{T_i\}_{i\in I}$, a node $x \in V_i$ is blocked, if it has an ancestor $y \in V_i$ such that $\mathcal{L}_i(x) \subseteq \mathcal{L}_i(y)$. In such a case we also say that x is blocked by y. Node $y \in V_i$ is said to be an *R-successor* of $x \in V_i$, if $\langle x, y\rangle \in E_i$ and $\mathcal{L}_i(\langle x, y\rangle) = R$.

The distributed tableaux algorithm for deciding satisfiability of concepts with respect to a distributed TBox takes a distributed TBox $\mathfrak{T}$, a concept C in NNF and $i \in I$ as its inputs. The algorithm then continues in three steps:

1. *Initialization.* Create new completion tree $T = \{T_j\}_{j\in I}$ such that $T_j = \langle\{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \emptyset\rangle$ for $j = i$ and $T_j = \langle\emptyset, \emptyset, \emptyset, \emptyset\rangle$ for $j \neq i$.
2. *Tableau expansion.* Apply the tableaux expansion rules of Fig. 7 exhaustingly.
3. *Answer.* If none of the tableaux expansion rules in Fig. 7 is applicable any more (i.e., the completion tree is now complete), answer "C is satisfiable" if a clash-free completion tree has been constructed. Answer "C is unsatisfiable" otherwise.

Below we present a formal correctness proof for the newly introduced algorithm. The proof is based on the classic proof for $\mathcal{ALC}$ as in fact the only new thing to prove here is that the algorithm uses the $\overset{\sqsupseteq}{\rightarrow}$-rule and the $\overset{\sqsubseteq}{\rightarrow}$-rule to combine several autonomous local $\mathcal{ALC}$ reasoners correctly. The proof is done in three parts. We first prove termination: on every input the algorithm always terminates and never ends up in an infinite loop; then soundness: if the algorithm answers that C is satisfiable with respect to $\mathfrak{T}$ for some i-local concept C and a distributed TBox $\mathfrak{T}$ then there actually exists some model of $\mathfrak{T}$ that supports this; and finally we prove completeness: for every concept C that is satisfiable with respect to $\mathfrak{T}$ the algorithm indeed gives a correct answer.

Theorem 6. *Given a distributed TBox $\mathfrak{T}$ over $\mathcal{ALC}$ with acyclic bridge graph $\mathrm{G}_{\mathfrak{T}}$ and an i-local concept C on the input, the distributed tableaux algorithm for deciding satisfiability of concepts with respect to a distributed TBox over $\mathcal{ALC}$ for DDL with transitive domain relation always terminates and it is sound and complete.*

$\sqcap$-rule:
If $C_1 \sqcap C_2 \in \mathcal{L}_i(x)$ for some $x \in V_i$ and $\{C_1, C_2\} \nsubseteq \mathcal{L}_i(x)$, and x is not blocked,
then set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{C_1, C_2\}$.

$\sqcup$-rule:
If $C_1 \sqcup C_2 \in \mathcal{L}_i(x)$ for some $x \in V_i$ and $\{C_1, C_2\} \cap \mathcal{L}_i(X) = \emptyset$, and x is not blocked,
then either set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{C_1\}$ or set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{C_2\}$.

$\forall$-rule:
If $\forall R.C \in \mathcal{L}_i(x)$ for some $x \in V_i$, and there is R-successor y of x s.t. $C \notin \mathcal{L}_i(y)$, and x is not blocked,
then set $\mathcal{L}_i(y) = \mathcal{L}_i(y) \cup \{C\}$.

$\exists$-rule:
If $\exists R.C \in \mathcal{L}_i(x)$, for some $x \in V_i$ with no R-successor y s.t. $C \in \mathcal{L}_i(y)$, and x is not blocked,
then add new node z to V_i, add the edge $\langle x, z \rangle$ to E_i, and set $\mathcal{L}_i(z) = \{C\}$ and $\mathcal{L}_i(\langle x, z \rangle) = \{R\}$.

$\mathcal{T}$-rule:
If $C \sqsubseteq D \in \mathcal{T}$ and for some $x \in V_i$ $\mathrm{nnf}(\neg C \sqcup D) \notin \mathcal{L}_i(x)$, and x is not blocked,
then set $\mathcal{L}_i(x) = \mathcal{L}_i(x) \cup \{\mathrm{nnf}(\neg C \sqcup D)\}$.

$\overset{\sqsupseteq}{\longrightarrow}$-rule:
If $G \in \mathcal{L}_j(x)$ for some $x \in V_j$, $i : C \overset{\sqsupseteq}{\longrightarrow} j : G \in \mathfrak{B}$, and there is no $y \in V_i$ s.t. $C \in \mathcal{L}_i(y)$ and $j : x \in r_i(y)$, and x is not blocked,
then add new node y to V_i and set $\mathcal{L}_i(y) = \{C\}$, and set $r_i(y) = \{j : x\} \cup r_j(x)$.

$\overset{\sqsubseteq}{\longrightarrow}$-rule:
If $D \in \mathcal{L}_i(x)$ for some $x \in V_i$, $i : D \overset{\sqsubseteq}{\longrightarrow} j : H \in \mathfrak{B}$ and there is $y \in V_j$ s.t. $j : y \in r_i(x)$ and $H \notin \mathcal{L}_j(y)$
then set $\mathcal{L}_j(y) = \mathcal{L}_j(y) \cup \{H\}$.

Figure 7: Tableaux expansion rules for DDL over $\mathcal{ALC}$ under the transitivity requirement. First five rules are standard $\mathcal{ALC}$ tableaux rules. Note that the $\sqcup$-rule is non-deterministic. The final two rules are new and are triggered by bridge rules.

Proof. Termination. Given a distributed TBox $\mathfrak{T}$ with local TBox $\mathcal{T}_i$ and an i-local concept C, we ought to prove that the algorithm, once started with $\mathfrak{T}$, C and i on input, eventually terminates. The algorithm initializes the completion tree to $T = \{T_j\}_{j \in I}$ such that $T_j = \langle \{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \emptyset \rangle$ for $j = i$ and $T_j = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$ for $j \neq i$. The computation then continues by expanding $\mathcal{T}_i$. If there are no onto-bridge rules ingoing into $\mathcal{T}_i$ the algorithm eventually terminates thanks to subset blocking, this result is known for $\mathcal{ALC}$. If there are some ingoing onto-bridge rules then possibly in some $x \in V_i$ such that $C \in \mathcal{L}_i(x)$ and C appears on a right hand side of an onto-bridge rule, say $k : D \overset{\sqsupseteq}{\longrightarrow} i : C$, then the $\overset{\sqsupseteq}{\longrightarrow}$-rule is applied and computation is triggered in T_k. By structural subsumption we assume that this computation eventually terminates (the length of the longest incoming path of into-bridge rules decreased for $\mathcal{T}_k$ compared to $\mathcal{T}_i$, and all such paths are finite because of acyclicity). During this process we possibly get some new concepts in $\mathcal{L}_i(x)$ that are introduced thanks to incoming into-bridge rules that trigger the $\overset{\sqsubseteq}{\longrightarrow}$-rule – but only finitely many. The computation now continues in x and its descendants and possibly the $\overset{\sqsupseteq}{\longrightarrow}$-rule is triggered again in some $y \in V_i$, a descendant of x. But thanks to subset blocking, this happens only finitely many times. Hence the algorithm eventually terminates.

Soundness. Now that we know that the algorithm terminates, we shall prove that if it answers "C is satisfiable" on input $\mathfrak{T}$, C and i, then it also holds that C is satisfiable in $\mathcal{T}_i$ with respect to $\mathfrak{T}$, that is we must show that in such a case there exists a distributed model $\mathfrak{I}$ of $\mathfrak{T}$ such that $C^{\mathcal{I}_i} \neq \emptyset$. Given $\mathfrak{T}$, C and i and let T be the complete and clash-free completion tree that the algorithm has constructed to support the decision. Let us construct the distributed interpretation $\mathfrak{I}$ as follows:

1. Let $\Delta^{\mathcal{I}_i} = V_i$, for each $i \in I$.
2. Let $x \in A^{\mathcal{I}_i}$, for each atomic concept $A \in \mathcal{L}_i(x)$, for each $x \in V_i$ and each $i \in I$.
3. Let $\langle x, y \rangle \in R^{\mathcal{I}_i}$ for each $\langle x, y \rangle \in E_i$ such that $\mathcal{L}_i(\langle x, y \rangle) = R$, for each $i \in I$, if y is not blocked.
4. Let $\langle x, z \rangle \in R^{\mathcal{I}_i}$ for each $\langle x, y \rangle \in E_i$ such that $\mathcal{L}_i(\langle x, y \rangle) = R$, for each $i \in I$, in case that y is blocked by z.
5. Let $r_{ij}(x) = y$ for each $x \in V_i$ and for each $j : y \in r_i(x)$.

Please observe that if computation was never triggered within some T_j of T during the run of the algorithm, then $\mathcal{I}_j = \mathcal{I}^\epsilon$. It remains to show that $\mathfrak{I}$ is in fact a model of $\mathfrak{T}$ and $C^{\mathcal{I}_i} \neq \emptyset$. We will first prove the following proposition:

Given any $i \in I$, for each $E \in \mathcal{L}_i(x)$ (i.e., also for complex concepts) we have $x \in E^{\mathcal{I}_i}$.

This is proved by induction on the structure of E. We need to consider the following cases:

1. E is atomic. We know that $x \in E^{\mathcal{I}_i}$ from the construction.
2. $E = \neg E_1$, E_1 atomic. Since T is clash-free, $E_1 \notin \mathcal{L}_i(x)$ and by construction $x \notin {E_1}^{\mathcal{I}_i}$. In that case however $x \in E^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \setminus {E_1}^{\mathcal{I}_i}$.
3. $E = E_1 \sqcap E_2$. Since T is complete, the $\sqcap$-rule is not applicable and hence also $E_1 \in \mathcal{L}_i(x)$ and $E_2 \in \mathcal{L}_i(x)$. By induction we now have that $x \in {E_1}^{\mathcal{I}_i}$ and $x \in {E_2}^{\mathcal{I}_i}$ and hence also $x \in E^{\mathcal{I}_i}$.
4. $E = E_1 \sqcup E_2$. Since T is complete, the $\sqcup$-rule is not applicable and hence either $E_1 \in \mathcal{L}_i(x)$ or $E_2 \in \mathcal{L}_i(x)$. By induction we now either have $x \in {E_1}^{\mathcal{I}_i}$ or we have $x \in {E_2}^{\mathcal{I}_i}$. In either case however also $x \in E^{\mathcal{I}_i}$.
5. $E = \exists R.E_1$. Since T is complete, the $\exists$-rule is not applicable and hence there must be $y \in V_i$, an R-successor of x such that $E_1 \in \mathcal{L}_i(y)$. By induction $y \in {E_1}^{\mathcal{I}_i}$ and by construction of $\mathfrak{I}$ $\langle x, y \rangle \in R^{\mathcal{I}_i}$. But that means that $x \in E^{\mathcal{I}_i}$.
6. $E = \forall R.E_1$. Since T is complete, the $\forall$-rule is not applicable and hence for every $y \in V_i$, an R-successor of x, we have $E_1 \in \mathcal{L}_i(y)$. By induction $y \in {E_1}^{\mathcal{I}_i}$ and by construction of $\mathfrak{I}$ $\langle x, y \rangle \in R^{\mathcal{I}_i}$. But that means that $x \in E^{\mathcal{I}_i}$.

Thus we have verified that the local interpretation $\mathcal{I}_i$ is indeed an $\mathcal{ALC}$ interpretation and that C has an instance in this interpretation (since $C \in \mathcal{L}_i(s_0)$). In addition we have in fact also proved that each i-local GCI axiom $E_1 \sqsubseteq E_2$ is satisfied by $\mathcal{I}_i$ – from the completeness of T, $\mathrm{nnf}(\neg E_1 \sqcup E_2) \in x$, for each $x \in V_i$, and hence $x \in (\mathrm{nnf}(\neg E_1))^{\mathcal{I}_i} \cup {E_2}^{\mathcal{I}_i}$. It follows that $x \in {E_1}^{\mathcal{I}_i}$ implies $x \in {E_2}^{\mathcal{I}_i}$.

It remains to show that $\mathfrak{I}$ is indeed a distributed model of $\mathfrak{T}$ in the adjusted semantics. First, all the bridge rules must be satisfied. Given an onto-bridge rule $k : E_1 \overset{\sqsupseteq}{\longrightarrow} l : E_2 \in \mathfrak{B}$, we ought to show that $r_{kl}({E_1}^{\mathcal{I}_k}) \supseteq {E_2}^{\mathcal{I}_l}$. So let $x \in {E_2}^{\mathcal{I}_l}$, that is $x \in V_l$ and $E_2 \in \mathcal{L}_l(x)$. But T is complete, $\overset{\sqsupseteq}{\longrightarrow}$-rule is not applicable, and hence there must be $y \in V_k$ with $E_1 \in \mathcal{L}_k(y)$ and $l : x \in r_k(y)$. That means however that $y \in {E_1}^{\mathcal{I}_k}$ and $\langle x, y \rangle \in r_{kl}$, and so $x \in r_{kl}({E_1}^{\mathcal{I}_k})$. Therefore the bridge rule is satisfied by $\mathfrak{I}$.

Given an into-bridge rule $k : E_1 \xrightarrow{\sqsubseteq} l : E_2 \in \mathfrak{B}$, we ought to show that $r_{kl}(E_1{}^{\mathcal{I}_k}) \subseteq E_2{}^{\mathcal{I}_l}$. Let $x \in r_{kl}(E_1{}^{\mathcal{I}_k})$. Then there is $y \in V_k$ such that $l : x \in r_l(y)$. But since T is complete, it must be the case that $E_2 \in \mathcal{L}_l(x)$ and hence $x \in E_2{}^{\mathcal{I}_l}$. Therefore the bridge rule is satisfied by $\mathfrak{I}$.

The last thing in order to verify that $\mathfrak{I}$ is indeed a model of $\mathfrak{T}$ is to show that $\mathfrak{I}$ satisfies the transitivity requirement. We ought to show that for each $k, l, m \in I$, if $y \in r_{kl}(x)$ and $z \in r_{lm}(y)$ then also $z \in r_{km}(x)$. Given the two assumptions we know by construction that $l : y \in r_k(x)$ and $m : z \in r_l(y)$. That means that $x \in V_k$ was initialized after several "chained" applications of the $\xrightarrow{\sqsupseteq}$-rule starting from $y \in V_l$ and y in turn after several "chained" $\xrightarrow{\sqsupseteq}$-rule applications starting from $z \in V_m$. In that case however $r_l(y) \subseteq r_k(x)$. Hence $m : z \in r_k(x)$ and so $z \in r_{km}(x)$.

And thus $\mathfrak{I}$ is a distributed model of $\mathfrak{T}$ that satisfies the transparency condition and $C^{\mathcal{I}_i} \neq \emptyset$ – in other words, C is satisfiable in $\mathcal{T}_i$ with respect to $\mathfrak{T}$.

Completeness. Given $\mathfrak{T}$ with index set I, a concept C and $i \in I$ such that C is satisfiable in $\mathcal{T}_i$ with respect to $\mathfrak{T}$, we shall prove that the algorithm answers "C is satisfiable", if run on input $\mathfrak{T}$, C and i. Let $\mathfrak{I}$ be a distributed model of $\mathfrak{T}$ with $C^{\mathcal{I}_i} \neq \emptyset$, we know that one must exist. We will simulate the run of the algorithm. During the initialization T_i is set to $\langle \{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \emptyset \rangle$ and all the other T_j, $i \neq j$, are set to $\langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$. There is no clash in T. We will show by induction (on the number of tableau expansion steps) that after each tableau expansion step the completion tree T is expanded in such a way that no clash is introduced. In order to demonstrate this, we inductively construct an auxiliary mapping $\pi : \bigcup_{i \in I} V_i \to \bigcup_{i \in I} \Delta^{\mathcal{I}_i}$ to track the relation between $\mathfrak{I}$ and T. This mapping will keep the property $(*)$:

For each node $x \in V_i$, for each $i \in I$: if $C \in \mathcal{L}_i(x)$ then $\pi(x) \in C^{\mathcal{I}_i}$.

Let x be arbitrary member of $C^{\mathcal{I}_i}$, place $\pi(s_0) = x$. So, if a tableaux expansion rule is triggered in some $x \in V_i$ of the clash-free completion tree T (from induction hypothesis), we consider several cases, based on the kind of rule that was triggered:

1. $\sqcap$-rule is triggered in x because $E_1 \sqcap E_2 \in \mathcal{L}_i(x)$. Then by induction hypothesis $\pi(x) \in E_1 \sqcap E_2{}^{\mathcal{I}_i}$. In that case also $\pi(x) \in E_1{}^{\mathcal{I}_i}$ and $\pi(x) \in E_2{}^{\mathcal{I}_i}$, and hence the property $(*)$ is maintained after the algorithm adds E_1 and E_2 to $\mathcal{L}_i(x)$.
2. $\sqcup$-rule is triggered in x because $E_1 \sqcup E_2 \in \mathcal{L}_i(x)$. Then by induction hypothesis $\pi(x) \in E_1 \sqcup E_2{}^{\mathcal{I}_i}$. In that case however either $\pi(x) \in E_1{}^{\mathcal{I}_i}$ or $\pi(x) \in E_2{}^{\mathcal{I}_i}$. Without loss of generality let it be the case that $\pi(x) \in E_1{}^{\mathcal{I}_i}$. Without loss of generality we assume that the algorithm adds E_1 to $\mathcal{L}_i(x)$, as non-deterministic decision takes place. Hence the property $(*)$ is maintained after this step.
3. $\exists$-rule is triggered in x because it has an R-successor y, and $\exists R.E_1 \in \mathcal{L}_i(x)$. Then $\pi(x) \in \exists R.E_1{}^{\mathcal{I}_i}$ and so there is some $y' \in \Delta^{\mathcal{I}_i}$ such that $y' \in E_1{}^{\mathcal{I}_i}$ and $\langle \pi(x), y' \rangle \in R^{\mathcal{I}_i}$. When the algorithm generates new node y in this step we set $\pi(y) = y'$ and the property $(*)$ is maintained.
4. $\forall$-rule is triggered in x because it has an R-successor y, and $\forall R.E_1 \in \mathcal{L}_i(x)$, as a consequence E_1 is added to $\mathcal{L}_i(y)$. We know that y was created by an application of the $\exists$-rule and hence $\pi(y)$ is an R-successor of x in $\mathcal{I}_i$. But $\mathcal{I}_i$ is a model of $\mathcal{T}_i$ and hence $\pi(y) \in y^{\mathcal{I}_i}$ since from induction hypothesis we know that $\pi(x) \in \forall R.E_1{}^{\mathcal{I}_i}$. Hence $(*)$ is maintained after this step.
5. $\mathcal{T}$-rule is triggered in x, resulting to adding $\mathrm{nnf}(\neg E_1 \sqcup E_2)$ into $\mathcal{L}_i(x)$. Then $(*)$ is maintained as $\mathcal{I}_i$ is a model of $\mathcal{T}_i$ and so it holds that $\pi(x) \in (\mathrm{nnf}(\neg E_1 \sqcup E_2))^{\mathcal{I}_i}$.
6. $\xrightarrow{\sqsupseteq}$-rule is triggered in x because $E_2 \in \mathcal{L}_i(x)$ and $j : E_1 \xrightarrow{\sqsupseteq} i : E_2 \in \mathfrak{B}$. From induction hypothesis, $\pi(x) \in E_2{}^{\mathcal{I}_i}$, and hence $\pi(x) \in r_{ji}(y')$ for some $y' \in E_1{}^{\mathcal{I}_j}$. Set $\pi(y) = y'$ for the node y that is newly created in V_i as the result of this expansion step. The label $\mathcal{L}_j(y)$ has been set to E_1 but since $y' \in E_1{}^{\mathcal{I}_j}$ then $(*)$ is maintained.
7. $\xrightarrow{\sqsubseteq}$-rule is triggered in x because of $E_1 \in \mathcal{L}_i(x)$, $j : y \in r_i(x)$ and $i : E_1 \xrightarrow{\sqsubseteq} j : E_2 \in \mathfrak{B}$, resulting to adding E_2 into $\mathcal{L}_j(y)$. From induction hypothesis have $\pi(x) \in E_1{}^{\mathcal{I}_i}$. Observe that the definition of $\xrightarrow{\sqsupseteq}$-rule and the inductive construction of $\pi(\cdot)$ in previous steps also assure that $j : y \in r_i(x)$ implies $\pi(y) \in r_{ij}(\pi(x))$. This is because initialization of new $y \in V_j$ always follows incoming r-edge and r is transitive because of computational consistency. It follows that $\pi(y) \in E_2{}^{\mathcal{I}_j}$, since $\mathfrak{I}$ is a distributed model of $\mathfrak{T}$ and the bridge rule assures this. Hence $(*)$ is maintained even after this step.

We already know that the algorithm always terminates. Once this happens, it follows that T is now complete, because no rule is applicable, and clash-free, because the property $(*)$ is maintained all the way up to this point. Hence the algorithm answers "C is satisfiable" and hence the theorem. $\square$

The algorithm for the original DDL semantics of (Serafini et al. 2005) is obviously truly distributed in that sense that it supports a scenario in which several autonomous reasoning services run independently, one for each local ontology, and communicate by passing queries. While this is also the case for the newly introduced algorithm, it is not necessarily that obvious. In order ti clarify this, we provide a message protocol that handles $\xrightarrow{\sqsupseteq}$-rule and $\xrightarrow{\sqsubseteq}$-rule execution in a truly distributed fashion, and it also collects the information that the completion tree is complete within the reasoner that has initialized the computation. The algorithm employs three kinds of messages $\mathsf{querySat}(x, r, C)$, $\mathsf{answerSat}(x, C, \mathit{answer})$, and $\mathsf{pushConcept}(x, C)$. In the message based version, the algorithm starts by initializing the local completion tree T_j when asked for satisfiability of some j-local concept C with respect to $\mathfrak{T}$. More local completion trees are initialized during the runtime by passing a $\mathsf{querySat}(x, r, C)$ message every time the $\xrightarrow{\sqsupseteq}$-rule is fired. When $\xrightarrow{\sqsubseteq}$-rule is fired, the consequences of the into-bridge rule are propagated by passing a $\mathsf{pushConcept}(x, C)$ message. Once a local completion tree T_k is complete this is announced by passing a $\mathsf{answerSat}(x, C, \mathit{answer})$ message to the local reasoner that has triggered the computation in $\mathcal{T}_k$. Detailed specification of the protocol messages is given in Fig. 8.

6 Related Work

A distributed tableaux reasoning algorithm for the original semantics of DDL has been introduced in (Serafini & Tamilin 2005, Serafini et al. 2005) and implemented in system DRAGO. This algorithm is

querySat(x, r, C):
Send: if $G \in \mathcal{L}_j(x)$ for some $x \in V_j$, $i : C \stackrel{\sqsupseteq}{\longrightarrow} j : G \in \mathfrak{B}$, and x is not blocked, pass the message querySat$(x, r_j(x), C)$ to T_i.
Receive: upon receipt of a message querySat(x, r, C) from T_i, create new completion tree $\langle \{s_0\}, \emptyset, \{s_0 \mapsto \{C\}\}, \{s_0 \mapsto \{i : x\} \cup r\} \rangle$ within T_j and start the computation. Record the sent message in S_j.

answerSat$(x, C, answer)$:
Send: if the local completion tree for concept C within T_j, that has been initialized due the message querySat(x, r, C) previously received from T_i, is now complete, and there are no outgoing messages recorded in S_j, send the message answerSat$(x, C, answer)$ to $\mathcal{T}_i$ with $answer$ set to true if T_j is clash-free and to false otherwise.
Receive: upon receipt of a message answerSat$(x, C, answer)$ from $\mathcal{T}_i$, remove the message querySat(x, r, C) from S_j, that has been previously send to T_i. If $answer$ is false then add $E \sqcup \neg E$ to $\mathcal{L}_j(x)$ for some new concept name E.

pushConcept(x, C):
Send: if D has been added to $\mathcal{L}_j(x)$ in the previous step, for some $x \in V_j$, then for each $j : D \stackrel{\sqsubseteq}{\longrightarrow} i : H \in \mathfrak{B}$ s.t. $i : y \in r_j(x)$ send the message pushConcept(y, H) to T_i.
Receive: upon receipt of a message pushConcept(x, C) from $\mathcal{T}_i$, add C to $\mathcal{L}_j(x)$.

Figure 8: The message protocol for the newly introduced distributed tableaux algorithm. For each kind of message we specify when the message is sent from some local reasoning service T_j and also what happens when the message is received in some local reasoning service T_j (T_j is always the local reasoning service). An auxiliary data structure S_j is introduced in each T_j to track messages that have been sent in order to assure termination.

based on a fix-point characterization of the original DDL semantics. Given a set of bridge rules $\mathfrak{B}_{ij}$ between $\mathcal{T}_i$ and $\mathcal{T}_j$, define the operator $\mathfrak{B}_{ij}(\cdot)$ as follows: $\mathfrak{B}_{ij}(\mathcal{T}_i) = \{G \sqsubseteq \bigsqcup_{k=1}^{n} H_k \mid \mathcal{T}_i \models A \sqsubseteq \bigsqcup_{k=1}^{n} B_k, i : A \stackrel{\sqsupseteq}{\longrightarrow} j : G \in \mathfrak{B}_{ij}, i : B_k \stackrel{\sqsubseteq}{\longrightarrow} j : H_k \in \mathfrak{B}_{ij}, 1 \leq k \leq n\}$. Then the $\mathfrak{B}$ operator is defined: $\mathfrak{B}(\{\mathcal{T}_i\}_{i \in I}) = \{\mathcal{T}_i \cup \bigcup_{j \neq i} \mathfrak{B}_{ji}(\mathcal{T}_j)\}$. As given in (Serafini et al. 2005), the $\mathfrak{B}$ operator always has a fix-point $\mathfrak{B}^*(\mathfrak{T})$ when repeatedly applied on a distributed TBox $\mathfrak{T}$. Moreover, $\mathfrak{T} \models_\epsilon i : \phi$ if and only if the i-th component TBox of $\mathfrak{B}^*(\mathfrak{T})$ locally entails ϕ. Consequently, the standard $\mathcal{SHIQ}$ tableaux reasoning algorithm (Horrocks et al. 1999) is used with one additional bridge rule (see Fig. 9). The unsatisfiability check called by $\mathfrak{B}_{ij}$-rule is done by running the same algorithm again for the i-concept that is being checked. The algorithm assumes distributed TBoxes with acyclic bridge graph to insure termination. Compared to the algorithm introduced in this paper, this algorithm does not keep track of the domain relation r and does not in fact construct a true distributed tableau that would correspond to a particular distributed model of the input concept. Instead it cleverly uses the fix-point characterization and constructs multiple local tableaux in order to guarantee the existence of such a model. Most prominently, all consequences that are added to the triggering node if the unsatisfiability check is successful are estimated prior to the unsatisfiability check is executed. In contrast, in our approach computation in remote ontology is triggered by one message and consequences are announced back to the triggering node once they are computed, possibly by several independent messages. We believe that keeping precise track of all model structures, including the domain relation and dividing the communication into more fine-grained messages may provide better grounds for future optimization. Finally, it is worth noting that the newly introduced algorithm is easily adjusted to correspond to the original DDL semantics (by setting $r_i(y)$ to $\{j : x\}$ and not to $\{j : x\} \cup r_j(x)$ in the $\stackrel{\sqsupseteq}{\longrightarrow}$-rule).

$\mathfrak{B}_{ij}$-rule:
If $G \in \mathcal{L}_j(x)$, $i : A \stackrel{\sqsupseteq}{\longrightarrow} j : H \in \mathfrak{B}_{ij}$,
$BH \subseteq \{\langle B_k, H_k \rangle \mid i : B_k \stackrel{\sqsubseteq}{\longrightarrow} j : H_k \in \mathfrak{B}_{ij}\}$,
$B = \{B \mid \langle B, X \rangle \in BH\}$, $H = \{H \mid \langle Y, H \rangle \in BH\}$, $H \notin \mathcal{L}_j(x)$
and $A \sqcap \neg \bigsqcup B$ is unsatisfiable w.r.t $\mathfrak{T}$ in $\mathcal{T}_i$
then set $\mathcal{L}_j(x) = \mathcal{L}_j(x) \cup \{\bigsqcup H\}$.

Figure 9: The tableaux expansion rule used in the original DDL algorithm (Serafini & Tamilin 2005).

Another distributed ontology framework is $\mathcal{E}$-connections (Cuenca Grau et al. 2004). Here, inter-ontology roles (called links) are employed instead of concept mapping. A dedicated set ϵ_{ij} of symbols is used for (directed) links between $\mathcal{T}_i$ and $\mathcal{T}_j$. Links are then used in existential and value restrictions, and complex concepts involving links are formed (by instance the i-concept $\exists E.C$ is composed using the link $E \in \epsilon_{ij}$ and the j-concept C). Semantically, a combined interpretation consists of local interpretations $\mathcal{I}_i$ with non-empty domains $\Delta^{\mathcal{I}_i}$ and each link $E \in \epsilon_{ij}$ is interpreted by $E^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. Reasoning support for $\mathcal{E}$-connections is provided by extending the tableaux reasoning algorithm for $\mathcal{SHIF}(D)$. The two tableaux rules that are essential to handle links are depicted in Fig. 10. There is a notable correspondence between the $\exists_{link}$-rule and our $\stackrel{\sqsupseteq}{\longrightarrow}$-rule and also between the $\forall_{link}$-rule and our $\stackrel{\sqsubseteq}{\longrightarrow}$-rule. This is not that surprising, given the known correspondence between DDL and $\mathcal{E}$-connections (Kutz et al. 2004). Given the nature of E-connections the $\forall_{link}$-rule works exactly the other way around, compared to our $\stackrel{\sqsubseteq}{\longrightarrow}$-rule, and in this respect the instantiation of a j-tree by one application of the $\exists_{link}$-rule and possibly multiple applications of the $\forall_{link}$-rule resembles an unsatisfiability check call from the $\mathfrak{B}_{ij}$-rule of the original DDL algorithm. Remarkably, the exact mechanism how this happens within the $\mathcal{E}$-connections algorithm seems to be more transparent and prone to optimization. The algorithm introduced in this paper in addition keeps track on the domain relation, which is needed to handle subsumption propagation along chains of bridge rules. There is no domain relation within $\mathcal{E}$-connections, and hence no need for such a feature.

$\exists_{link}$-rule:
If $\exists E.C \in \mathcal{L}_i(x)$, $E \in \epsilon_{ij}$, x is not blocked and x has no E-successor y with $\mathcal{L}(\langle x, y, \rangle) = \{E\}$,
then create a new E-successor (a j-node) y of x with $\mathcal{L}_j(y) = \{C\}$. (The new j-tree rooted in y will not be expanded until no more rules apply to the i-tree.)

$\forall_{link}$-rule:
If $\forall E.C \in \mathcal{L}_i(x)$, $E \in \epsilon_{ij}$, x is not blocked and there is an E-successor y of x such that $C \notin \mathcal{L}_j(y)$
then set $\mathcal{L}_j(y)$ to $\mathcal{L}_j(y) \cup \{C\}$.

Figure 10: The tableaux expansion rules of the algorithm for $\mathcal{E}$-connections (Cuenca Grau et al. 2004).

Yet another point of view on distributed ontologies is Package-based description logics, or P-DL (Bao et al. 2006). In P-DL, the intuition of importing is pursued: every concept name belongs to some local ontology, but can be also imported to and used by other ontologies in the system. The P-DL semantics uses domain relations. In contrast to other approaches, only one-to-one domain relations are allowed and also compositional consistency (that we have borrowed and applied on DDL) is required. Local domains in P-DL semantics are viewed as partially overlapping, and not disjoint. As a result, some of the problems known for DDL, such as restricted knowledge propagation between remote ontologies, that we also address in this paper, are not an issue for P-DL according to (Bao et al. 2006). On the other hand, in P-DL reasoning over a distributed ontology is always equivalent to reasoning over the union of local ontologies, which is not a goal of DDL. The distributed tableaux algorithm for P-DL is introduced in (Bao et al. 2006). It uses $\mathcal{ALC}$ as local language and requires acyclic importing relation. It uses message-based protocol, similar to the ours, and keeps track of the domain relation. Since imported concepts may appear anywhere in the importing ontology, messages are invoked directly from $\mathcal{ALC}$-tableaux rules.

A completely different approach to distributed ontology reasoning is taken in (Schlicht & Stuckenschmidt 2008), where distributed reasoning algorithm based on resolution techniques is introduced. Yet another distributed ontology framework called Integrated Distributed Description Logics (IDDL) is introduced in (Zimmermann 2007), where also an incomplete decision procedure is outlined.

7 Conclusion and Future Work

In a recent paper (Homola 2008) we have proposed an adjusted semantics for DDL, dubbed "DDL under compositional consistency", that features improved subsumption propagation between remote ontologies. More specifically, subsumption propagates along chains of bridge rules that span throughout multiple ontologies. In this paper we take steps forward in order to develop a distributed tableaux reasoning algorithm that would decide satisfiability of concepts and subsumption with respect to the adjusted semantics. The algorithm that is introduced in this paper handles chaining onto-bridge rules correctly, but it is unable to cope with chaining into-bridge rules. We provide precise characterization of the semantics that the algorithm actually implements.

As in the case of the tableaux algorithm that is known for the original semantics (Serafini et al. 2005), the newly introduced algorithm is also truly distributed and it permits the scenario where every local ontology is governed by an autonomous reasoning service and these services communicate by passing queries. The local reasoner that has started the computation collects all the answers and makes the decision at the end. To demonstrate this fact more clearly, we have provided a message-based protocol for the algorithm. Besides the fact that each of these algorithms works under a different semantics, the main distinguishing feature of the newly introduced algorithm is that communication between local reasoners is divided into multiple messages. Computation is sparked in a remote reasoner at some point and both reasoners continue to run independently. If subsumption propagation is proved by the remote reasoner, the local reasoner is acknowledged by a message. This possibly repeats several times, if subsumption is proved between different pairs of concepts. We believe that such behaviour may serve as a base for more fine-grained optimization in future.

The two most prominent open issues put forward by this paper are: extending the algorithm so that it would handle the DDL under compositional consistency semantics in full extent, that is, dealing with chaining into-bridge rules; and, extending the algorithm towards more expressive DL, since the current version only supports $\mathcal{ALC}$ as the local representation language and requires acyclic concept mapping. Computational handling of distributed knowledge bases in which the concept mapping is not necessarily acyclic is an interesting problem, which is to our best knowledge still unresolved also for the original DDL framework. As is the problem of extending the DDL framework with nominals. We would like to address these issues in the near future. Remaining research problems that are closely related to this work include practical evaluation of the algorithm by implementation, complexity analysis for the decision problems, and combining, within a unified framework, the current approach with that of (Homola 2007), which has addressed the problem of interaction between bridge-rules in DDL.

Acknowledgement

Support from the VEGA project of Slovak Ministry of Education and Slovak Academy of Sciences no. 1/0173/03 and from the grant GUK no. UK/365/2008 awarded by Comenius University is gratefully acknowledged. Special thanks to Ján Kľuka for advanced LaTeX symbol hacking.

References

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

Bao, J., Caragea, D. & Honavar, V. G. (2006), A distributed tableau algorithm for package-based description logics, *in* 'Procs. of CRR 2006'.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001), 'The semantic web', *Scientific American* **284**(5), 34–43.

Borgida, A. & Serafini, L. (2003), 'Distributed description logics: Assimilating information from peer sources', *Journal of Data Semantics* **1**.

Cuenca Grau, B., Parsia, B. & Sirin, E. (2004), Working with multiple ontologies on the semantic web., *in* 'Procs. of ISWC2004', Vol. 3298 of *LNCS*, Springer.

Homola, M. (2007), Distributed description logics revisited, *in* 'Procs. of DL-2007', Vol. 250 of *CEUR-WS*.

Homola, M. (2008), Subsumption propagation between remote ontologies in distributed description logic, *in* 'Procs. of DL2008', Vol. 353 of *CEUR-WS*.

Horrocks, I., Sattler, U. & Tobies, S. (1999), Practical reasoning for expressive description logics, *in* 'Procs. of LPAR'99', number 1705 *in* 'LNAI', Springer, pp. 161–180.

Kutz, O., Lutz, C., Wolter, F. & Zakharyaschev, M. (2004), '$\mathcal{E}$-connections of abstract description systems', *Artificial Intelligence* **156**(1), 1–73.

Schlicht, A. & Stuckenschmidt, H. (2008), Distributed resolution for alc, *in* 'Procs. of DL2008', Vol. 353 of *CEUR-WS*.

Serafini, L., Borgida, A. & Tamilin, A. (2005), Aspects of distributed and modular ontology reasoning, *in* 'Procs. of IJCAI'05', pp. 570–575.

Serafini, L. & Tamilin, A. (2004), Local tableaux for reasoning in distributed description logics, *in* 'Procs. of DL'04', CEUR-WS.

Serafini, L. & Tamilin, A. (2005), DRAGO: Distributed reasoning architecture for the semantic web, *in* 'Procs. of ESWC'05', Vol. 3532 of *LNCS*, Springer-Verlag, pp. 361–376.

Zimmermann, A. (2007), Integrated distributed description logics, *in* 'Procs. of DL-2007', Vol. 250 of *CEUR-WS*.

Enhancing Subjective Ontologies with Social Tagging Systems

Dennis Hooijmaijers[1] **Markus Stumptner**[1]

[1] Advanced Computing Research Centre
University of South Australia,
Mawson Lakes Blvd, Mawson Lakes, South Australia 5095,
Email: {dennis,mst}@cs.unisa.edu.au

Abstract

Social computing websites provide a framework for users to submit (and discuss) content, while often providing a user created categorization system, dubbed folksonomies. Additionally users may have the ability to specify friends (and foes) within the social network, populated by registered users of the social website. Integrating folksonomies with ontologies provides the ability to capture the semantic relationships between terms. This allows for greater reasoning to refine searches.

Most categorisation for social computing is currently achieved by a meta-tagging system dubbed folksonomies, which fails to take into consideration the semantics that occur between terms. By integrating social networks with an ontology we aim to provide mechanisms to capture agreement in opinions and to capture and enhance the semantics of the social network, allowing for greater reasoning to refine searches.

Keywords: Ontology, Subjective Logic, Uncertainty, Folksonomy

1 Introduction

Knowledge is subjective and this can be seen by the many and varied uses of terminology in on-line communities. User generated content has expanded from comments on forums to include, reviews, synopsis, blogs, and multimedia. Where originally the website provider would post the content for discussion, current sites provide frameworks for users to upload content, or synopsis of on-line content, and discussion boards to allow for interaction between users. Categorizing content allows users to find objects of interest within a particular domain.

Some sites (e.g. Digg[1], Slashdot[2]) provide content areas for a user to select when uploading, while others augment this with user defined tags (meta-data), such as Flickr[3] and YouTube[4]. This allows the users to select, or create, appropriate descriptive terms for their content. These user defined tags are collectively known as folksonomies (Al-Khalifa & Davis 2006), and can provide insight into popular terminology within the domain of a community.

[1]http://digg.com
[2]http://slashdot.org
[3]http://www.flickr.com
[4]http://www.youtube.com

To further assist users to sort through on-line submissions a user-based rating system is applied. This allows users to find the most popular submissions providing a collaborative voting mechanism. These rating systems can be as simple as a 5-star rating system where each user provides a rating between 0 and 5 (YouTube, Flickr), or a positive (*thumbs up*) and negative (*thumbs down*) rating approach where users state whether they like or dislike an object (Digg). Additionally the ratings can be categorised (Slashdot, Digg[5] with additional terms (*informative*, *insightful*, *troll* etc.) to assist in clarity of ratings.

These frameworks also allow users to specify other users as friends (and foes), allowing for users to refine the popularity ratings to find submissions that their friends have found popular.

RIPOSTE is a knowledge management framework, that allows for user ratings to be captured within context. It is based on OWL DL, as defined by the w3c[6], a knowledge representation language for the semantic web and Subjective Logic(Jøsang 2002), an extension of probabilistic logic. RIPOSTE provides a meta-ontology which captures authors, and their opinions with the ontological resources they provide. The combined ontology and meta-ontology is a *subjective ontology.* RIPOSTE provides mechanisms to capture, integrate, manipulate and reason over subjective ontologies. This work extends the RIPOSTE framework to provide additional functionality for capturing social networks and to integrate folksonomies with the subjective ontologies.

Adding semantic value to folksonomies (Angeletou et al. 2007, van Damme et al. 2007, Szomszor et al. 2007) has been identified as a solution for improving *recall* for queries (i.e. finding objects related to Sydney when querying for cities). The identification of relations is achieved by using background information from multiple sources. Most current techniques for integration use on-line lexicons (e.g. WordNet[7]) and Semantic Web technologies. Lexicons are used for discovering homonyms, meronyms and synonyms, while the Semantic Web for richer semantic relationships (i.e. object properties, disjoint, hierarchical). Within the Semantic Web relationships between terms may exist within a single ontology or multiple ontologies (Angeletou et al. 2007). Our approach also uses online lexicons and the Semantic Web, although by including probabilistic approach we can capture all ambiguities of a term within the resultant ontology.

Disambiguation of the terminology, can then be discovered by dynamic filtering for a specific query (Hooijmaijers & Stumptner 2006). Additionally by including the social network within the ontology the ability to capture the author's opinion for specific con-

[5]Digg only provides categorisation for negative ratings. i.e. the object is considered *lame* or a *duplicate* etc.
[6]http://www.w3.org/2004/OWL/
[7]http://wordnet.princeton.edu/

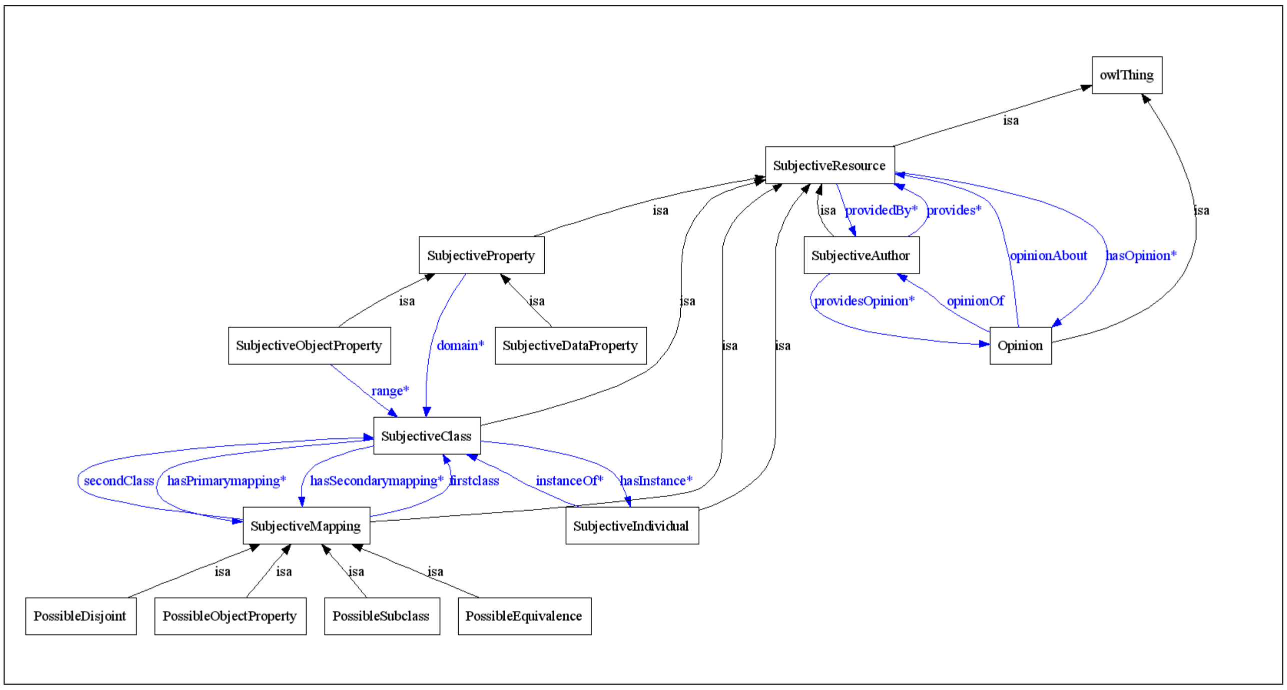


Figure 1: The RIPOSTE Meta-Ontology

cepts and easily discover the group of authors that agree and/or disagree with that opinion.

2 RIPOSTE

RIPOSTE (Hooijmaijers & Stumptner 2006) is a knowledge management framework, based on OWL DL and subjective logic, that provides mechanisms for capturing, integrating and evolving ontologies with the *authors* that provide the ontological resources. The term author is used to differentiate between members of a community who provide resources and the users who only utilise content. We allow both authors and users to provide opinions and a user will become an author when they provide any resource. It uses a meta-ontology to provide additional information about each resource. The resources are subjective in that each author may have a different opinion on how a resource should be used and this allows for filtering and reasoning based on similar opinions. The subjective nature of the knowledge is captured using subjective logic, an extension of probabilistic logic (Jøsang 2002). This allows for opinions to be compared to capture and manipulate the probability that two authors agree (or disagree) on the usage of a term.

To assist with capturing ontology mappings RIPOSTE allows for the capture of the mappings between ontologies by creating a set of individuals of SubjectiveMapping. This allows for opinions to be formed about the mapping and for the author (which can be an automated mapping agent) to be captured. This can be used to generate a final integrated ontology based on multiple mappings (Hooijmaijers & Stumptner 2008).

2.1 Subjective Logic

Subjective Logic is an extension to probabilistic logic that represents an author's (A) beliefs about an object (x) as *opinions* (w_x^A). An opinion is the combination of belief (b_x), disbelief (d_x) and uncertainty (u_x) in a given resource (x), where

$$b_x + d_x + u_x = 1$$

and the opinion,

$$(w_x^A) = (b_x, d_x, u_x, a_x)$$

where a_x represents the size of the state space, atomicity, from which x is taken, Figure 2 and is used as the *a priori* probability in the absence of evidence. The expectation value represents the likelihood of a positive belief given no evidence and is calculated by:

$$E(x) = b_x + a_x u_x \tag{1}$$

For resources within the ontology we use the sense count from WordNet (e.g. Figure 6) for the opinions, which represent the possible choices for use of a keyword for the state space. For classes and tags we use a_x to work out the possible relationships which for *unrelated* and *equivalent* classes is given by:

$$a_x = \frac{1}{senseCount + 1} \tag{2}$$

For trust between authors we use the state space of {trust, distrust} $a_x = 0.5$.

In this work we use the subjective logic operators, *Consensus* and *Discount*, as defined by Jøsang (Jøsang 2002). Consensus ($\oplus$) is used to, fairly and equally, combine two possibly conflicting opinions about a given resource and Discount ($\otimes$) is used to capture reputation (trust transitivity).

Consensus (Jøsang 2002) Let $w_x^P = (b_x^P, d_x^P, u_x^P, a_x^P)$ and $w_x^Q = (b_x^Q, d_x^Q, u_x^Q, a_x^Q)$ be opinions respectively held by P and Q about the same proposition x. Let $w_x^{PQ} = (b_x^{PQ}, d_x^{PQ}, u_x^{PQ}, a_x^{PQ})$ be the opinion such that:

1. $b_x^{P,Q} = (b_x^P u_x^Q + b_x^Q u_x^P)/k$
2. $u_x^{P,Q} = (u_x^P u_x^Q)/k$
3. $d_x^{P,Q} = (d_x^P u_x^Q + d_x^Q u_x^P)/k$
4. $a_x^{P,Q} = \frac{(a_x^P + a_x^Q - (a_x^P + a_x^Q) u_x^P u_x^Q)}{u_x^P + u_x^Q - 2 u_x^P u_x^Q}$

where $k = u_x^P + u_x^Q - u_x^P u_x^Q$ such that $k \neq 0$ and $a_x^{P,Q} = (a_x^P + a_x^Q)/2$ when $a_x^P, a_x^Q = 1$. Then w_x^{PQ} is called the consensus between w_x^P and w_x^Q, representing an imaginary agent $[P, Q]$'s opinion about x, as if she represented both P and Q. By using the symbol '$\oplus$' to designate this operator, we define $w_x^{PQ} \equiv w_Q^P \oplus w_x^Q$.

Discount (Jøsang 2002) Let P and Q be two agents where $w_Q^P = (b_Q^P, d_Q^P,\ u_Q^P, a_Q^P)$ is P's opinion about Q's advice, and let x be the proposition where $w_x^Q = (b_x^Q, d_x^Q, u_x^Q, a_x^Q)$ is Q's opinion about x expressed in an advice to P. Let $w_x^{PQ} = (b_x^{PQ}, d_x^{PQ}, u_x^{PQ}, a_x^{PQ})$ be the opinion such that:

1. $b_x^{PQ} = b_Q^P b_x^Q$
2. $u_x^{PQ} = d_Q^P + u_Q^P + b_Q^P u_x^Q$
3. $d_x^{PQ} = b_Q^P d_x^Q$
4. $a_x^{PQ} = a_x^Q$

By using the symbol '$\otimes$' to designate this operator, we define $w_x^{PQ} \equiv w_Q^P \otimes w_x^Q$.

Given two authors, Allan and Bob, if Bob uses the term 'Australia' to describe an object (x) from Figure 4, and *Allan* trusts that *Bob* is right 66% of the time then:

$$w_{Bob}^{Allan} = (0.66, 0.34, 0, 0.5)$$

and Bob believes he is correct:

$$w_x^{Bob} = (1, 0, 0, 0.17)$$

Then Allan's opinion that Bob has tagged x correctly is calculated using discount $\otimes$:

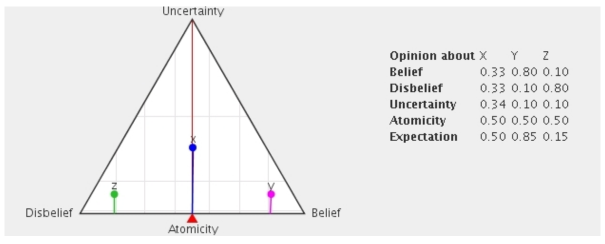
$$w_x^{Allan} = (0.66, 0, 0.34, 0.17)$$

If a third author Carl also provides the tag 'Australia' for x ($w_x^{Carl} = (1, 0, 0, 0.17)$) and Allan trusts Carl 50% using discount gives:

$$w_x^{Allan} = (0.5, 0, 0.5, 0.17)$$

This will be combined using consensus ($\oplus$):

$$w_x^{Allan} = (0.75, 0, 0.25, 0.17)$$



Opinion about	X	Y	Z
Belief	0.33	0.80	0.10
Disbelief	0.33	0.10	0.80
Uncertainty	0.34	0.10	0.10
Atomicity	0.50	0.50	0.50
Expectation	0.50	0.85	0.15

Figure 2: Subjective Logic opinion triangle

2.2 Meta-ontology

To capture the additional information about subjectivity and the author with a resource we have introduced a meta-ontology layer (see Figure 1). RIPOSTE is a framework that provides the ability to capture author information with the meta-data provided. It provides mechanisms for integrating author meta-data and the ability to filter the resultant ontology to allow for queries based on different opinions. The key concept of RIPOSTE is that all resources of an ontology are subjective to the author(s) that provides it. RIPOSTE meta-ontology, shown in Figure 1 provides the framework for capturing the author's opinion and the relationship between authors and the resources they provide. Each SubjectiveResource has an opinion, a Boolean filter isAccepted and an object property relating the resource to the providing authors. To annotate an ontology, O for Riposte it is necessary to make all classes, $C \in O$ an instance of SubjectiveClass. Additionally all $C \in O$ that have only *owlThing* as a super-class become a *subClass* of SubjectiveClass, as shown in Figure 3. By creating each class as an instance of SubjectiveClass we can apply the credibility attribute, filter and relationship to the author to the class, while the sub-classing allows the additional credibility information to be provided to the individual instances.

3 Classification

Classification can be performed by either creating an explicit semantic representation, such as an ontology, or by allowing the semantics to evolve from authors creating their own tags. The tags from all authors in a community are collated into a folksonomy.

An ontology can be described as a collection of resources that explicitly and formally conceptualise a domain model (Guarino 1997). Figure 3 shows an example class structure of an ontology for photographs taken in locations, the ellipses represent the classes in the ontology and the arcs the semantic relations between the classes. The white box (SubjectiveClass) shows the replaced owlThing. For simplicity axioms and instances have not been shown.

Definition An OWL ontology, O, is considered to be a 4-tuple of resources $< C, S, I, \chi >$ where C is a set of *classes*, S is a set of *semantic relations* which relate two or more classes, I is a set of *instances* of a class, and χ is a set of *axioms.*

An ontology in this work is a subjective view of a domain which we call a **subjective ontology**. A subjective ontology provides credibility ratings for all resources and links all resources within the ontology to the authors that provided them. This provides a mechanism to manage an integrated ontology by each author that has contributed resources.

Definition Let an author A_i be a provider of a resource R for an ontology $R \in O^D$ then their opinion is given by $\omega_R^{A_i}$. We define a subjective ontology as, O_{A_i}, is an extension of an ontology that relates an author, A_i, and their opinion $\omega_R^{A_i}$ to the provided ontological resource R. $(\omega_R^{A_i}, A_i, R)$

A folksonomy is a term used to describe a tag-based system that has evolved over time through the direct input by the user base (Al-Khalifa & Davis 2006). Flickr, a photograph sharing site, and del.icio.us, a social bookmarking site, are examples of implemented folksonomies. A folksonomy is created by users supplying objects and selecting, or creating, tags that describe the object. Tags are assumed to represent a

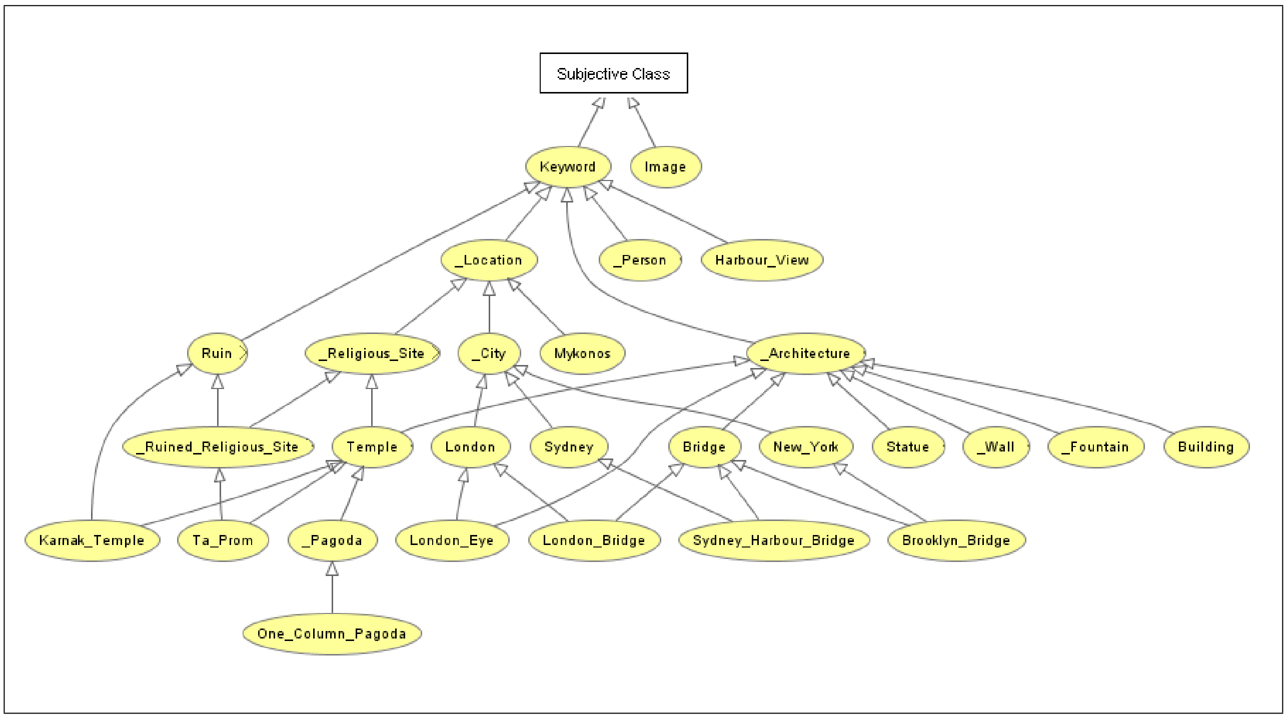


Figure 3: Section of example ontology created by Domain expert

collection of similar objects. This classification allows for the creation of a vocabulary to represent all tags that an author uses.

In this work we capture the importance of tags as they relate to both objects and authors to allow for the calculation of the probability that a tag is useful in another author's view of a domain.

Definition We define T_{ij} to be the set of tags $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, applied to an object, O_j, as a description provided by an author A_i.

By combining all of an author's tags we capture the vocabulary of that author.

Definition A vocabulary, V_i of an author, A_i is the combination of all tags used by that author for all objects:
$V_i = T_{i1} \cup T_{i2} \cup \ldots \cup T_{ij} \cup \ldots \cup T_{ik}$

We assume that all objects within the environment have at least one tag, $\exists i \forall j T_{ij} \neq \{\varnothing\}$, thus a vocabulary can be used to access all of an authors categorised objects.

Folksonomies have difficulty when a term's meaning in another domain is different. By lacking the ability to apply a context to a tag there is no ability for an application to differentiate the content on the tags alone. Redundant tagging or use of synonyms creates an additional difficulty that must be overcome. An effective tagging system could require the integration of a thesaurus (such as WordNet) to ensure that all appropriate synonyms are returned as the result of a search query. Spelling errors and use of plural form also cause difficulty and need to be addressed by the framework.

The main difficulty focused on in this work is the use of incorrect tags (tagging a photo about cities with rural landscapes), or conflicting tags (tags that can be classified as disjoint). Another problem is the lack of attribute information. This would prevent querying a folksonomy for any results based on an attribute (e.g. cities with a population $\geq$ 3 million).

4 Social Networking

The SubjectiveAuthor concept in Figure 1 is a sub-class of SubjectiveResource and inherits the attributes and relationships. This allows an author to be rated by credibility, to be accepted or rejected based on the isAccepted property, and to provide other authors as resources within their ontology. Additionally we extended the RIPOSTE framework with the additional semantic relationships between authors labeled 'friend' and 'foe'. These additions allow for a greater ability to replicate existing social networks such as a *Web of Trust* (Guha 2004), *Friend of a Friend* (FOAF) (Dumbill 2002), or an existing social computing site, such as Digg, or Slashdot. Initially we assume that an author is categorised as a friend if there is a high level of agreement on a previous topic and a foe if actively sabotaging the current author on a previous topic of discussion.

4.1 Rating System

In a distributed environment, such as the Semantic Web, any author can supply information. It can be difficult to discover credible information and reliable authors. To assist in this discovery social modelling approaches (Resnick et al. 2000, Dumbill 2002, Guha 2004) rate users and their information providing additional feedback to help establish trust.

Numerous trust models have been proposed to rate or quantify the credibility of authors and the information they provide. In this work trust is defined as:

> Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends. (Jøsang et al. 2005)

Trust is often calculated using social modelling to create networks that examine reputation (Resnick et al. 2000) captured in a directed graph with nodes (authors) and arcs (trust relationships) (Dumbill 2002). Additionally, when creating a social model, it is necessary to model trust and distrust between objects (ontologies, tags or instances), users, reviews, and ratings of reviews (Guha 2004). 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 also alter rating within the model to reflect their modified reputation. This is often referred to as a *Web of Trust*(Guha 2004). In this work trust is represented as an opinion. That is if an author (A_1) is said to trust another author, or object, (x) then A_1 has an opinion $\omega_x^{A_1}$ about the other author.

Tag clouds, shown in Figure 5 and Figure 8 are used to show the popularity (Γ) of a tag. A tag cloud can be separated into time periods (Δ)(i.e. last 24 hrs or week) and is generated by using size to represent the number of authors $(|A|)$ who use a tag (α). We calculate the popularity by the percentage of authors who use a given tag within a given time period:

$$\Gamma_\Delta(\alpha) = \Delta\frac{|A^\alpha|}{|A|}\ where\ \alpha \in V_i \tag{3}$$

where A^α is the set of authors that have $\alpha \in V$.

By introducing a time period, Δ, we are able to capture the evolutionary aspect of tags within the environment.

Content rating $\Gamma(\alpha)$ is based on user opinions and in its simplest form is a popularity measure, where users vote whether they like, or dislike, a submission and an aggregated value is applied to the content. This value is often used calculate popularity, which can provide a prominent position on the website (e.g. Digg).

We use the ratings of the tag cloud to provide a value for the expectation of a positive belief of the tag:

$$E(\alpha) = \frac{\Gamma(\alpha)}{max(\Gamma(\alpha))} \tag{4}$$

where $\max(\Gamma(\alpha))$ is the highest rated content. This can be combined with a_x calculated in equation 2, and an initial assumption that with the absence of evidence the opinions for all tags are completely uncertain (i.e. $u_x = 1$). By combining equations 1, 4 we calculate an initial belief b_x:

$$b_\alpha = \frac{\Gamma(\alpha)}{max(\Gamma(\alpha))} - a_\alpha \tag{5}$$

As shown in the tag cloud in figure 5 tag 'tower' has a popularity $\Gamma(tower) = 5$, while $\Gamma(building) = 10$,

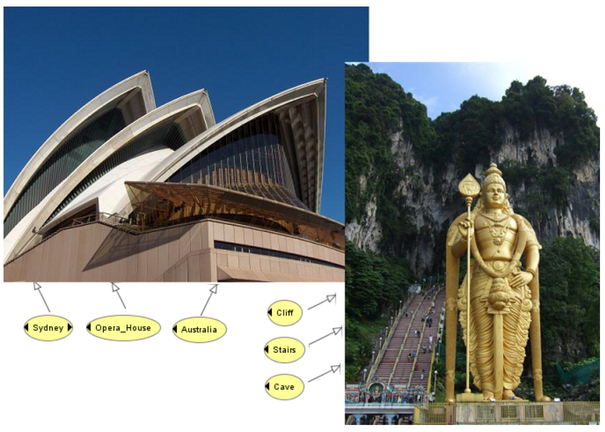


Figure 4: Example of user supplied content with user defined tags

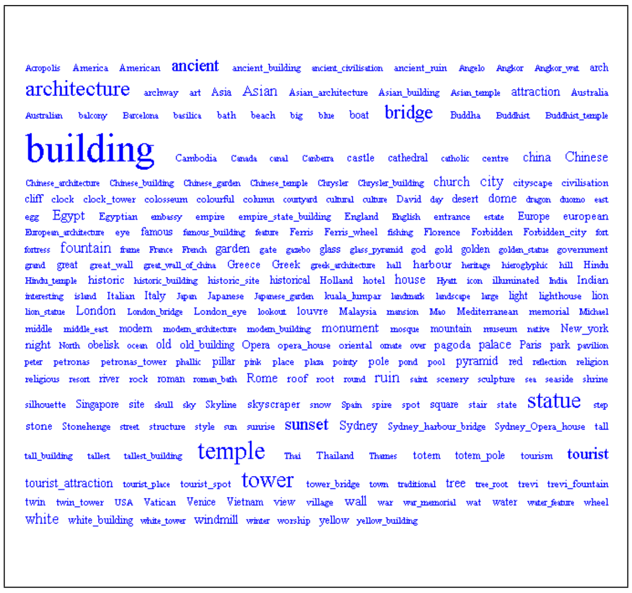

Acropolis America American ancient ancient_building ancient_civilisation ancient_ruin Angelo Angkor Angkor_wat arch
architecture archway art Asia Asian Asian_architecture Asian_building Asian_temple attraction Australia
Australian balcony Barcelona basilica bath beach big blue boat bridge Buddha Buddhist Buddhist_temple
building Cambodia Canada canal Canberra castle cathedral catholic centre china Chinese
Chinese_architecture Chinese_building Chinese_garden Chinese_temple Chrysler Chrysler_building church city cityscape civilisation
cliff clock clock_tower colosseum colourful column courtyard cultural culture David day desert dome dragon duomo east
egg Egypt Egyptian embassy empire empire_state_building England English entrance estate Europe european
European_architecture eye famous famous_building feature Ferris Ferris_wheel fishing Florence Forbidden Forbidden_city fort
fortress fountain frame France French garden gate gazebo glass glass_pyramid god gold golden golden_statue government
grand great great_wall great_wall_of_china Greece Greek greek_architecture hall harbour heritage hieroglyphic hill Hindu
Hindu_temple historic historic_building historic_site historical Holland hotel house Hyatt icon illuminated India Indian
interesting island Italian Italy Japan Japanese Japanese_garden kuala_lumpar landmark landscape large light lighthouse lion
lion_statue London London_bridge London_eye lookout louvre Malaysia mansion Mao Mediterranean memorial Michael
middle middle_east modern modern_architecture modern_building monument mosque mountain museum native New_york
night North obelisk ocean old old_building Opera opera_house oriental ornate over pagoda palace Paris park pavilion
peter petronas petronas_tower phallic pillar pink place plaza pointy pole pond pool pyramid red reflection religion
religious resort river rock roman roman_bath Rome roof root round ruin saint scenery sculpture sea seaside shrine
silhouette Singapore site skull sky Skyline skyscraper snow Spain spire spot square stair state statue step
stone Stonehenge street structure style sun sunrise sunset Sydney Sydney_harbour_bridge Sydney_Opera_house tall
tall_building tallest tallest_building temple Thai Thailand Thames totem totem_pole tourism tourist
tourist_attraction tourist_place tourist_spot tower tower_bridge town traditional tree tree_root trevi trevi_fountain
twin twin_tower USA Vatican Venice Vietnam view village wall war war_memorial wat water water_feature wheel
white white_building white_tower windmill winter worship yellow yellow_building

Figure 5: Resultant Tag-Cloud from study with a threshold of 5

while figure 6 shows 'tower' to have a $senseCount = 3$. Assuming 'tower' is unrelated to existing ontology classes we use equation 2 to calculate $a_{tower} = 0.25$. The initial belief for 'tower' is calculated by:

$$b_{tower} = \frac{5}{10} - 0.25 = 0.25$$

This becomes evidence reducing our uncertainty from 1 (initial uncertainty) to 0.75 and disbelief = 0.

4.1.1 Combining Ratings

One of the key issues of social computing is combining ratings. In traditional sites (Digg, Slashdot) a single positive value is provided for all content that friends rate as good, while all comments by foes are filtered out. To provide the ability to see both opinions it is necessary to combine ratings, using consensus $(\oplus)$ and discount$(\otimes)$, from both friends (ϕ) and foes (ψ), of the deciding author (D), to reflect that a foe may be correct. To capture the opinions of friends we use:

$$\omega_x^{D\phi} = (\omega_{A_1}^D \otimes \omega_x^{A_1}) \oplus \cdots \oplus (\omega_{A_n}^D \otimes \omega_x^{A_n}) \forall A \in \phi$$

and foes:

$$\omega_x^{D\psi} = (\omega_{A_1}^D \otimes \omega_x^{A_1}) \oplus \cdots \oplus (\omega_{A_n}^D \otimes \omega_x^{A_n}) \forall A \in \psi$$

This is combined with a weighted opinion ω_ϕ^W and ω_ψ^W which allows for friends to be held in higher $(\omega_\phi^W > \omega_\psi^W)$,or lower $(\omega_\phi^W < \omega_\psi^W)$, regard than foes:

$$\omega_x^D = (\omega_\phi^W \otimes \omega_x^{D\phi}) \oplus (\omega_\psi^W \otimes \omega_x^{D\psi})$$

Given the 3 authors Allan, Bob and Carl introduced earlier, let Allan and Bob be friends $(Bob \in \phi_{allan})$ and Allan and Carl be foes $(Carl \in \psi_{Allan})$. Then using an arbitrary weight for friends and foes of:

$$\omega_\phi^W = (0.8, 0.2, 0, 0.5)$$
$$\omega_\psi^W = (0.2, 0.8, 0, 0.5)$$

showing that we believe our friends 80% of the time they provide us with a tag and disbelieve them 20%, while foes are only believed 20% and disbelieved 80%. Then for the keyword 'Australia' the opinion is calculated as:

$$\begin{aligned}\omega^{Allan}_{Aus} &= (0.8, 0.2, 0, 0.5) \otimes (0.66, 0, 0.34, 0.17)\\ &\oplus (0.2, 0.8, 0, 0.5) \otimes (0.5, 0, 0.5, 0.17)\\ &= (0.55, 0, 0.45, 0.17)\end{aligned}$$

5 Integrating Folksonomies and Ontologies

The main advantage of integrating a folksonomy is to utilise the expressiveness of the ontology to improve querying (Angeletou et al. 2007) (i.e. querying for photos of cities should return images tagged as being photographs of Barcelona). If we query the vocabulary of Figure 4 for images of cities in China we would get no result. Once a mapping to the ontology, shown in Figure 3 is performed, the resultant images from User1 will be returned. To integrate folksonomy to ontology it is necessary to convert them to the same format. Since this work uses OWL DL, which is a richer language than the meta-tag folksonomy, the folksonomy is converted to OWL DL without loss by encoding the folksonomy.

5.1 Encoding Folksonomies

A folksonomy can be defined as an ontology that has no semantic relations, $S = \{\varnothing\}$, and no axioms $\chi = \{\varnothing\}$. This allows the folksonomy to be encoded in OWL DL as an ontology, O', by creating a concept, C_α, $\forall\alpha \in V_i$, such that $C_{alpha} \sqsubset owlThing$. Finally each object, O_j, described by the vocabulary, V_i should be encoded as an individual, Instance(O_j, C_{alpha}), $\forall\alpha \in T_ij$.

This conversion allows for the integration of the folksonomy with an ontology. The difficulty for integration is that the folksonomy has no semantic relations allowing for no graph matching (Noy & Musen 2000). This limits the integration algorithm to pattern matching on the labels (Noy & Musen 2000, McGuinness et al. 2000) or by examining external sources (Sabou et al. 2006).

Examining existing ontologies for semantics between concepts can assist in discovering mappings (Sabou et al. 2006). The Semantic Web provides a semantically rich environment that can be used as background knowledge. By utilising existing ontologies that contain a concept from each of the ontologies to be merged, it is possible to derive a mapping. To deal with contradictions provided by multiple background ontologies that contain the two concepts any multiple results from a Semantic Web search engine (such as Swoogle[8]) need to be compared to derive the best mapping. Currently this comparison is achieved by ranking the results by the authors reliability if it exists or by selecting the ontology with the shortest path between the concepts. An additional necessity of integration is the need to add an additional resource to describe semantic relationships between two concepts.

5.2 Integration Process

Our integration process involves finding similarities between tags in a folksonomy vocabulary and classes in an ontology. The integration is subjective based on the opinions of the user performing the operation.

1. If $A_i \notin O$ we add A_i to O and then examine syntactic equivalence for each $C \in O$ and for each $\alpha \in V_i$:

[8]http://swoogle.umbc.edu/

2. If $syntax(\alpha) = syntax(C)$ the two terms are possibly equivalent. Then the opinion is calculated using equation 5 where:
$a_x = \dfrac{1}{senseCount(C) + 1}$ where senseCount is calculated as the number of possible senses returned from a WordNet query of the term and example is shown in Figure 6

3. Else if $syntax(\alpha) = synonym(syntax(C))$ the two terms are possibly equivalent (e.g. city and metropolis shown in Figure 6). Add α as an individual of SubjectiveClass and create an individual of PossibleEquivalence linked with both α and C. The opinion is calculated using equation 5 where:
$a_x = \dfrac{simSense(C, \alpha)}{senseCount(C) + senseCount(\alpha) + 1}$
where simSense pertains to the number of senses in which both words occur (if they are synonyms this will be $simSense(C, \alpha) \geq 2$).

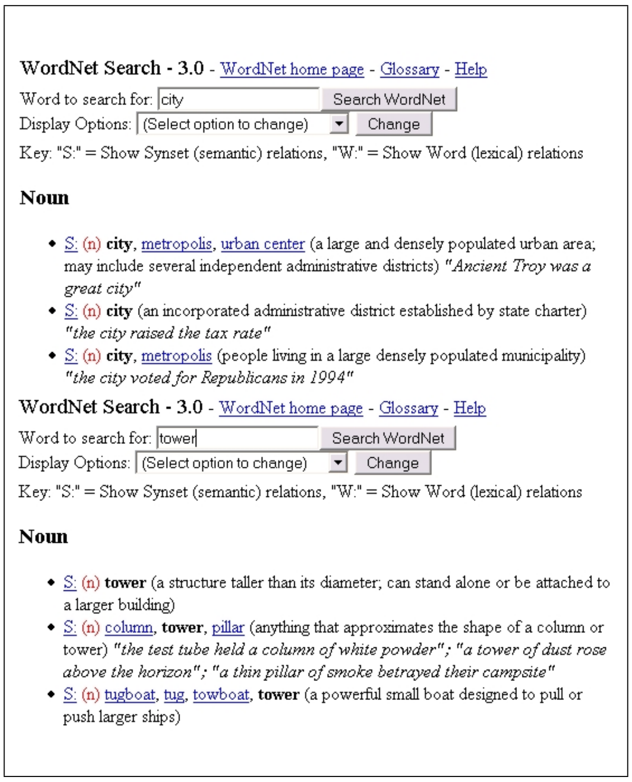
4. Else if $syntax(\alpha) \in syntax(C)$ the two terms are possibly in super-class relationship where the approximation is based on syntactic closeness (McGuinness et al. 2000)(e.g. $\alpha =$ "'tower'" and $C =$ "drum tower"). Add α as an individual of SubjectiveClass and create an individual of PossibleSubClass linked with both α and C where C as the superclass. Then the opinion is calculated using equation 5 where:
$a_x = \dfrac{1}{maxSense(\alpha, C) + 1}$ where maxSense is the maximum number of senses of either α or C.

5. Else if $syntax(C) \in syntax(\alpha)$ the two terms are possibly in sub-class relationship. Add α as an individual of SubjectiveClass and create an individual of PossibleSubClass linked with both α and C where α is the superclass. Then the opinion is calculated using equation 5 where:
$a_x = \dfrac{1}{maxSense(\alpha, C) + 1}$ where maxSense is the maximum number of senses of either α or C.

6. Else search for an existing ontology, O_n (using Swoogle (Sabou et al. 2006)) that contains both $\alpha, C \in O_n$ and import the minimum set of resources that semantically relate the terms, $O' \subseteq O_n$ where
$S(C', C'_1)$,..., $S(C'_{j-1}, C'_j S(C'_j, \alpha')$ and
$syntax(C') = syntax(C)$ and $syntax(\alpha') = syntax(\alpha)$ and $\forall C''$ from C'_1 to $C'_{j-1} syntax(C'')$ $\neq syntax(C_k \in O)$ or $syntax(\alpha_k \in V_i$
   (a) if $C'' \in O$ then remove all $S(C', C'_1)$ ..., $S(C'_n, C'')$ from O' as this will be the shortest path between the compared concepts in O and V_i.

This process will introduce additional authors and require the integration process to be performed between O and O' as above then add each R $\in O'$ to O.

If no existing ontology is found then α is added to O' as a class $C_a lpha$ such that $subclass(C_\alpha, owlThing)$ and $providedBy(C_\alpha, A_i)$ and the opinion is calculated using equation 5 and $a_x = \frac{1}{senseCount(C)+1}$


WordNet Search - 3.0 - WordNet home page - Glossary - Help

Word to search for: city Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- S: (n) **city**, metropolis, urban center (a large and densely populated urban area; may include several independent administrative districts) *"Ancient Troy was a great city"*
- S: (n) **city** (an incorporated administrative district established by state charter) *"the city raised the tax rate"*
- S: (n) **city**, metropolis (people living in a large densely populated municipality) *"the city voted for Republicans in 1994"*

WordNet Search - 3.0 - WordNet home page - Glossary - Help

Word to search for: tower Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- S: (n) **tower** (a structure taller than its diameter; can stand alone or be attached to a larger building)
- S: (n) column, **tower**, pillar (anything that approximates the shape of a column or tower) *"the test tube held a column of white powder"; "a tower of dust rose above the horizon"; "a thin pillar of smoke betrayed their campsite"*
- S: (n) tugboat, tug, towboat, **tower** (a powerful small boat designed to pull or push larger ships)

Figure 6: Results of WordNet queries for ‘city’ and ‘tower’

6 Evaluation

To evaluate the RIPOSTE Framework, an ontology (O_{A_0}) was created using Protégé [9] to capture a domain of landmarks for photography. This included locations and types of landmarks. The ontology, partially shown in Figure 3, used simplified classification and focused on taxonomy of classes and meronyms, providing a simplified ontology for classifying photos. Additionally the domain was reduced to only include concepts that related to a set of specific photographs. Additional concepts were included for clarification where necessary (preceded by ”-”).

A set of 60 photographs (Y), taken of famous landmarks in various locations around the world were suplied to an expert (A_0). The expert then annotated using O_{A_0}. This set of annotations was used as a benchmark for tag correctness.

The photographs were divided into two subsets of 40 photos (Y_1 and Y_2), containing an intersection of 20 photos ($|Y_1 \cap Y_2| = 20$) were supplied to a test group of 50 additional authors ($\{A_1...A_{50}\}$) to be annotated with at most 3 tags ($|T_{I_m}^{A_n}| \leq 3$). If a tag contained multiple words a new tag was added for each word of length > 2 and all were added as a superClass for the multi-worded tag.

Expertise of the additional providers were calculated by combing results from 1) a general question based on the previous experience with social tagging applications, and 2) a specific question for each image requiring the provider to state whether they recognised the image.

This created a set of Vocabularies $\{V_{A_1}, \ldots, V_{A_{50}}\}$ with the vocabulary sizes shown in Figure 7. These vocabularies were passed into WordNet to discover the synonyms and hypernyms shown in Figure 7.

[9]http://protege.stanford.edu/

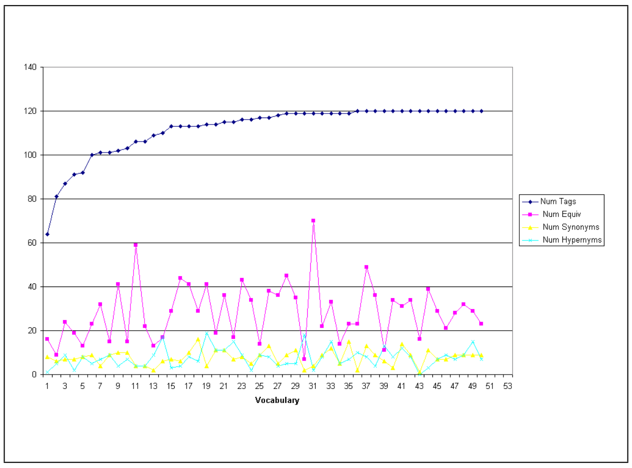


Figure 7: Vocabulary of the people involved in the evaluation

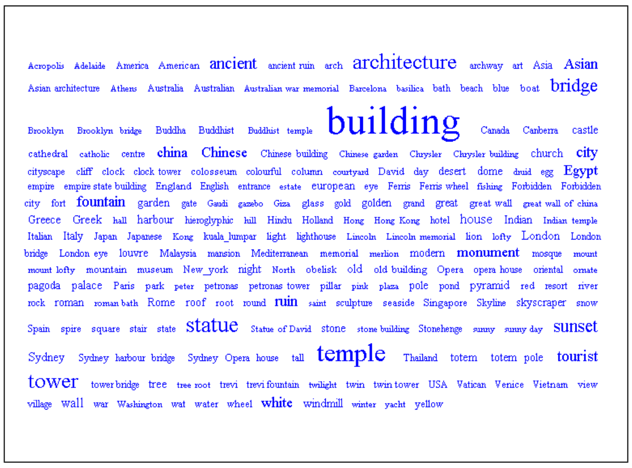

Acropolis Adelaide America American ancient ancient ruin arch architecture archway art Asia Asian Asian architecture Athens Australia Australian Australian war memorial Barcelona basilica bath beach blue boat bridge Brooklyn Brooklyn bridge Buddha Buddhist Buddhist temple building Canada Canberra castle cathedral catholic centre china Chinese Chinese building Chinese garden Chrysler Chrysler building church city cityscape cliff clock clock tower colosseum colourful column courtyard David day desert dome druid egg Egypt empire empire state building England English entrance estate european eye Ferris Ferris wheel fishing Forbidden Forbidden city fort fountain garden gate Gaudi gazebo Giza glass gold golden grand great great wall great wall of china Greece Greek hall harbour hieroglyphic hill Hindu Holland Hong Hong Kong hotel house Indian Indian temple Italian Italy Japan Japanese Kong kuala_lumpar light lighthouse Lincoln Lincoln memorial lion lofty London London bridge London eye louvre Malaysia mansion Mediterranean memorial merlion modern monument mosque mount mount lofty mountain museum New_york night North obelisk old old building Opera opera house oriental ornate pagoda palace Paris park peter petronas petronas tower pillar pink plaza pole pond pyramid red resort river rock roman roman bath Rome roof root round ruin saint sculpture seaside Singapore Skyline skyscraper snow Spain spire square stair state statue Statue of David stone stone building Stonehenge sunny sunny day sunset Sydney Sydney harbour bridge Sydney Opera house tall temple Thailand totem totem pole tourist tower tower bridge tree tree root trevi trevi fountain twilight twin twin tower USA Vatican Venice Vietnam view village wall war Washington wat water wheel white windmill winter yacht yellow

Figure 8: Resultant Tag Cloud with a belief value above 50%

6.1 Comparison of folksonomies

The vocabulary was filtered into two subsets for comparison. The first, F_1 was based on the number of occurrences of a tag within the vocabulary while the second, F_2 took into account the expertise of the user (0.5 or 1.0), based on their knowledge of tagging systems combined with their knowledge of the image. The image knowledge was calculated from whether they recognised the image. A threshold was applied to improve precision of the tags. F_1 was filtered by a threshold value of 5 on tag count, Figure 5, while a threshold value of 0.5 based on the subjective belief of the tag (calculated using consensus and discount described in Section 2.1) was applied to F_2, Figure 8.

The result shows that although the approximate size of the clouds is similar the selection of tags is not. It can be seen that generic terms, such as ’building’, still occur in both filters. An advantage of F_2 is that for images that are not as well recognised the tags created by the experts have a greater weighting. This provides the resultant folksonomy with more specialised tags (i.e. the actual names of the landmark ”Statue of David”) as opposed to more generic tags (”Statue”).

6.2 Ontology Creation

The tags in F_2 were then converted into OWL classes with no semantics or axioms except that each class was a subclass of SubjectiveClass to create an ontology O_V. Then each hypernym/hyponym relation was

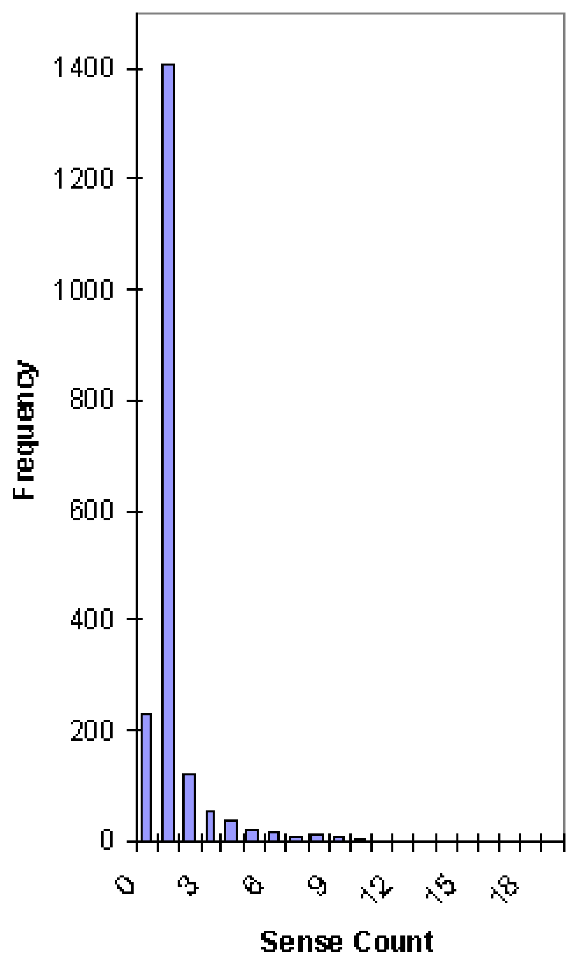


Figure 9: Tag sense count

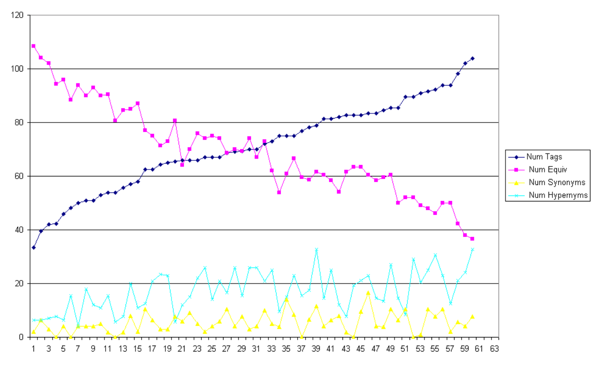


Figure 10: Vocabulary for each image

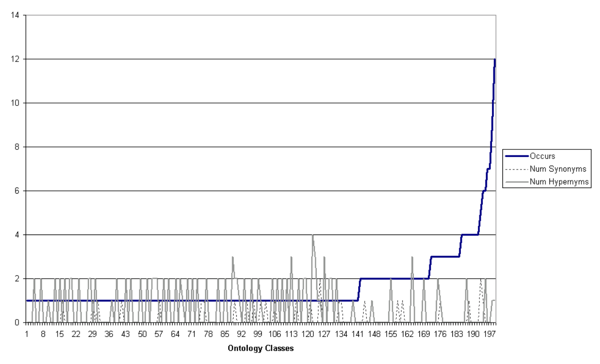


Figure 11: Vocabulary comparison with Ontology

added such that $hypernym(\alpha_1, \alpha_2) = \alpha_1 \sqsubseteq \alpha_2$ and each synonym relation was added as an equivalent relation. Additionally each author was included as a SubjectiveAuthor and a relation between each author and the tags in their vocabulary was created.

The sense count, also taken from WordNet, was applied to all tags in the folksonomy, shown in Figure 9, and all classes in the domain ontology. Finally each image was added as a SubjectiveIndividual with relationships between the authors and the tags that they were annotated with. The syntax of each class in O_V was then compared with the syntax of each class in O_{A_0} to discover equivalence, synonyms and hypernyms, Figure 11. Of the 199 ontology classes, 145 classes existed in the combined vocabulary. There were 333 synonyms and 89 hypernyms. The ontology is improved by the inclusion of additional terms allowing for greater use of synonyms and hypernyms supplied by the folksonomy to improve querying.

7 Related Work

Most categorisation for social computing is currently achieved by a meta-tagging system dubbed folksonomies, which fails to take into consideration the semantics that occur between terms. To overcome this a folksonomy can be integrated with an ontology allowing for greater reasoning to refine searches.

Current rating systems provide a popularity, or karma, measure to the content provider which is then used to rate the content submitted. This is a simplified mechanism that fails to take into account the expertise of the author. By integrating the folksonomy with an existing subjective ontology a rating of the content domain can be supplied. This can be compared with an authors previous submissions. This allows for examination of an author's consistency within a selected sub-ontology, allowing for a more fine grained approach to ratings ensures an expert in one field is not as highly regarded in a dissimilar field.

(Schmitz 2006) recommends using a probabilistic model to induce a faceted ontology from Flickr tags. By examining the number of images and the number of users that a tag belongs to a threshold is applied to remove noise. This removes noise but also minimises recall (Schmitz 2006). The thresholded set of tags is then used to discover subsumption pairs, which are manually examined for correctness and synonyms (including multilingual synonyms). It can be seen in our study that the reduction in recall of images will effect those images with more concise, or concrete, tags. This will result in many leaf nodes not being discovered reducing the ability to refine searches within the subsumption hierarchy.

The need to merge Semantic Web technologies with Web 2.0 is necessary to reduce the individual weaknesses of each approach (Heath & Motta 2007, Gruber 2007, Ankolekar et al. 2007). The AI centric nature of the Semantic Web makes it difficult for users while the user centric nature of Web 2.0 provides limited ability for AI assisted reasoning, model checking and reuse. Where as some researchers advocate replacing Web 2.0 with RDF publishing and SPARQL querying (Heath & Motta 2007) others recommend linking solutions to combine Web 2.0 and the Semantic Web (Bojārs et al. 2007, Ankolekar et al. 2007). By providing a protocol to link the tagging systems to an ontology the expressiveness can be utilised for improved querying. It is seen that a threshold is necessary to remove noise (Schmitz 2006) and the preliminary work in linking approaches also shows this. Unfortunately these approaches do not take into account the pedigree of the authors of tags which results in any threshold removing many specific tags.

Providing an ontology combined with a social site (Gruber 2006, Heath & Motta 2007) allows for additional querying that will allow photographs of 'Sydney' to be found in a search about 'Australia'. RealTravel[10], an implementation of (Gruber 2006) provides a static approach to the ontology, which does not take into consideration that as additional tags are added to the site the ontology may need to grow to provide the same reasoning abilities to the newly expanded domain.

By capturing tags in RDF (Heath & Motta 2007) the ontology becomes dynamic allowing for greater flexibility this relies on new interfaces to be created to simplify ontology evolution. By having no mechanisms in place to control the insertion of the new RDF triples it can be easy for users to provide incorrect or misleading information, which may result in a corrupted and unusable ontology.

8 Conclusion

A novel approach to integrating folksonomies to ontologies is discussed. By appending probabilities to all resources within an ontology the ability to estimate the usefulness of a result provides a mechanism for ranking based on this usefulness.

Additionally by incorporating authors with the specific resources they provide a fine grained trust calculation can be used and allows for an author's area of expertise to be taken into account when accepting results. Currently the ability to calculate an area of expertise allows for ranking of results based on the credibility and reliability of the author. Our approach differs from other social network research by considering the semantic relations between provided content. While our approach extends the capabilities of ontology integration research by allowing incomplete or incorrect ontologies to be augmented with probability. This provides a user the ability to see where more credible sources are required to improve the semantic representation of a particular sub-domain within the ontology.

8.1 Future Work

Currently we only provide mechanisms for capturing a relationship between one tag and one class. We aim to examine the possibility of capturing more complex relationships (i.e. multiple tags combined to form the ObjectProperties of multiple classes). Currently we have only evaluated tags that relate to classes and would like to examine tags that will match to semantic relationships (i.e. the tag is a verb).

Additionally we would like to evaluate the Area of Expertise to examine the possible time reduction due to less calculations and how the generalised calculation of trust will effect the precision of result rankings.

[10]http://realtravel.com

References

Al-Khalifa, H. S. & Davis, H. C. (2006), Measuring the semantic value of folksonomies, *in* 'Proceedings IEEE Conf Innovation in IT'.

Angeletou, S., Sabou, M., L.Specia & Motta, E. (2007), Bridging the gap between folksonomies and the semantic web: An experience report, *in* 'Proceedings The International Workshop Bridging the Gap between Semantic Web and Web 2.0'.

Ankolekar, A., Krötzsch, M., Tran, T. & Vrandečić (2007), 'The two cultures: Mashing up web 2.0 and the semantic web', *Journal of Web Semantics* .

Bojārs, U., Breslin, J., Finn, A. & Decker, S. (2007), 'Using the semantic web for linking and reusing data across web 2.0 communities', *Journal of Web Semantics* .

Dumbill, E. (2002), 'XML watch: Finding friends with XML and RDF'.
URL: *http://www.ibm.com/developerworks/x-foaf.html*

Gruber, T. (2006), Where the social web meets the semantic web, *in* 'Keynote at 5th International Semantic Web Conference', Vol. 4273 of *LNCS*, Springer, Athens, GA, USA.

Gruber, T. (2007), 'Collective knowledge systems: Where the social web meets the semantic web', *Journal of Web Semantics* .

Guarino, N. (1997), Semantic matching: Formal ontological distinctions for information organization, extraction, and integration., *in* 'SCIE', Vol. 1299 of *LNCS*, Springer, Frascati, Italy, pp. 139–170.

Guha, R. V. (2004), Open rating systems, *in* '1st Workshop on Friend of a Friend, Social Networking and the Semantic Web', Galway, Ireland.

Heath, T. & Motta, E. (2007), 'Ease of interaction plus ease of integration: Combing web2.0 and the semantic web in a reveiwing site', *Journal of Web Semantics* .

Hooijmaijers, D. & Stumptner, M. (2006), Trust based ontology integration for the community services sector, *in* 'Advances in ontologies. Proceedings of AOW', Hobart, Australia.

Hooijmaijers, D. & Stumptner, M. (2008), Improving integration with subjective combining of ontology mappings, *in* A. An, S. Matwin, Z. W. Ras & D. Slezak, eds, 'ISMIS', Vol. 4994 of *Lecture Notes in Computer Science*, Springer, pp. 552–562.

Jøsang, A. (2002), A logic for uncertain probabilities, *in* 'Int. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems', Vol. 9.

Jøsang, A., Ismail, R. & Boyd, C. (2005), A survey of trust and reputation systems for online service provision, *in* 'Decision Support Systems', Vol. 43 of *LNCS*, Science Direct, pp. 618 – 644.

McGuinness, D. L., Fikes, R., Rice, J. & Wilder, S. (2000), The Chimaera ontology environment., *in* 'Proceedings of AAAI', Austin, Texas, USA., pp. 1123–1124.

Noy, N. F. & Musen, M. A. (2000), PROMPT: algorithm and tool for automated ontology merging and alignment., *in* 'Proceedings of AAAI', Austin, Texas, USA., pp. 450–455.

Resnick, P., Kuwabara, K., Zeckhauser, R. & Friedman, E. (2000), 'Reputation systems.', *Communications of the ACM* **43**(12), 45–48.

Sabou, M., d'Aquin, M. & Motta, E. (2006), Using the Semantic Web as background knowledge for ontology mapping, *in* 'Proceedings of the 1st Int. Workshop on Ontology Matching', Athens, Georgia, USA.

Schmitz, P. (2006), Inducing ontology from flickr tags, *in* 'Proceedings of The International Workshop Bridging the Gap between Semantic Web and Web 2.0', IW3C2, Edinburgh, UK.

Szomszor, M., Cattuto, C., Alani, H., O'hara, K., Baldassarri, A., Loreto, V. & Servedio, V. (2007), Folksonomies, the semantic web, and movie recommendation, *in* 'Proceedings of The International Workshop Bridging the Gap between Semantic Web and Web 2.0'.

van Damme, C., Hepp, M. & Siorpaes, K. (2007), Folksontology: An integrated approach for turning folksonomies into ontologies, *in* 'Proceedings of The International Workshop Bridging the Gap between Semantic Web and Web 2.0'.

Heterogeneously Structured Ontologies
Integration, Connection, and Refinement

Oliver Kutz[1] **Dominik Lücke**[1] **Till Mossakowski**[1,2]

[1] SFB/TR 8 Spatial Cognition, University of Bremen
Cartesium, Enrique-Schmidt Strasse
28359, Bremen, Germany
Email: {okutz,luecke}@informatik.uni-bremen.de

[2] DFKI GmbH, Bremen, Germany
Cartesium, Enrique-Schmidt Strasse
28359, Bremen, Germany
Email: Till.Mossakowski@dfki.de

Abstract

This paper systematically applies tools and techniques from the area of algebraic specification theory to corresponding ontology structuring and design tasks.

We employ the heterogeneous structuring mechanisms of the heterogeneous algebraic specification language HetCasl for defining an abstract notion of structured heterogeneous ontology. This approach enables the designer to split up a heterogeneous ontology into semantically meaningful parts and employ dedicated reasoning tools to them.

In particular, we distinguish three fundamentally different kinds of combining heterogeneous ontologies: integration, connection, and refinement.

Keywords: Ontology Design & Reasoning; Modularity; Combination Techniques; Algebraic Specification

1 Introduction

Ontologies play an increasingly important role in various areas of Knowledge Representation ranging from the life sciences and engineering domains to linguistic semantics. In the process, ontologies are being designed in a broad spectrum of logics, with considerably varying expressivity and supporting quite different reasoning methods. Logics being used include description logics like $\mathcal{SROIQ}(D)$ (Horrocks et al. 2006), relational database schemes, as well as first-order and modal logics. While modularity, aligning and re-using (parts of) ontologies has received considerable attention recently,[1] there is little work on formal structuring and the aspect of heterogeneity.

We believe that a lot can be learned in this respect from techniques developed for (algebraic) specification in software engineering, and provide a systematic account that parallels structuring techniques from algebraic specification with typical problems found in ontology design.

Our work builds on and generalises and extends ideas of (Bench-Capon & Malcolm 1999), (Alagić & Bernstein 2002), (Madhavan et al. 2002), and in particular (Goguen 2005, 2006). In recent publications, we have discussed the problem of inheriting conservativity properties from parts (or modules) of an ontology to an overall integration obtained through a colimit operation (Kutz & Mossakowski 2007, 2008), and analysed various alignment approaches from the institutional viewpoint (Kutz et al. 2008).

This paper addresses the following: (1) we develop a rather abstract view of heterogeneously structured ontologies encompassing essentially all logics used in ontology design today and allowing to model the most complex relationships between various ontologies; (2) we systematise the field of 'combining ontologies' by identifying three classes of such combinations: *integrations*, *connections*, and *refinements*. The differentiating criteria are the use of signatures in the overall combination and the corresponding model-theoretic properties; (3) we analyse how various well-known ontology design and combination techniques fit into these abstract categories, including structuring through conservative extensions, ontology alignments, $\mathcal{E}$-connections, and database scheme–ontology reconciliation; (4) finally, the appendix contains a discussion how the Heterogeneous Tool Set (Hets) can support various reasoning and ontology engineering tasks; we also indicate the current and planned tool support for existing ontology languages and reasoners.

2 Heterogeneous Ontologies and Structuring

The study of modularity principles can be carried out to a quite large extent independently of the details of the underlying logical system that is used. The notion of **institutions** was introduced by Goguen and Burstall in the late 1970s exactly for this purpose (see (Goguen & Burstall 1992)). They capture in a very abstract and flexible way the notion of a logical system by describing how, in any logical system, signatures, models, sentences (axioms) and satisfaction (of sentences in models) are related.

The importance of the notion of institutions lies in the fact that a surprisingly large body of logical notions and results can be developed in a way that is completely independent of the specific nature of the underlying institution.[2]

We assume some acquaintance with the basic notions of category theory and refer to (Adámek et al. 1990) or (Mac Lane 1998) for an introduction. If $\mathcal{C}$ is a category, $\mathcal{C}^{op}$ is the dual category where all arrows

[1] For work on modularity in ontologies compare the workshop series (Haase et al. 2006, Cuenca Grau et al. 2007, Sattler & Tamilin 2008), and for work on ontology alignment/matching compare the textbook (Euzenat & Shvaiko 2007), as well as the workshops of the 'Ontology Alignment Evaluation Initiative' (see http://oaei.ontologymatching.org/).

[2] For an extensive treatment of model theory in this setting, see (Diaconescu 2008).

are reversed. For a category $\mathcal{C}$, we denote by $|\mathcal{C}|$ the class of its objects.

Definition 1 (Institutions)**.** An **institution** is a quadruple $I = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$ consisting of

- a category **Sign** of *signatures*,
- a functor $\mathbf{Sen}\colon \mathbf{Sign} \longrightarrow \mathbf{Set}$[3] giving, for each signature Σ, the set of *sentences* $\mathbf{Sen}(\Sigma)$, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *sentence translation map* $\mathbf{Sen}(\sigma)\colon \mathbf{Sen}(\Sigma) \longrightarrow \mathbf{Sen}(\Sigma')$, where often $\mathbf{Sen}(\sigma)(\varphi)$ is written as $\sigma(\varphi)$,
- a functor $\mathbf{Mod}\colon \mathbf{Sign}^{op} \longrightarrow \mathcal{CAT}$[4] giving, for each signature Σ, the category of *models* $\mathbf{Mod}(\Sigma)$, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *reduct functor* $\mathbf{Mod}(\sigma)\colon \mathbf{Mod}(\Sigma') \longrightarrow \mathbf{Mod}(\Sigma)$, where often $\mathbf{Mod}(\sigma)(M')$ is written as $M'{\restriction}_\sigma$, and $M'{\restriction}_\sigma$ is called the *σ-reduct* of M', while M' is called a *σ-expansion* of $M'{\restriction}_\sigma$,
- a satisfaction relation $\models_\Sigma \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$ for each $\Sigma \in |\mathbf{Sign}|$,

such that for each $\sigma\colon \Sigma \longrightarrow \Sigma'$ in **Sign** the following *satisfaction condition* holds:

$$(\star) \qquad M' \models_{\Sigma'} \sigma(\varphi) \text{ iff } M'{\restriction}_\sigma \models_\Sigma \varphi$$

for each $M' \in |\mathbf{Mod}(\Sigma')|$ and $\varphi \in \mathbf{Sen}(\Sigma)$, expressing that truth is invariant under change of notation and enlargement of context.[5] ◊

Examples of institutions include, among others, first- and higher-order classical logic, description logics, and various (quantified) modal logics:

Example 1. First-order Logic. In the institution $\mathbf{FOL}^{\mathsf{ms}=}$ of many-sorted first-order logic with equality, signatures are many-sorted first-order signatures, consisting of sorts and typed function and predicate symbols. Signature morphisms map symbols such that typing is preserved. Models are many-sorted first-order structures. Sentences are first-order formulas. Sentence translation means replacement of the translated symbols. Model reduct means reassembling the model's components according to the signature morphism. Satisfaction is the usual satisfaction of a first-order sentence in a first-order structure. ◊

Example 2. Relational Schemes. A signature consists of a set of sorts and a set of relation symbols, where each relation symbol is indexed with a string of sorted field names. Signature morphisms map sorts, relation symbols and field names. A model consists of a carrier set for each sort, and an n-ary relation for each relation symbol with n fields. A model reduction just forgets the parts of a model that are not needed. A sentence is a link (integrity constraint) between two field names of two relation symbols. Sentence translation is just renaming. A link is satisfied in a model if for each element occurring in the source field component of a tuple in the source relation, the same element also occurs in the target field component of a tuple in the target relation. ◊

[3]**Set** is the category having all small sets as objects and functions as arrows.

[4]$\mathcal{CAT}$ is the category of categories and functors. Strictly speaking, $\mathcal{CAT}$ is not a category but only a so-called quasicategory, which is a category that lives in a higher set-theoretic universe.

[5]Note, however, that non-monotonic formalisms can only indirectly be covered this way, but compare, e.g., (Guerra 2001).

Example 3. Description Logics. Signatures of the description logic $\mathcal{ALC}$ consist of a set of B of atomic concepts and a set R of roles, while signature morphisms provide respective mappings. Models are single-sorted first-order structures that interpret concepts as unary and roles as binary predicates. Sentences are subsumption relations $C_1 \sqsubseteq C_2$ between concepts, where concepts follow the grammar

$$C ::= B \mid \top \mid \bot \mid C_1 \sqcup C_2 \mid C_1 \sqcap C_2 \mid \neg C \mid \forall R.C \mid \exists R.C$$

Sentence translation and reduct is defined similarly as in $\mathbf{FOL}^=$. Satisfaction is the standard satisfaction of description logics. $\mathcal{ALC}^{\mathsf{ms}}$ is the many-sorted variant of $\mathcal{ALC}$. $\mathcal{ALCO}$ is obtained from $\mathcal{ALC}$ by extending signatures with nominals.

The (sub-Boolean) description logic $\mathcal{EL}$ restricts $\mathcal{ALC}$ as follows: $C ::= B \mid \top \mid C_1 \sqcap C_2 \mid \exists R.C$. $\mathcal{SHOIN}$ extends $\mathcal{ALC}$ with role hierarchies, transitive and inverse roles, (unqualified) number restrictions, and nominals, etc. ◊

Example 4. Modal Logics. The modal logic $\mathbf{S4_u}$ has signatures as classical propositional logic, consisting of propositional variables. Sentences are built as in propositional logic, but add two unary modal operators, $\Box$ and $\blacksquare$. Models are standard Kripke structures but based on reflexive and transitive relations. Satisfaction is standard modal satisfaction, where $\Box$ is interpreted by the transitive reflexive relation, and $\blacksquare$ by universal quantification over worlds.

The standard formulation of first-order modal logic **QS5** (due to Kripke) has signatures similar to $\mathbf{FOL}^=$, including variables and predicate symbols. Sentences follow the grammar for $\mathbf{FOL}^=$-sentences using Booleans, quantifiers, and identity, while adding the $\Box$ operator, but leaving out constants and function symbols. Models are constant-domain first-order Kripke structures, with the usual first-order modal satisfaction. ◊

2.1 Structured Ontologies

The essential advantage of the theory of institutions is the possibility of providing structuring operations and module concepts independently of the underlying logical system. Hence, in the sequel, let us fix some arbitrary institution $I = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$. The basic structuring operation for ontologies is surely that of *importing* other ontologies. The notion of *development graph* captures this, and also renaming of symbols.

Definition 2. A **development graph** is an acyclic, directed graph[6] $\mathcal{DG} = \langle \mathcal{N}, \mathcal{L} \rangle$. Here, $\mathcal{N}$ is a set of nodes. Each node $N \in \mathcal{N}$ is labelled with a pair (Σ^N, Ψ^N) such that Σ^N is a signature and $\Psi^N \subseteq \mathbf{Sen}(\Sigma^N)$ is the set of **local axioms** of N. $\mathcal{L}$ is a set of directed links, so-called (global[7]) **definition links** $K \xrightarrow{\sigma} N$, annotated with a signature morphism $\sigma : \Sigma^K \to \Sigma^N$.

Given a node $N \in \mathcal{N}$, its associated theory $\mathbf{Th}_{\mathcal{DG}}(N)$ is inductively defined to consist of

- all the local axioms Ψ^N, and
- for each $K \xrightarrow{\sigma} N \in \mathcal{DG}$, all of $\mathbf{Th}_{\mathcal{DG}}(K)$ translated by σ.

The class of models $\mathbf{Mod}_{\mathcal{DG}}(N)$ of a node N is defined as $\mathbf{Mod}(\mathbf{Th}_{\mathcal{DG}}(N))$. ◊

[6]In (Kutz & Mossakowski 2007, 2008), we have identified a structured ontology with a diagram.

[7]There are also local and hiding definition links, which require a more refined model-theoretic semantics.

Complementary to definition links, which *define* the theories of related nodes, we also allow for **theorem links** with the help of which we are able to *postulate* relations between different theories. A (global) theorem link is an edge $K \overset{\sigma}{\dashrightarrow} N$, where $\sigma\colon \Sigma^K \longrightarrow \Sigma^N$. $\mathcal{DG}$ implies a theorem link $K \overset{\sigma}{\dashrightarrow} N$ (denoted $\mathcal{DG} \models K \overset{\sigma}{\dashrightarrow} N$) iff for all $M \in \mathbf{Mod}_{\mathcal{DG}}(N)$, $M{\restriction_\sigma} \in \mathbf{Mod}_{\mathcal{DG}}(K)$.

A global definition (or also theorem) link $K \xrightarrow{\sigma} N$ can be strengthened to a **conservative extension link** (denoted as $K \xrightarrow[cons]{\sigma} N$); it holds if every K-model has a σ-expansion to an N-model. Such annotations can be seen as another kind of proof obligations. Definitional extensions are introduced in a similar way (annotated with *def*); here the σ-expansion has to be unique.

Many languages for structuring, modularity and alignment of ontologies can be mapped into this formalism of development graphs.[8] In this paper, we use the language CASL (Bidoit & Mosses 2004). A CASL library consists of specification definitions of the form

```
spec <name> =
  <spec>
end
```

A specification `<spec>` can be a basic specification consisting of a signature and some axioms (with syntax specific to the given institution). It corresponds to a node with local axioms in a development graph. Concerning structuring, specifications can be extended or united, written `<spec> then <spec>` or `<spec> and <spec>`, and leading to definition links in the development graph. Extensions add some new signature elements and axioms, and can be declared to be conservative or definitional. Unions unite the requirements of two specifications, thereby intersecting their model classes. Renamings, written `<spec> with <signature-morphism>`, rename a specification along a signature morphism; again, this leads to a definition link in the development graph. The declaration `view view1: sp1 to sp2` will generate a theorem link between the nodes representing `sp1` and `sp2` in the development graph. Details of the translation to development graphs, as well as a treatment of hiding, can be found in (Mossakowski et al. 2006). Moreover, the appendix contains additional details about the syntax of CASL (and HETCASL introduced below) as well as example specifications.

2.2 Heterogeneous Ontologies

As (Schorlemmer & Kalfoglou 2008) argue convincingly, since ontologies are being written in many different formalisms, like relation schemata, description logics, first-order logic, and modal (first-order) logics, alignments of ontologies need to be constructed across different institutions.

To obtain heterogeneous logical theories, one needs to fix some graph of logics and logic translations, usually formalised as institutions and so-called institution comorphisms, see (Goguen & Roşu 2002). The latter are again governed by the satisfaction condition, this time expressing that truth is invariant also under change of notation across different logical formalisms:

$$M' \models^J_{\Phi(\Sigma)} \alpha_\Sigma(\varphi) \Leftrightarrow \beta_\Sigma(M') \models^I_\Sigma \varphi.$$

Here, $\Phi(\Sigma)$ is the translation of signature Σ from institution I to institution J, $\alpha_\Sigma(\varphi)$ is the translation of the Σ-sentence φ to a $\Phi(\Sigma)$-sentence, and $\beta_\Sigma(M')$ is the translation (or perhaps: reduction) of the $\Phi(\Sigma)$-model M' to a Σ-model.

The so-called **Grothendieck institution** is a technical device for giving a semantics to heterogeneous theories involving several institution (see Diaconescu 2002, Mossakowski 2002). The Grothendieck institution is basically a flattening, or disjoint union, of the logic graph. A signature in the Grothendieck institution consists of a pair (L, Σ) where L is a logic (institution) and Σ is a signature in the logic L. Similarly, a Grothendieck signature morphism $(\rho, \sigma) : (L_1, \Sigma_1) \to (L_2, \Sigma_2)$ consists of a logic translation $\rho = (\Phi, \alpha, \beta)\colon L_1 \longrightarrow L_2$ plus an L_2-signature morphism $\sigma\colon \Phi(\Sigma_1) \longrightarrow \Sigma_2$. Sentences, models and satisfaction in the Grothendieck institution are defined in a componentwise manner.

We now arrive at the following:

Definition 3. An **abstract structured heterogeneous ontology** (w.r.t. some logic graph) is a node O in a development graph $\mathcal{DG}$ in the corresponding Grothendieck institution. We sometimes also identify O with its theory $\mathbf{Th}_{\mathcal{DG}}(O)$; however, note that then the structuring is lost. $\diamondsuit$

To be able to write down such heterogeneous ontologies in a concise manner, we extend CASL to HETCASL as follows: HETCASL provides the notation `logic <logic-name>`, which defines the institution of the following specifications until that keyword occurs again. Also, a specification can be translated along a comorphism; this is written `<spec> with logic <comorphism-name>`. See the extended example on Page 7 for the look-and-feel of HETCASL specifications. Of course, abstract structured heterogeneous ontologies can be formulated in different notations, and HETCASL is only one of them. Another option would be an extension of OWL structuring mechanisms by keywords dealing with heterogeneity.

3 Heterogeneous Integration

Informally, an *integration* of two ontologies O_1, O_2 into a third ontology $\mathcal{O}$ is any operation by which O_1, O_2 are 're-interpreted' from the (global) point of view of $\mathcal{O}$.

Definition 4. Given ontologies O_1, O_2, and an ontology $\mathcal{O}$, in institutions I_1, I_2 and I, respectively, we say that $\mathcal{O}$ **heterogeneously integrates** O_1 and O_2 if there are theorem links (i.e. theory interpretations) $\lambda_1 : O_1 \mapsto \mathcal{O}$ and $\lambda_2 : O_2 \mapsto \mathcal{O}$. $\diamondsuit$

Thus, given $O_1 \models_{I_1} \phi$ and $O_2 \models_{I_2} \psi$, we have $\mathcal{O} \models_I \lambda_1(\phi), \lambda_2(\psi)$, i.e., consequence is preserved upwards.

In the approach of (Schorlemmer & Kalfoglou 2008), two ontologies O_1 and O_2 are aligned by mapping them into a *common reference ontology* $\mathcal{O}$ as follows: theories O_1 and O_2 are said to be **semantically integrated** with respect to a theory $\mathcal{O}$ if (1) there exist *consequence-preserving sentence translations* $\alpha_1\colon O_1 \longrightarrow \mathcal{O}, \alpha_2\colon O_2 \longrightarrow \mathcal{O}$; (2) there exist *structure reducts* $\beta_1\colon \mathbf{Mod}(\mathcal{O}) \longrightarrow \mathbf{Mod}(O_1), \beta_2\colon \mathbf{Mod}(\mathcal{O}) \longrightarrow \mathbf{Mod}(O_2)$; and (3) $\mathcal{O}$ is consistent.

[8]By 'modularity' we here refer to the notion of 'ontological module' defined through conservativity properties, as it has been investigated for instance in Konev et al. (2008), Cuenca Grau, Horrocks, Kazakov & Sattler (2008), Kutz & Mossakowski (2008).

$$\begin{tikzcd} & \mathcal{O} & \\ O_1 \arrow[ur, "\alpha_1"'] & & O_2 \arrow[ul, "\alpha_2"] \end{tikzcd}$$

Figure 1: Integration into reference ontology

Example 5. (From Schorlemmer & Kalfoglou 2008, abridged) Suppose that O_1 is a relational scheme. It contains `author_of(person,paper)` and `person(id,name)` with a relationship from `person` to `id`. O_2 is a description logic theory. It contains $\mathsf{Article} \sqsubseteq \exists \mathsf{\mathit{author}}.\top \sqcap \exists \mathsf{\mathit{title}}.\top$, etc.

The reference ontology $\mathcal{O}$ is a first-order theory. It contains, among others:

$$\begin{array}{rcl} \forall x.(\mathit{Working\_Person}(x) & \rightarrow & (\mathit{Thing}(x) \wedge \\ & & \exists y.(\mathit{String}(y) \wedge \mathit{Name}(x,y)))) \\ \forall x.(\mathit{Researcher}(x) & \rightarrow & \mathit{Working\_Person}(x)) \end{array}$$

Theory interpretations α_1, α_2 can be given as follows:

$$\begin{array}{rcl} \alpha_1(\mathtt{person(p,n)}) & = & \mathit{Researcher}(p) \wedge \mathit{String}(n) \wedge \\ & & \mathit{Name}(p,n) \\ \alpha_1(\mathtt{author\_of(p,a)}) & = & \mathit{Researcher}(p) \wedge \mathit{Article}(a) \wedge \\ & & \mathit{Author}(a,p) \wedge \exists j.(\mathit{Journal}(j) \\ & & \wedge\ \mathit{Has\_Article}(j,a)) \\ \alpha_2(\mathsf{Article}(x)) & = & \mathit{Paper}(x) \end{array}$$

◊

We see a number of problems with this approach (and therefore we will reformulate the example as a heterogeneous refinement on Page 7). First, allowing for arbitrary sentence maps α_i is simply too liberal. For example, α_i could map every sentence to *true*.[9] It seems more reasonable to use *signature morphisms* and their induced sentence translation maps instead. This approach, however, is less flexible in one aspect: with the approach of (Schorlemmer & Kalfoglou 2008) (using first-order logic), a predicate symbol p may be mapped to a formula φ. However, this is usually better captured by allowing for *derived signature morphisms* (see (Sannella & Burstall 1983)), which here are just signature morphisms into a conservative extension (e.g. an extension by the definition $p(x) \Leftrightarrow \varphi$). Secondly, and more importantly perhaps, there may simply be no suitable common reference ontology at hand. Rather, the common super-ontology should be suitably *constructed* from O_1 and O_2, identifying certain concepts, while keeping others distinct, leading to the concept of V-*alignment* discussed in the next section. Another possibility is to compare the two ontologies using a view, leading to the concept of *refinement* discussed on Page 6.

4 Heterogeneous Connection

Intuitively, the difference between 'integrations' and 'connections' is that in the former we combine two ontologies O_1 and O_2 using a typically large and previously-known reference ontology $\mathcal{O}$. The models of $\mathcal{O}$ are typically much richer than those of O_1 and O_2. By contrast, *connection* of two ontologies is done in such a way that the respective theories, signatures, and models are kept disjoint, and a (usually small and flexible) *bridge theory* formulated (in a bridge language) over a signature that *goes across* the sort structure of the components is used to *link together* the two ontologies. Using the general approach of colimits, an overall connection ontology can be *automatically computed* from the bridge theory and the involved ontologies. Moreover, the models of this overall ontology are obtained as amalgamations of models of the individual ontologies — no new structure is added (expect from new definitions, which however can always be interpreted uniquely).

[9](Schorlemmer & Kalfoglou 2008) suggest to solve this problem by a possible restriction to conservative translations; however, even then the translation mapping every theorem in O_i to *true* and every non-theorem to *false* still is a valid but useless example.

4.1 Connection through Alignments

4.1.1 V-Alignments

(Zimmermann et al. 2006) address the problem of alignment without a common reference ontology. Given ontologies O_1 and O_2, an **interface** (for O_1, O_2)

$$\langle \Sigma, \sigma_1 \colon \Sigma \longrightarrow \mathsf{Sig}(O_1), \sigma_2 \colon \Sigma \longrightarrow \mathsf{Sig}(O_2) \rangle$$

specifies that (using informal but suggestive notation)

- concepts $\sigma_1(c)$ in O_1 and $\sigma_2(c)$ in O_2 are identified for each concept c in Σ, regardless of whether the concepts have the same name or not, and
- concepts in $O_1 \setminus \sigma(\Sigma_1)$ and $O_2 \setminus \sigma(\Sigma_2)$ are kept distinct, again regardless of whether they have the same name or not.

The resulting common ontology $\mathcal{O}$ is not given a priori, but rather it is computed from the aligned ontologies via the interface. This computation is a pushout in the sense of category theory, which in this case is just a disjoint union with identification of specific parts (namely those given through $\langle \Sigma, \sigma_1, \sigma_2 \rangle$).

V-alignments can thus deal with basic alignment problems, such as *synonymy* (identifying different symbols with the same meaning) and *homonymy* (separating (accidentally) identical symbols with different meaning)—see Figure 2.

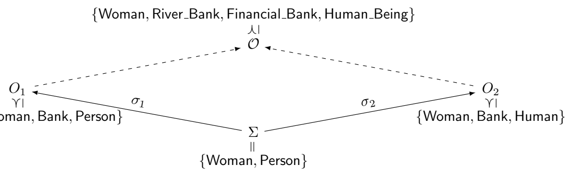


Figure 2: V-alignment: integration through interface (dashed arrows mean definition links automatically computed via colimits)

Note that alignments are encoded entirely into the interface; *finding* an appropriate alignment (i.e. an interface $\langle \Sigma, \sigma_1, \sigma_2 \rangle$) is a different problem, compare (Euzenat & Shvaiko 2007), and independent from the chosen formal representation.

Example 6. In Figure 2, the interface $\langle \Sigma, \sigma_1, \sigma_2 \rangle$ specifies that the two instances of the concept Woman as well as Person and Human are to be identified. This yields two concepts Woman and Human_Being in the push-out ontology $\mathcal{O}$ obtained along the dashed arrows. Because Bank does not appear in the interface, it also determines that the two instances of Bank are to be understood as homonyms, and thus generates two new distinct concepts. ◊

However, notion such as *polysemy* are typically understood to relate terms that have a different, but *related* meaning, and can thus not be dealt with by simply identifying symbols or keeping them apart. This problem can be solved, however, by considering

$\mathcal{E}$-connections a general form of alignment (see (Kutz et al. 2008)). Similarly, (Zimmermann et al. 2006) themselves raise the criticism that V-Alignments do not cover the case where a concept Woman in O_1 is aligned with a concept Person in O_2: here, the resulting ontology should turn Woman into a subconcept of Person. This is not directly possible with the pushout approach.

4.1.2 W-Alignments

In order to solve this problem of V-Alignments, (Zimmermann et al. 2006) introduce W-Alignments. They consist of two V-Alignments, using an intermediate bridge ontology B. The latter can be used to specify subconcept relationships like Woman $\sqsubseteq$ Person as mentioned above.

$$\begin{tikzcd}
\{\mathsf{Woman}\} \arrow[d, phantom, "\curlywedge"] & & \{\mathsf{Woman} \sqsubseteq \mathsf{Person}\} \arrow[d, phantom, "\curlywedge"] & & \{\mathsf{Person}\} \arrow[d, phantom, "\curlywedge"] \\
O_1 & & B & & O_2 \\
& \Sigma_1 \arrow[ul] \arrow[ur] \arrow[d, phantom, "\parallel"] & & \Sigma_2 \arrow[ul] \arrow[ur] \arrow[d, phantom, "\parallel"] & \\
& \{\mathsf{Woman}\} & & \{\mathsf{Person}\} &
\end{tikzcd}$$

Figure 3: W-alignment: integration through bridge ontology

(Zimmermann et al. 2006) list the behaviour of compositions as a weak point of this approach. However, we see as the main weak point the rather loose coupling of O_1 and O_2; indeed, the bridge ontology is something like a super-ontology of a sub-ontology and hence can be anything. Nevertheless, an advantage of W-alignments over semantic integrations into a reference ontology remains: it is not possible to map sentences into the integration ontology in a completely arbitrary fashion. In (Kutz et al. 2008), we have shown that various kinds of alignments can be analysed as certain 'shapes' of diagrams that can be represented and reasoned with in HETS.

4.2 Connection through Interface and Colimit

The general idea of combination through an interface by computing a colimit is shown in Fig 4. Here, Σ_1

$$\begin{tikzcd}
& & \mathcal{O} & & \\
O_1 \arrow[urr, dashed] & & B \arrow[u, dashed] & & O_2 \arrow[ull, dashed] \\
& \Sigma_1 \arrow[ul] \arrow[ur, "c"'] & & \Sigma_2 \arrow[ul, "c"] \arrow[ur] &
\end{tikzcd}$$

Figure 4: Connection through interface and colimit

is a subsignature of ontology O_1, Σ_2 a subsignature of ontology O_2, B an interface formalised in a bridge logic such as $\mathbf{FOL}^{\mathsf{ms}=}$, and $\mathcal{O}$ the colimit ontology computed from the diagram.

Example 5 can be reformulated in this setting by taking O_1 to be the relational scheme formalisation, O_2 the DL knowledge base, and B the necessary first-order axioms to achieve the desired reconciliation. The 'reference ontology' is now obtained as a pushout. The complete specification for this scenario is given in Appendix B.2.

4.3 $\mathcal{E}$-Connections

Heterogeneous knowledge representation was a major motivation also for the design of 'modular ontology languages', such as DDLs (Borgida & Serafini 2003) and $\mathcal{E}$-connections (Kutz et al. 2002, 2004). We here concentrate on the latter. $\mathcal{E}$-connections were originally conceived as a versatile and computationally well-behaved technique for combining logics, but were subsequently quickly adopted as a framework for the combination of ontologies in the Semantic Web (Cuenca Grau, Parsia & Sirin 2008).

We here show how the combination of ontologies via such modular languages can be re-formulated as structured heterogeneous ontologies, and indicate how this idea can be generalised to the institutional level.

The general idea behind this combination method is that the interpretation domains of the connected logics are interpreted by disjoint (or sorted) vocabulary and interconnected by means of *link relations*. The language of the $\mathcal{E}$-connection is then the union of the original languages enriched with operators capable of talking about the link relations.

$\mathcal{E}$-connections, just as DLs, offer an appealing compromise between expressive power and computational complexity: although powerful enough to express many interesting concepts, the coupling between the combined logics is sufficiently loose for proving general results about the transfer of decidability: if the connected logics are decidable, then their connection will also be decidable.

For lack of space, we can only roughly sketch the formal definitions, and refer the reader to (Kutz et al. 2004) for details.

We assume that the **languages** $\mathcal{L}_1$ and $\mathcal{L}_2$ of two ontologies O_1 and O_2 are pairwise disjoint. To form a connection $\mathcal{C}^{\mathcal{E}}(O_1, O_2)$, fix a non-empty set $\mathcal{E} = \{E_j \mid j \in J\}$ of binary relation symbols. The **basic $\mathcal{E}$-connection language** is then defined inductively by enriching the respective languages with the basic $\mathcal{E}$-connection-operators $\langle E_j \rangle^1, \langle E_j \rangle^2$, interpreting the link relations.

Fig. 5 displays the connection of two ontologies, by means of a single link relation E. Here, the concept $\langle E \rangle^1 (\{a\})$ of O_1 'corresponds' to the nominal $\{a\}$ of ontology O_2: it collects the set of all those points in O_1 that 'can be seen' from a (in O_2) along the relation E.

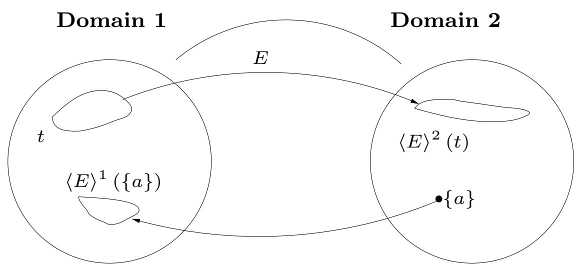


Figure 5: A two-dimensional connection.

Formally, the semantics is as follows: a structure

$$\mathfrak{M} = \left\langle \mathfrak{W}_1, \mathfrak{W}_2, \mathcal{E}^{\mathfrak{M}} = (E_j^{\mathfrak{M}})_{j \in J} \right\rangle,$$

where $\mathfrak{W}_i = (W_i, .^{\mathfrak{W}_i})$ is an interpretation of O_i for $i \in \{1, 2\}$ and $E_j^{\mathfrak{M}} \subseteq W_1 \times W_2$ for each $j \in J$, is called an **interpretation** for $\mathcal{C}^{\mathcal{E}}(O_1, O_2)$. Given concepts C_i of ontology O_i, for $i = 1, 2$, denoting subsets of W_i, the semantics of the basic $\mathcal{E}$-connection operators is

$$\begin{aligned}
(\langle E_j \rangle^1 C_2)^{\mathfrak{M}} &= \{x \in W_1 \mid \exists y \in C_2^{\mathfrak{M}} \ (x, y) \in E_j^{\mathfrak{M}}\} \\
(\langle E_j \rangle^2 C_1)^{\mathfrak{M}} &= \{x \in W_2 \mid \exists y \in C_1^{\mathfrak{M}} \ (x, y) \in E_j^{\mathfrak{M}}\}
\end{aligned}$$

$$\begin{tikzcd}
 & \mathcal{C}^{\mathcal{E}}(\mathcal{S}_1^m, \mathcal{S}_2^m) & \\
 & \mathcal{S}_1^m \uplus \mathcal{S}_2^m \arrow[u] & \\
\mathcal{S}_1^m \arrow[ur, dashed] & & \mathcal{S}_2^m \arrow[ul, dashed] \\
 & \emptyset \arrow[ul] \arrow[ur] & \\
\mathcal{S}_1 \arrow[uu] & & \mathcal{S}_2 \arrow[uu]
\end{tikzcd}$$

Figure 6: $\mathcal{E}$-connections as a structured heterogeneous theory

$\mathcal{E}$-connections can be considered as many-sorted heterogeneous theories: component ontologies can be formulated in different logics, but have to be build from many-sorted vocabulary, and link relations are interpreted as relations connecting the sorts of the component logics.

The main difference between distributed description logics (DDLs) (Borgida & Serafini 2003) and various $\mathcal{E}$-connections now lies in the expressivity of the 'link language' $\mathcal{L}$ connecting the different ontologies. While the link language of DDL is a certain sub-Boolean fragment of many sorted $\mathcal{ALC}$, the basic link language of $\mathcal{E}$-connections is $\mathcal{ALCI}^{\mathsf{ms}}$.

Such many-sorted theories can easily be represented in a diagram as shown in Fig. 6. Here, we first (conservatively) obtain a disjoint union $\mathcal{S}_1^{\mathsf{m}} \uplus \mathcal{S}_2^{\mathsf{m}}$ as a pushout, where the component ontologies have been turned into sorted variants (using an institution comorphism from the single-sorted to the many-sorted logic), and the empty interface guarantees that no symbols are shared at this point. An $\mathcal{E}$-connection KB in language $\mathcal{C}^{\mathcal{E}}(\mathcal{S}_1^{\mathsf{m}}, \mathcal{S}_2^{\mathsf{m}})$ is then obtained as a (typically not conservative) theory extension.

The idea to 'connect' logics can be elegantly generalised to the institutional level (compare Baader & Ghilardi (2007) who note that their 'connections' are an instance of a more general co-comma construction), but details have to remain sketchy here. However, it should be clear from the above that our Grothendieck institution approach is general enough to formally capture such connections.

Note that this generalises the $\mathcal{E}$-connections of (Kutz et al. 2002), the DDLs of (Borgida & Serafini 2003), as well as the connections of Baader & Ghilardi (2007) in two important respects: first, the institutional level generalises the term-based abstract description languages (ADS) that are an abstraction of modal and description logics, and the rather general definition of bridge theory similarly abstracts from the languages previously employed for linking that were similarly inspired by modal logic.

While there are no implemented, specialised algorithms available deciding satisfiability in $\mathcal{E}$-connections (except limited support in the Pellet system (Cuenca Grau, Parsia & Sirin 2008) which is currently being added to HETS as a supported prover), semi-decidable reasoning for more expressive $\mathcal{E}$-connections is provided by HETS through suitable translation by a comorphisms in a supported logic.

5 Heterogeneous Refinements

Integrations and connections are essentially *symmetric* combination techniques in the sense that the order in which component ontologies participate in the overall combination is irrelevant. Rather, the difference lies in the way 'local' signatures are mapped into the overall signature. In contrast to this a heterogeneous refinement is a asymmetric technique, stating that all

$$\begin{tikzcd}
O_1' \arrow[r, dotted, "\sigma"] & O_2' \\
O_1 \arrow[u, "\theta"] & O_2 \arrow[u, "c"', "\eta"]
\end{tikzcd}$$

Figure 7: A refinement Diagram.

axioms of a 'coarser' ontology are also true in a 'finer' (refined) one.

Definition 5. Given two ontologies O_1 and O_2 in the same logic, O_2 is called a **refinement** of O_1 if there is a theorem link $O_1 \xrightarrow{\sigma} O_2$ that follows from the underlying development graph.

Now let ontologies O_1 and O_2 in logics $\mathcal{L}_{O_1}$ and $\mathcal{L}_{O_2}$ be given, such that there is a logic $\mathcal{L}$ with comorphisms $\mathcal{L}_{O_1} \xrightarrow{\theta} \mathcal{L}$ and $\mathcal{L}_{O_2} \xrightarrow{\eta} \mathcal{L}$, where η is model-theoretically conservative.

The translations of the ontologies along the comorphism are referred to as ${O_1}'$ and ${O_2}'$. We call O_2 a **heterogeneous refinement** of O_1 if there is a theorem link $O_1' \xrightarrow{\sigma} O_2'$ that follows from the underlying development graph. $\Diamond$

Proposition 1. *For a heterogeneous refinement, any O_2-model can be translated to an O_1-model, and moreover, logical consequence is preserved along refinement: for $\sigma(\theta(\varphi)) = \eta(\psi)$, $O_1 \models \varphi$ implies $O_2 \models \eta(\psi)$.*

A heterogeneous refinement as given in Def. 5 is depicted in Fig. 7 (here, c denotes conservativity). In HETCASL, this concept is addressed by the notion of `view` creating a proof obligation, as discussed in more detail in Appendix. A.

The notion of a heterogeneous refinement also leads to the definition of heterogeneous sub-ontology.

Definition 6. We call an ontology O_1 a *(heterogeneous) sub-ontology* of O_2 iff O_2 is a (heterogeneous) refinement of O_1.

The standard notion of sub-ontology as a subset of the axioms (Haase et al. 2005, Kalyanpur et al. 2007) is recovered in the homogeneous case if σ additionally is an injection.

Moreover, our abstract definition of refinement entails the common definition in software engineering: consider O_1 to be a specification of a program in an algebraic specification language, and O_2 its implementation in a programming language. Refinements are a common problem in the world of ontologies as well: establish whether a domain ontology is consistent with respect to the knowledge represented in a foundational ontology. Consider a domain ontology written in $\mathcal{OWL}$-DL that is expected to refine the abstract knowledge given in DOLCE, written in **FOL**. In this case, the domain ontology should be a heterogeneous refinement of DOLCE (or of a part of DOLCE, if one considers hiding).

5.1 From Relational Scheme to Ontology

Recall Example 5 of a semantic integration into a reference ontology. The kind of integration required here can be dealt with much more elegantly as a heterogeneous refinement. Consider the HETCASL specification given in Fig. 9. Here, `Biblio_DL` is a DL ontology about bibliographical information, written in a concrete syntax close to Manchester Syntax (Horridge et al. 2006), and `Biblio_RS` is the scheme of a

$$\begin{tikzcd}
\mathtt{Biblio\_DL}' & \mathtt{Biblio\_RS}' \arrow[l, dotted] \\
\mathtt{Biblio\_DL} \arrow[u, "def"] & \mathtt{Biblio\_RS} \arrow[u, "def"']
\end{tikzcd}$$

Figure 8: A view from RelScheme to HetDL.

relational database intended to capture similar knowledge. Assume we want to show that the ontology is a refinement of the database schema, as illustrated in Fig. 8. A view Biblio_RS_in_DL is used for this pur-

```
logic DL
spec Biblio_DL =
  Class: Researcher
    SubclassOf: name some Thing
  Class: Article
    SubclassOf: author some Thing, title some Thing
  Class: Journal
    SubclassOf: name some Thing, hasArticle some Thing,
                impactFactor some Thing
end

logic RelScheme
spec Biblio_RS =
  Tables
    person(key id:integer, name:string)
    author_of(person, paper:integer)
    paper(key id:integer,title:string,
          published_in:integer)
    journal(key id:integer,name:string,
          impact_factor:float)
  Relationships
    author_of[person]      -> person[id]  one_to_many
    author_of[paper]       -> paper[id]   one_to_many
    paper[published_in]    -> journal[id] one_to_many
end

logic CASL
view Biblio_RS_in_DL : Biblio_RS to
  { Biblio_DL with logic DL -> CASL
    then %def
      preds
        journal(j,n,f:Thing) <=>
          Journal(j) /\ name(j,n) /\ impactFactor(j,f);
        paper(a,t,j:Thing) <=>
          Article(a) /\ Journal(j) /\ hasArticle(j,a) /\
          title(a,t);
        author_of(p,a:Thing) <=>
          Researcher(p) /\ Article(a) /\ author(p,a);
        person(p,n:Thing) <=> Researcher(p) /\ name(p,n)
  } = logic RelationalScheme -> CASL
end
```

Figure 9: Heterogeneous specification in HetCasl.

pose, stating that the ontology satisfies the relational scheme axioms (referential integrity constraints). Of course, this is not possible literally, but rather the ontology is mapped to first-order logic (Casl) and then definitionally (look at the %def) extended to Biblio_DL′ with a definition of the database tables in terms of the ontology classes and properties. Also, Biblio_RS is translated to first-order logic, yielding Biblio_RS′, and the view expresses a theory morphism from Biblio_RS′ to Biblio_DL′.

The involved signature and theory morphisms live in the Grothendieck institution. Thus, we can avoid the use of arbitrary maps α_i as in (Schorlemmer & Kalfoglou 2008), and instead rely entirely on (Grothendieck) signature morphisms. Actually, the above view is *not* provable—but it is if an inverse of the role hasArticle is introduced and used to restrict the class Article.

5.2 From Ontology to Relational Scheme

Dually, assume we are given an ontology that is supposed to logically describe a database scheme.

$$\begin{tikzcd}
\mathtt{Biblio\_DL}' \arrow[r, dotted] & \mathtt{Biblio\_RS}' \\
\mathtt{Biblio\_DL} \arrow[u, "def"] & \mathtt{Biblio\_RS} \arrow[u, "def"']
\end{tikzcd}$$

Figure 10: A view from HetDL to RelScheme.

We again use Biblio_RS and Biblio_DL, as given in Fig. 9. We now have a heterogeneous refinement from Biblio_DL to Biblio_RS, as depicted in Fig. 10. The rest of the discussion is analogous to the previous section, and hence left out. The specification of this refinement containing the integrity constraints is given in Appendix B.1.

6 Outlook and Future Work

We have introduced an abstract framework for the study of structured heterogeneous ontologies, allowing for a systematic analysis of conceptual and algorithmic problems in heterogeneous environments that were previously considered rather disparate. In particular, we have analysed in detail a specific example heterogeneous ontology from the literature, consisting of a bibliographical database and a related ontology. We have shown that these ontologies can be heterogeneously combined in different ways: (1) via *integration* in a common reference ontology, which is known beforehand, (2) via *connection* through a bridge theory, which operates directly on the involved ontologies and allows for the *automatic* construction of a combined ontology via colimits, and (3) via *refinement*, which provides the strongest relation between the ontologies: namely, that each bibliographical database satisfying the given relational scheme's integrity constraints also gives rise to a model of the ontology, and vice versa. It is not surprising that this strong combination via refinement was only possible after extending the ontology in a suitable way, leading to a better matching between the relational scheme and the ontology.

We believe that these general patterns of ontology combination can also be found in other examples, also involving completely different application domains (and not necessarily being connected with databases). Indeed, the structured reasoning support that our approach allows has already been used to answer questions that 'standard' automated reasoning can not tackle: the consistency of the first-order version of the foundational ontology Dolce (reformulated as a HetCasl specification) can be verified by model-checking a view into a finite specification of a model for Dolce.

Currently, we are working on integrating a tool for the discovery of theory morphisms into the Heterogeneous Tool Set, which would allow (semi)-automatic structuring of ontologies.

Acknowledgements

Work on this paper has been supported by the Vigoni program of the DAAD, by the DFG-funded collaborative research center SFB/TR 8 'Spatial Cognition' and by the German Federal Ministry of Education and Research (Project 01 IW 07002 FormalSafe).

We thank John Bateman, Mihai Codescu, Joana Hois, and Lutz Schröder for fruitful discussions.

References

Adámek, J., Herrlich, H. & Strecker, G. (1990), *Abstract and Concrete Categories*, Wiley, New York.

Alagić, S. & Bernstein, P. A. (2002), A Model Theory for Generic Schema Management, *in* 'Proc. of DBPL-01', Vol. 2397 of *LNCS*, Springer, pp. 228–246.

Baader, F. & Ghilardi, S. (2007), 'Connecting Many-Sorted Theories', *The Journal of Symbolic Logic* **72**(2), 535–583.

Bench-Capon, T. J. M. & Malcolm, G. (1999), Formalising Ontologies and Their Relations, *in* 'Proc. of DEXA-99', Vol. 1677 of *LNCS*, Springer, pp. 250–259.

Bidoit, M. & Mosses, P. D. (2004), CASL *User Manual*, LNCS Vol. 2900 (IFIP Series), Springer.

Borgida, A. & Serafini, L. (2003), 'Distributed Description Logics: Assimilating Information from Peer Sources', *Journal of Data Semantics* **1**, 153–184.

Codescu, M. & Mossakowski, T. (2008), Heterogeneous colimits, *in* F. Boulanger, C. Gaston & P.-Y. Schobbens, eds, 'MoVaH'08 Workshop on Modeling, Validation and Heterogeneity'.

CoFI (The Common Framework Initiative) (2004), CASL *Reference Manual*, LNCS Vol. 2960 (IFIP Series), Springer.

Cuenca Grau, B., Honavar, V., Schlicht, A. & Wolter, F., eds (2007), *2nd International Workshop on Modular Ontologies (WoMO-07)*, Vol. 315, CEUR Workshop Proceedings, (K-CAP) Whistler, BC, Canada.

Cuenca Grau, B., Horrocks, I., Kazakov, Y. & Sattler, U. (2008), 'Modular Reuse of Ontologies: Theory and Practice', *J. of Artificial Intelligence Research (JAIR)* **31**. To appear.

Cuenca Grau, B., Parsia, B. & Sirin, E. (2008), Ontology Integration Using $\mathcal{E}$-connections, *in* H. Stuckenschmidt & S. Spaccapietra, eds, 'Ontology Modularization', Springer. To Appear.

Diaconescu, R. (2002), 'Grothendieck institutions', *Applied Categorical Structures* **10**, 383–402.

Diaconescu, R. (2008), *Institution-independent Model Theory*, Studies in Universal Logic, Birkhäuser.

Euzenat, J. & Shvaiko, P. (2007), *Ontology Matching*, Springer, Heidelberg.

Goguen, J. A. (2005), 'Data, Schema, Ontology and Logic Integration', *Logic J. of the IGPL* **13**, 685–715.

Goguen, J. A. (2006), Information Integration in Institutions, *in* L. Moss, ed., 'Jon Barwise Memorial Volume', Indiana University Press. To appear.

Goguen, J. A. & Burstall, R. M. (1992), 'Institutions: Abstract Model Theory for Specification and Programming', *Journal of the ACM* **39**, 95–146.

Goguen, J. A. & Roşu, G. (2002), 'Institution morphisms', *Formal aspects of computing* **13**, 274–307.

Guerra, S. (2001), 'Composition of Default Specifications', *J. Log. Comput.* **11**(4), 559–578.

Haase, P., Honavar, V., Kutz, O., Sure, Y. & Tamilin, A., eds (2006), *1st International Workshop on Modular Ontologies (WoMO-06)*, Vol. 232, CEUR Workshop Proceedings, (ISWC) Athens, Georgia, USA.

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. & Sure, Y. (2005), A framework for handling inconsistency in changing ontologies, *in* 'Proc. of the 4th International Semantic Web Conference (ISWC-05)', Vol. 3729 of *LNCS*, Springer, pp. 353–367.

Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R. & Wang, H. H. (2006), The Manchester OWL Syntax, *in* 'OWL: Experiences and Directions'.

Horrocks, I., Kutz, O. & Sattler, U. (2006), The Even More Irresistible $\mathcal{SROIQ}$, *in* 'Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)', AAAI Press, pp. 57–67.

Kalyanpur, A., Parsia, B., Horridge, M. & Sirin, E. (2007), Finding all Justifications of OWL DL Entailments, *in* 'Proc. of ISWC/ASWC2007', LNCS Vol. 4825, Springer, pp. 267–280.

Konev, B., Lutz, C., Walther, D. & Wolter, F. (2008), Formal properties of modularization, *in* H. Stuckenschmidt & S. Spaccapietra, eds, 'Ontology Modularization', Springer.

Kutz, O., Lutz, C., Wolter, F. & Zakharyaschev, M. (2004), '$\mathcal{E}$-Connections of Abstract Description Systems', *Artificial Intelligence* **156**(1), 1–73.

Kutz, O. & Mossakowski, T. (2007), Modules in Transition: Conservativity, Composition, and Colimits, *in* '2nd Int. Workshop on Modular Ontologies (WoMO-07)'. K-CAP, Whistler BC, Canada.

Kutz, O. & Mossakowski, T. (2008), Conservativity in Structured Ontologies, *in* 'Proc. of the 18th European Conference on Artificial Intelligence (ECAI-08)', Patras, Greece. Forthcoming.

Kutz, O., Mossakowski, T. & Codescu, M. (2008), Shapes of Alignments: Construction, Combination, and Computation, *in* U. Sattler & A. Tamilin, eds, 'Proc of the 1st Workshop on Ontologies: Reasoning and Modularity (WORM-08)', CEUR-WS, Vol-348, ESWC, Tenerife, Spain.

Kutz, O., Wolter, F. & Zakharyaschev, M. (2002), Connecting abstract description systems, *in* 'Proc. of the 8th Conference on Principles of Knowledge Representation and Reasoning (KR-02)', Morgan Kaufmann, pp. 215–226.

Mac Lane, S. (1998), *Categories for the Working Mathematician*, 2nd edn, Springer, Berlin.

Madhavan, J., Bernstein, P., Domingos, P. & Halevy, A. (2002), Representing and reasoning about mappings between domain models, *in* 'Proc. of AAAI 2002', Edmonton, Canada.

Mossakowski, T. (2002), Comorphism-based Grothendieck logics, *in* 'Mathematical Foundations of Computer Science', Vol. 2420 of *LNCS*, Springer, pp. 593–604.

Mossakowski, T., Autexier, S. & Hutter, D. (2006), 'Development graphs—proof management for structured specifications', *Journal of Logic and Algebraic Programming* **67**(1–2), 114–145.

Mossakowski, T., Maeder, C. & Lüttich, K. (2007*a*), The Heterogeneous Tool Set, *in* O. Grumberg & M. Huth, eds, 'TACAS 2007', Vol. 4424 of *LNCS*, Springer, pp. 519–522.

Mossakowski, T., Maeder, C. & Lüttich, K. (2007*b*), The Heterogeneous Tool Set, *in* B. Beckert, ed., 'VERIFY 2007', Vol. 259, CEUR-WS.

Sannella, D. & Burstall, R. (1983), Structured theories in LCF, *in* 'Proc. 8th Colloq. on Trees in Algebra and Programming', Vol. 159 of *LNCS*, Springer, pp. 377–391.

Sattler, U. & Tamilin, A., eds (2008), *Workshop on Ontologies: Reasoning and Modularity (WORM-08)*, Vol. 348, CEUR Workshop Proceedings, ESWC, Tenerife, Spain.

Schorlemmer, M. & Kalfoglou, Y. (2008), 'Institutionalising Ontology-Based Semantic Integration', *Journal of Applied Ontology*. . To appear.

Wölfl, S., Mossakowski, T. & Schröder, L. (2007), Qualitative constraint calculi: Heterogeneous verification of composition tables, *in* 'Proc. FLAIRS 2007', pp. 665–670.

Zimmermann, A., Krötzsch, M., Euzenat, J. & Hitzler, P. (2006), Formalizing Ontology Alignment and its Operations with Category Theory, *in* 'Proc. of FOIS-06', pp. 277–288.

Appendix

A Heterogeneous Reasoning with HETS

A heterogeneous proof calculus based on development graphs, as they have been defined earlier in this paper, is implemented in the Heterogeneous Tool Set HETS (see (Mossakowski et al. 2007*a*,*b*)). It supports a multitude of logics, given as institutions, that can be used in formal specification. Examples are many-sorted first-order logic $\mathbf{FOL}^{\mathsf{ms}=}$ with equality underlying CASL and the description logic $\mathcal{SROIQ}(D)$ (and its sublogics $\mathcal{EL}$ and $\mathcal{ALC}$) underlying $\mathcal{OWL}$ 1.1. Several concrete syntaxes for $\mathcal{OWL}$ are supported, including Manchester Syntax (Horridge et al. 2006) and its extension to $\mathcal{SROIQ}(D)$. Moreover, relational schemes, as well as **QS5** (using syntax of the CASL language) are supported.

Between many of the logics, comorphisms are defined and implemented. For most of them, a comorphism into CASL exists, allowing heterogeneous specifications. A heterogeneous proof calculus (see (CoFI (The Common Framework Initiative) 2004)) for development graphs shifting proof obligations between nodes into nodes is implemented, too.

To prove obligations of a node (= in a single logic), several (first-order) **FOL** (`SPASS`, `Darwin`) and (higher-order) **HOL** (`Isabelle`) provers are integrated into HETS, enabling proof support for heterogeneously specified ontologies. The DL reasoner `Pellet` is supported as a consistency checker for ontologies.

The development graph calculus integrated into HETS can be used for all the integration, connection, and refinement techniques for ontologies discussed in this paper: the verification of semantic integrations as well as refinements are directly supported. In the case colimit computation is needed, as in W-alignments, this can be calculated via HETS' built-in colimit feature described in (Codescu & Mossakowski 2008), that also allows the 'approximation' of colimits.

A specific example for such a heterogeneous, multi-level reasoning is provided by the verification of the RCC composition tables (Wölfl et al. 2007). Here, the majority of proof obligations can be resolved by various, fully automatic reasoners. However, a proof obligation requiring the second-order theory of real numbers needs an interactive proof in the `Isabelle` prover.

B Heterogeneous Specifications in HETCASL

In the following, we present additional specifications for the interested reader.

B.1 Specification: From Ontology to Relational Scheme

We give the specification for Section 5.2. The specification `Biblio_DL` is written in HETDL, a concrete Syntax for $\mathcal{SROIQ}(D)$, that is very closely related to the Manchester Syntax (Horridge et al. 2006). Our language is almost self-explanatory, and follows the naming conventions of $\mathcal{OWL}$.

E.g., the stanza

```
Class: Researcher
SubclassOf: name some string
```

defines a class `Researcher` that is a subclass of all `Things` having a `name` of type `string`.

`Biblio_RS` is written in RELSCHEME, see Example 2. Following the keyword `Tables`, the signature of a specification is defined. It consists of a set of tables, each having columns of a particular data type. In `person(key id:pointer, name:string)`, a table with the name `person` is defined, having the columns `id` and `name` with `id` being the key of this table. If `key` occurs more than once in a table, we have a compound key. The sentences following the keyword `Relationships` define relations between different tables. The sentence `author_of[person] -> person[id] one_to_many` means that the column `person` of the table `author_of` is in `one_to_many` relationship with the column `id` of `person`.

The view `Biblio_DL_in_RS` is written in CASL. The link between `Biblio_DL` and `Biblio_RS` is established via several CASL sentences. Please note that the `view` here goes into the opposite direction compared to the one in Fig. 9.

```
logic DL
spec Biblio_DL =
  Class: Researcher
  SubclassOf: name some string

  ObjectProperty: hasArticle
  InverseOf: hasJournal

  Class: Article
  SubclassOf: author some Thing, title some string,
              hasJournal some Journal

  Class: Journal
  SubclassOf: name some string,
              hasArticle some Thing,
              impactFactor some integer
end

logic RelScheme
spec Biblio_RS =
  Tables
    person(key id:pointer, name:string)
    author_of(person, paper:pointer)
    paper(key id:pointer,title:string,
          published_in:pointer)
    journal(key id:pointer,name:string,
          impact_factor:integer)

  Relationships
    author_of[person]      -> person[id]  one_to_many
    author_of[paper]       -> paper[id]   one_to_many
    paper[published_in]    -> journal[id] one_to_many
end

logic CASL
view Biblio_DL_in_RS : Biblio_DL to
 { Biblio_RS with logic RelScheme -> CASL
  then %def
   preds Researcher(x:pointer)             <=>
            (exists n:string; a:pointer.
             person(x,n) /\ author_of(x,a));
         Article(x:pointer)                <=>
            (exists t:string; j:integer.paper(x,t,j));
         Journal(x:pointer)                <=>
            (exists n:string; i:float.journal(x,n,i));
         name(x:pointer;n:string)          <=>
            person(x,n);
         hasArticle(x,j:pointer)           <=>
            (exists n,t:string; i:integer .
             journal(x,n,i) /\ paper(j,t,x));
         hasJournal(j,x:pointer)           <=>
            (exists n,t:string; i:integer .
             journal(x,n,i) /\ paper(j,t,x));
         author(a,p:pointer)               <=>
            (exists t,n:string; p:pointer .
             paper(a,t,p) /\ person(p,n));
         title(a:pointer; t:string)        <=>
            (exists p:pointer . paper(a,t,p));
         impact_factor(j:pointer;f:integer) <=>
            (exists n:string . journal(j,n,f));
  }
end
```

Such a specification can be parsed and analysed by the tool HETS, which displays it as a development graph as in Fig. 11.

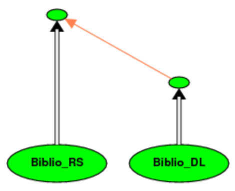


Figure 11: Development graph in HETS.

In this picture, red arrows are theorem links, and outlined thick black arrows are heterogeneous definition links. The picture shows the situation shown in the diagram of Fig. 10.

To prove this `view`, the development graph calculus has to be applied first. Then a prover needs to be invoked on the upper left node, that by now has become red: a node with proof obligations. The proof obligations can e.g. be proved with the SPASS theorem prover.

B.2 Specification: Integration through Interface and Colimit

In this section, an example specification for a W-alignment is given.

```
logic DL
spec Biblio_DL =
     Biblio_DL_Sign
then
  Class: Researcher
  SubclassOf: name some string

  ObjectProperty: hasArticle
  InverseOf: hasJournal

  Class: Article
  SubclassOf: author some Thing, title some string,
              hasJournal some Journal

  Class: Journal
  SubclassOf: name some string,
              hasArticle some Thing,
              impactFactor some integer

end
```

```
spec Bibli_DL_Sign =
  Class: Researcher
  DataProperty: name

  Class: Article
  ObjectProperty: author
  DataProperty: title
  ObjectProperty: hasJournal


  Class: Journal
  ObjectProperty: hasArticle
  DataProperty: impactFactor
end

logic RelScheme
spec Biblio_RS_Sign =

  Tables
    person(key id:pointer, name:string)
    author_of(person, paper:pointer)
    paper(key id:pointer,title:string,
          published_in:pointer)
    journal(key id:pointer,name:string,
          impact_factor:integer)
end

spec Biblio_RS =
     Biblio_RS_Sign
then
  Relationships
    author_of[person]      -> person[id]  one_to_many
    author_of[paper]       -> paper[id]   one_to_many
    paper[published_in]    -> journal[id] one_to_many
end

logic CASL
spec Interface =
   {Biblio_RS_Sign with logic RelScheme -> CASL}
and
   {Biblio_DL_Sign with logic DL -> CASL
    with Thing |-> pointer}
then
   forall a,j,p,x:pointer;n,t:string;f:integer
  . journal(j,n,f) <=> Journal(j) /\ name(j,n)
    /\ impactFactor(j,f)
  . paper(a,t,j) <=> Article(a) /\ Journal(j)
    /\ hasArticle(j,a) /\ title(a,t)
  . author_of(p,a) <=> Researcher(p) /\ Article(a)
    /\ author(p,a)
  . person(p,n) <=> Researcher(p) /\ name(p,n)
  . Researcher(x) <=> (exists q:pointer;m:string .
    person(x,m) /\ author(x,q))
  . Article(x) <=> (exists q:pointer;m:string .
    paper(x,m,q))
  . Journal(x) <=> (exists m:string;i:integer .
    journal(x,n,i))
end
```

The picture in figure 12 shows the development

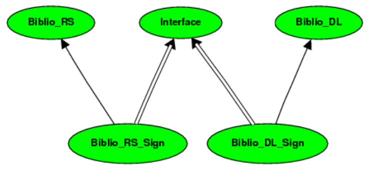


Figure 12: W-alignment in HETS.

graph of the above specification in HETS. In this figure, the outlined thick black arrows are heterogeneous definition links, while the normal black arrows are definition links. The picture shows the situation depicted in the diagram of Fig. 3. Its colimit can be automatically computed by HETS.

Supporting Coral Reef Ecosystems Research through Modelling Re-usable Ontologies.

Trina S. Myers[1], Ian Atkinson[1], Ron Johnstone[2].
School of Mathematics, Physics and Information Technology
James Cook University[1]
Townsville, Queensland.

[Trina.Myers, Ian.Atkinson]@jcu.edu.au

Centre for Marine Science,
University of Queensland[2]
Brisbane, Queensland.

RJohnstone@uq.edu.au

Abstract

The Semantic Reef project is an eco-informatics application that endeavours to combine multiple environmental datasets to test ecological hypotheses and derive information about environmental systems. The intention is to develop an automated data processing, problem-solving and knowledge discovery system to better understand and manage coral reef ecosystems.

Remote environmental monitoring (including sensor networks) is being widely investigated and deployed for collecting real-time data across widely distributed areas. As the volume of raw data increases, it is envisaged that bottlenecks will develop in the data analysis phases, where current data processing procedures still involve manual manipulation that will soon become unfeasible to manage. This 'data deluge' is being addressed in research communities, such as the Semantic Web and Knowledge Representation fields, through the development of the technologies and the use-case implementations such as described here. In an e-Research approach, highly diverse backgrounds and expertise were combined effectively to answer domain and locality specific hypotheses, such as coral bleaching, highlighting the value of a collaborative approach.

A set of re-usable ontologies, ranging from light to heavyweight, has been developed as separate components within the Knowledge Representation (KR) system, generically modelling a reef ecosystem. The ontology design aims to leverage the scalable characteristics of Semantic technologies, allowing for flexibility when posing hypotheses and support for future modifications to the domain due to new discoveries.

Keywords: Eco-informatics, Semantic Web, workflow, ontology, Climate Change, Semantic Reef.

1 Introduction

Eco-informatics is an emerging branch of e-Research whereby new IT techniques and infrastructure are being developed to give ecologists far greater scope for problem solving at otherwise impossible scales. With the predicted impact of climate change, and other environmental issues, these new cross-discipline tools are becoming more important.

Modern data gathering methods have seen the emergence of highly connected and efficient scientific instruments that produce vast quantities of data, sometimes termed as a "data deluge'(Hey and Trefethen, 2003). In the earth and marine sciences, environmental sensor networks are being deployed to gather data in real-time, across widely distributed areas, for applications such as environmental and seismic monitoring. The Integrated Marine Observing System (IMOS) - Great Barrier Reef Oceans Observing System (GBROOS) is one such infrastructure that uses sensor networks to remotely monitor chosen sites on the Great Barrier Reef (GBR) (IMOS, 2008). It is expected GBROOS and similar sensor network deployments, when fully implemented, will produce vast amounts of real-time streaming data from a variety of domain specific sensors, including meteorological, chemical and biological sensors. The vast amount of sensed data is necessary in order to produce the new information needed to answering the multitude of questions being posed in the scientific domains. This project aims to streamline the data analysis of remotely sensed data from the Great Barrier Reef.. A primary goal is to balance the inevitable growth in sensed data streams with the capacity of individual researchers to benefit from the volume of data available.

The Semantic Reef project is an application of eco-informatics where collaborations between disparate backgrounds and expertise have been effectively combined. Currently, a platform is being developed that combines emerging Semantic Web and Scientific Workflow technologies to access and infer information from unconnected datasets. On completion, the Semantic Reef model will produce a tool for hypothesis-driven research and problem-solving methods, which will allow for more efficient investigation methods of disparate data streams. Ultimately, coupling the disparate data to the

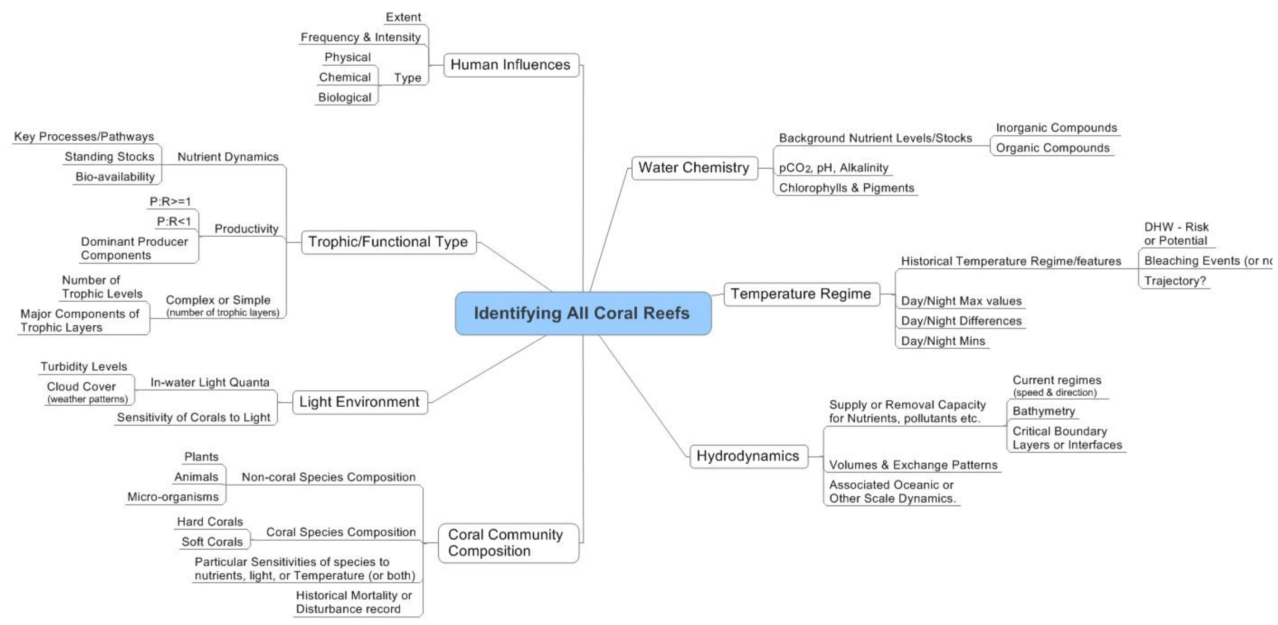

Identifying All Coral Reefs
Human Influences
Extent
Frequency & Intensity
Type
Physical
Chemical
Biological
Trophic/Functional Type
Nutrient Dynamics
Key Processes/Pathways
Standing Stocks
Bio-availability
Productivity
P:R>=1
P:R<1
Dominant Producer Components
Complex or Simple (number of trophic layers)
Number of Trophic Levels
Major Components of Trophic Layers
Light Environment
In-water Light Quanta
Turbidity Levels
Cloud Cover (weather patterns)
Sensitivity of Corals to Light
Coral Community Composition
Non-coral Species Composition
Plants
Animals
Micro-organisms
Coral Species Composition
Hard Corals
Soft Corals
Particular Sensitivities of species to nutrients, light, or Temperature (or both)
Historical Mortality or Disturbance record
Water Chemistry
Background Nutrient Levels/Stocks
Inorganic Compounds
Organic Compounds
pCO2, pH, Alkalinity
Chlorophylls & Pigments
Temperature Regime
Historical Temperature Regime/features
DHW - Risk or Potential
Bleaching Events (or no
Trajectory?
Day/Night Max values
Day/Night Differences
Day/Night Mins
Hydrodynamics
Supply or Removal Capacity for Nutrients, pollutants etc.
Current regimes (speed & direction)
Bathymetry
Critical Boundary Layers or Interfaces
Volumes & Exchange Patterns
Associated Oceanic or Other Scale Dynamics.

Figure 1 – Identifying any coral reef – Semantic building blocks

ontologies described herein, for querying and deriving inferences, may provide alerting for unusual domain-specific events.

One goal of the Semantic Reef is to improve our capacity to generate timely warnings of environmental conditions that are conducive to coral bleaching. A hierarchy of re-usable light to heavy weight ontologies has been developed to present marine scientists, from around the globe, tools to test hypotheses and probabilities through enlisting a semantic 'richness on demand' environment. The level of granularity required, or the depth of the analysis, is flexible, scalable by design, and independent of location, reef type or standing stock. The design purposefully separates the functionality and relationships, which are emulated in the integrated logic systems, in particular Description Logics (DL) and Propositional logics, from the instance data that make up a reef ecosystem. Thus, the modularity permits changes of minor and major entities, within the model, to be made trivially, effectively supporting future propositions, queries and even changes in the relationships themselves.

A description of the domain specific problems relating to reef systems will be followed by a brief overview of the goals and achievements of the Semantic Reef Project to date. Subsequently, the methodology, specific descriptions and justifications employed to create the hierarchy of ontologies, along with lessons learned and future works, is presented.

2 The Coral Reef Domain

Globally, marine ecosystems face many pressures from both natural and human-induced stresses and managing these threats requires major scientific insight and coordination on a global scale. Figure 1 depicts a domain expert's perspective of the core constituents required to make up any coral reef (Davey et al., 2008).

According to the Australian Institute for Marine Science (AIMS) threats to Australia's coral reefs fall into three categories:

- natural stresses that they have evolved to cope with for millions of years;
- direct anthropogenic pressures, including sediment and nutrient pollution from land run-offs, over-exploitation and damaging fishing practices, engineering and modification of shorelines; and
- global climate change.

Through studying the manifestations of global climate change in coral reefs, such as increased coral bleaching and coral disease, it has become clear that many threats are closely linked and exacerbate each other (AIMS, 2007). Due to its locality to us, and the close proximity of many major reef management and research facilities, the Great Barrier Reef (GBR) was an obvious choice as a use-case for trialling the Semantic Reef system. Furthermore, the outcome of a particular test, for example forecasting a coral bleaching event or a crown-of-thorns outbreak, could result in a useful tool for supporting management decisions on the reef.

The GBR is the largest coral reef system in the world and is a large contributor to Australia's economy supporting fishing and tourism activities. Covering an area of 348,000 square kilometres and spanning ~2300 kilometres of Queensland's, the GBR is sometimes referred to as the single largest organism in the world, although in reality it is made up of many billions of tiny coral formations (GBRMPA, 2006).

2.1 The Coral Bleaching Phenomenon

The impact of climate change on the GBR threatens this iconic wonder - climate change induced warming of ocean temperatures is a major contributor to the coral bleaching phenomenon (Hughes et al., 2003). Coral bleaching results from a stress condition in corals that induces a breakdown of the symbiotic relationship between coral and their symbiotic unicellular algae (zooxanthellae). The survival of both organisms is reliant on this symbiosis as the zooxanthellae reside in every cell of the coral animal's tissue and in exchange providing a major food source for the coral. The zooxanthellae produce energy-rich sugars for the corals via photosynthesis, and also give corals their brilliant colours from photosynthetic pigments. Reef corals are very sensitive to Sea Surface Temperatures (SST) outside their normal range. When the temperature elevates enough it can trigger stress factors in many dominant coral species, which causes the coral to expel the zooxanthellae leaving the white skeleton of the coral exposed, giving the bleached white appearance (Jones et al., 1998). Although spatially extensive, or mass, coral bleaching events have been largely attributed to anomalously high temperatures, bleaching can also be caused by other factors such as low-salinity, high-light intensity, pollutants, exposure, pH and even sedimentation (Hughes et al., 2003).

2.2 The Initial Validation Test Case

In the late summer (February and March) of 1998 and 2002, two mass bleaching events occurred in the GBR and Coral Sea off the coast of Australia. The work and studies carried out on these events offered relevant building blocks and information in which to populate the ontologies used herein. In particular, the stress factor most commonly associated with bleaching of elevated sea surface temperature (Marshall and Schuttenberg, 2006), so data was acquired through the Great Barrier Reef Marine Park Authority (GBRMPA) for use in the initial validation test. The metrics currently used to forecast possible bleaching events include, among others, the SST anomaly (SST+) and Degree Heating Days (DHD). Where SST+ is calculated as the number of °C above the Long-term Mean Summer Temperature (LMST) for that location and a DHD describes the accumulation of thermal stress. Where one DHD is calculated as one degree above the local LMST for one day and two DHDs is either two degrees above the local LMST for one day, or one degree above LMST for two days and so forth. The validation was carried out using a reverse hypothesis method which entailed ground-truthing the outcome of inference rules, which mimicked bleaching indices, against actual historical events, data and research from the 1998 and 2002 mass bleaching events (Myers et al., 2007a).

3 Background – The Semantic Reef Project

The Semantic Reef Project is a platform designed as a hypothesis tool for research on reef eco-systems (Myers et al., 2007b). This Knowledge Representation (KR) system will employ Semantic Web technologies and scientific workflow tools to access a range of datasets for automated processing (refer Figure 2). The data is routed through the workflow and mapped to the ontologies contained in the knowledge base for querying and/or hypothesis testing. Ultimately, this system may address the issue of efficient and effective data analysis, which is a dilemma being faced by many disciplines from the onslaught of vast amounts of new information. Initially, the focus domain is in the field of marine biology, where the first trials of the system involved hind-casting coral bleaching events (Myers et al., 2007a).

Current research in coral reef ecology spans a diverse range of environmental and chemical studies. In order to test and validate the system, datasets were needed along with a well researched topic.. The coral bleaching problem was an obvious test case for the ground truthing of the Semantic Reef architecture as the datasets that were relevant to the historic findings, and observations by experts in the field, were readily available.

3.1 The Semantic Web Component

Semantic Web technologies, allow for a flexible scalable environment to model worldly concepts in a way that is understandable to the machine (Antoniou and van Harmelen, 2004). Ontologies are the foundation of these technologies, and within the context of this paper, are descriptions to represent both abstract and specific concepts. Where declarations of a domain's vocabulary and the terms that describe the entities within and the relationships they have with each other, are defined using axioms and restrictions that constrain the interpretation for use by the computer, making the concept machine-understandable (Guarino, 1997).

Once data and information are 'understandable' or interpretable by a computer, Semantic technologies enable the machine to make intelligent decisions based on conclusions derived through logic systems such as Description Logics (DL) and inference rules (Antoniou and van Harmelen, 2004). DL are sets of logical statements to describe relationships and parameters of a concept and can be used to reason over classes and individuals to infer assumption and subsumption, thus allowing for negation/complement of classes, disjoint information and existential and universal quantification. The DL reasoning engines used in this project are Pellet (Mindswap, 2007), Racer PRO (RacerPRO, 2007) and FaCT++ (FaCT++, 2008) as implemented through the Protégé ontology editor (Protégé, 2007).

An environment that supports suppositions is enabled with utilities such as the Semantic Web Rules Language (SWRL). SWRL manages inference using horn-like rules, which is a subset of predicate logic (first-order logic), and is orthogonal to description logic. The SWRL

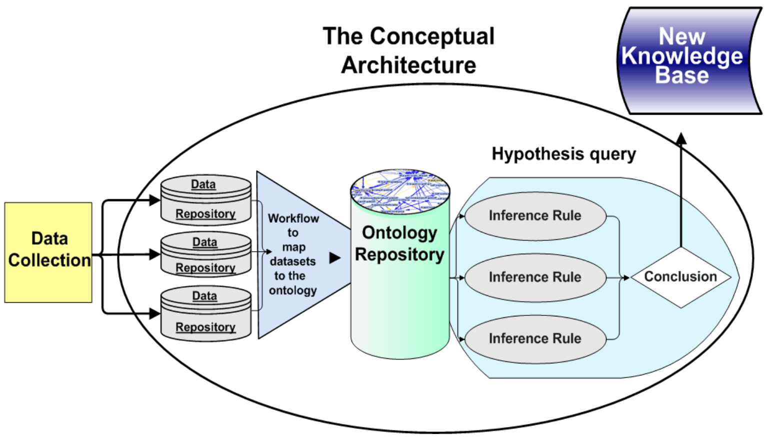


Figure 2 – The Semantic Reef workflow.

rules are passed to the Jess inference engine (Jess, 2006) for inferring over the ontology instances, directly or using the 'SWRL to Jess bridge' component of Protégé (O'Connor et al., 2005).

The Web Ontology Language (OWL), which is the W3C's recommended specification (McGuinness and van Harmelen, 2004), has three different varieties: OWL Lite, OWL DL and OWL Full, each providing different levels of logical expressiveness. The logical semantics of OWL DL (and Lite, which is a subset of DL) are based on DL, therefore, all inferences available in an OWL-Lite or OWL DL ontology can be computed using the reasoning engine. Ontologies in OWL Full, however, are not decidable, and have insufficient application reasoning support available. Therefore, to maximise the full richness of using logic systems to automate the reasoning and inference functions, the ontology design within this project are maintained in OWL DL or Owl Lite.

3.2 The Work Flow Component

In this application, we use Kepler (Atkinson et al., 2007) to automatically process raw data, and pass the results onto the ontology. Scientific workflow technologies and tools, such as the Kepler system used here, are adaptive software programs to capture complex analyses in a flow of which the data is taken through one analytical step after another (Altintas et al., 2004). It provides access to the continually expanding amount of geographically distributed data repositories, computing resources, and workflow libraries.

Currently, the processing capabilities of the expert system are being tested to infer a coral bleach warning with Sea Surface Temperature (SST) data. The data used includes spatial data from the. National Oceanic and Atmospheric Administration (NOAA) (Gleeson and Strong, 1995) and real time SST data being streamed directly from the Davies Reef GBROOS site (Kininmonth et al., 2004). Each workflow step is represented by a Kepler "actor". An example workflow would be to create an actor to tag streaming data with a URI, then map to the 'Environmental Element' ontology, which includes a 'Temperature' class, with 'Sensor Network Temperature' and 'Spatial Temperature' subclasses. The actors would then be directed to populate the 'Sensor Network Temperature' class instances with date/time and temperature data type property values. The workflow actors would also compute menial operations such as determining the SST anomaly, which would be passed to an instance of the 'Named Coral Reef' class as the value of the appropriate property.

Ultimately the knowledge base will be filled with relevant data from diverse sources on real time salinity, $C0_2$, pH levels, bathometry information, standing stocks, etc. Where knowledge may then be derived from the data by questioning for semantic correlation and analysis through the utilization of logic systems, such as DL and inference rules, from within the ontology (refer Figure 2). By inferring patterns from data, such questions may help to inform research scientists on many issues other then coral bleaching events, such as rising acidity, crown of thorn starfish outbreaks, etc.

3.3 Results to Date

The semantic reef project has evolved through a validation stage that has tested the accuracy of the model. In order to accomplish this, SST and community make-up data from the 1998 and the 2002 bleaching events were entered into the KR system. Using DL to infer relationships, for instance, between the community standing stocks (e.g. Acropora from one database is equivalent to Staghorn coral in another) or trophic layers (e.g. a animal that eats both plant and animal is inferred to the omnivore class). Inference rules were then applied to detect a problematic area based on the use of the two coral bleaching metrics; SST+ and DHDs. The rules inferred any instances of a particular reef to belong to a categorised bleach watch class automatically. To illustrate a simplified rule, written in Semantic Web Rules Language (SWRL), to assess the SST anomaly and the presence of zooxanthellae in the coral species (i.e. hermatypic), was as follows:

```
Environmental_Element(?x) ^
Coral_Reef(?z) ^
hasEnvironmentOf(?z, ?x) ^
SST_anomaly(?x, ?anomalyTot) ^
swrlb:greaterThan(?anomalyTot, 2.5) ^
swrlb:lessThanOrEqual(?anomTot, 3.5) ^
Coral(?y) ^ hasPart(?z, ?y) ^
isHermatypic(?y, true) ^
→
Category_3_Bleached_Reef(?z)
```

The ability to infer a bleaching alert autonomically which matched the historic outcomes for that reef and timestamp verified a successful test (Myers et al., 2007a).

It must be noted that bleaching is not uniform, but instead occurs in discreet reefs across the many that comprise the GBR, not to mention other reef systems around the world. At present there is still only a limited understanding of the causal factors of bleaching, although sea temperatures are clearly involved. Since the initial validation exercise, a richer set of ontologies have been developed and are described herein. These ontologies represent the functionality and complexity of a coral reef, including the composite population of the reef community, human influence contributors, hydrodynamics, bathometry, environmental factors, among many others. The design is modelled in a bottom-up fashion, from light to heavy-weight ontologies; therefore, with each level of granularity, the individual ontologies are designed to serve a more specific purpose. Using this modular approach, reusability is afforded, where the knowledge being sought or the hypothesis being posed will determine the choice of which ontologies to utilise.

4 Ontologies to represent any Reef Ecosystem

4.1 The Importance of the Open World Assumption

Semantic technologies can be used to better model the world as it is currently known, as there is support for added flexibility through the Open World Assumption (OWA). The OWA means what cannot be proven to be true is not automatically false, it assumes that its

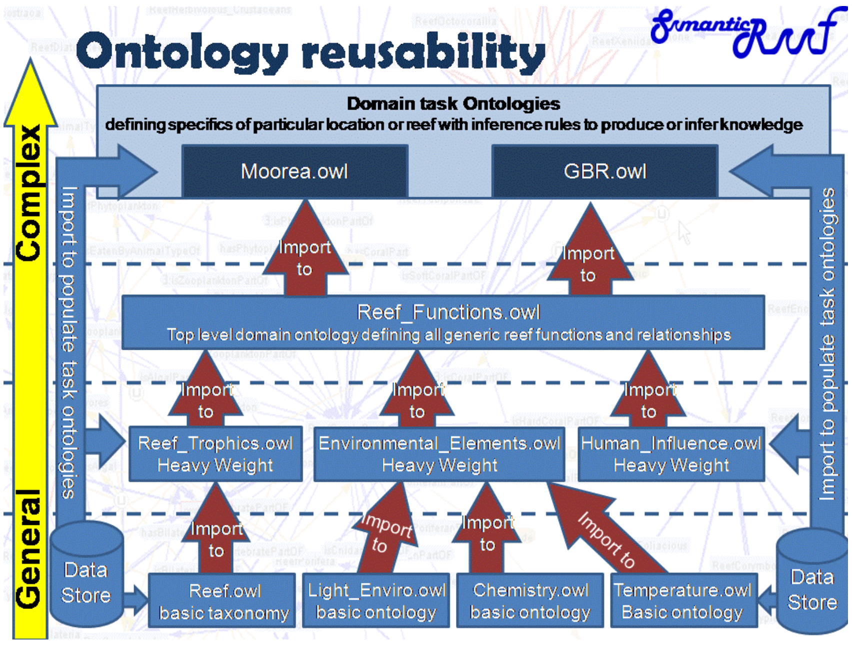

Ontology reusability
SemanticReef
Complex
General
Domain task Ontologies
defining specifics of particular location or reef with inference rules to produce or infer knowledge
Moorea.owl
GBR.owl
Import to
Import to
Reef_Functions.owl
Top level domain ontology defining all generic reef functions and relationships
Import to
Import to
Import to
Reef_Trophics.owl Heavy Weight
Environmental_Elements.owl Heavy Weight
Human_Influence.owl Heavy Weight
Import to
Import to
Import to
Import to
Data Store
Reef.owl basic taxonomy
Light_Enviro.owl basic ontology
Chemistry.owl basic ontology
Temperature.owl Basic ontology
Data Store
Import to populate task ontologies
Import to populate task ontologies

Figure 3 – The bottom-up ontology design – from light weight to heavy weight.

knowledge of the world is incomplete (Horrocks et al., 2003). Therefore, what is not stated is considered unknown, rather than wrong; it simply assumes the extra information needed has not, as yet, been added to the knowledge base (Rector et al., 2004).

Science, in general, moves forward by maintaining an OWA, that is, nothing is false until explicitly proven thus. Traditional data base implementations such as relational data bases are required to maintain a closed world assumption, that is, everything that is known about the world exists within the boundaries of the data base. This creates a mismatch between the researchers need to remain open to new discoveries and the current technological capabilities for flexibility and scalability.

When there is a change in relationships within a conceptual world due to a new discovery or fact, which is quite feasible in a marine biology domain, modifications to these relationships can be easily accomplished in OWL ontologies, as they are designed with an OWA. For example, describing the makeup of a particular reef includes statements such as:

Davies_Reef $\sqsubseteq$

hasCoral $\exists$ Favites $\sqcup$

hasCoral $\exists$ Porites

Where a query of whether Davies Reef contains any Acropora would return an unknown. It is assumed in an OWA, that not all is known of the world, therefore unless there is a statement that explicitly declares ‘Davies_Reef hasCoral $\neg$ $\exists$ Acropora’ the reasoning engine will conclude there may be Acropora, it is just not asserted. Where the same query run in an environment that supports the closed world assumption (CWA), such as relational databases, the result would be the negative statement that Davies Reef does not contain Acropora, where in reality it does.

In the marine biology domain there is a great need for multi-scalability. As the relationships within an ecosystem can change depending on research findings and/or defeasible current knowledge, the capacity for flexible modifications in lieu of new developments is very important. The OWA allows for a certain degree of flexibility when applying queries across domains where not all is explicitly proven knowledge. An OWA is considered implicit, that is, every tuple not explicitly contained in the ontology is implicitly assumed to represent a fact that is unknown, rather than false.

4.2 The bottom-up design methodology

Concepts can be modelled through ontologies in a variety of ways and varying degrees of granularity. The most common types of ontologies are the general or common ontologies, top-level or upper-level ontologies, domain ontologies, task ontologies, domain-task ontologies, method ontologies and application ontologies all serving different functions. The type of ontology to build is determined purely by the extensibility and expressiveness required, in conjunction with what information or knowledge the ontology is designed to produce. The more complex the ontology, the higher the restriction on flexibility and reusability (Gomez-Perez et al., 2004).

Clearly reefs are highly complex, interdependent ecosystems. It became apparent the use of a singular top-down, large ecosystem ontology would not be simple to create, implement nor maintain, as the number of variables and multi-scale relationships are immense. Therefore, a ‘mix and match’ fashion in the ontological design is adopted within this project’s architecture. The decision for this design method aimed to maximise the advantages the Semantic Web technologies offers, such as flexibility, scalability and rich computer-understandable descriptive languages. That is, the main ontologies within the project were segmented in a ‘bottom-up’ design methodology, where the support available for importation and re-usability of pre-defined ontologies within OWL was employed. The current set of ontologies, shown in figure 3, consist of reef and environmental instance taxonomies at the lowest layers, which import to heavier-weight DL trophic layer and environmental element ontologies, for richer relational descriptions. Complexity in relationships is added at each higher level, through to specific domain-task ontologies at the highest layer. Dependant on the hypothesis to be posed, these task-specific and method ontologies will employ all, or some, of the general and domain ontologies beneath. Where finally, finely detailed inference rules can be written to the system, as propositions, to infer conclusions about a specific issue on a particular reef, regardless of location, as the re-usable ontologies can be populated with instance data pertinent only to that locale (e.g. coral spawning events on the Moorea Island reefs in French Polynesia or a coral bleaching watch on the GBR in Australia).

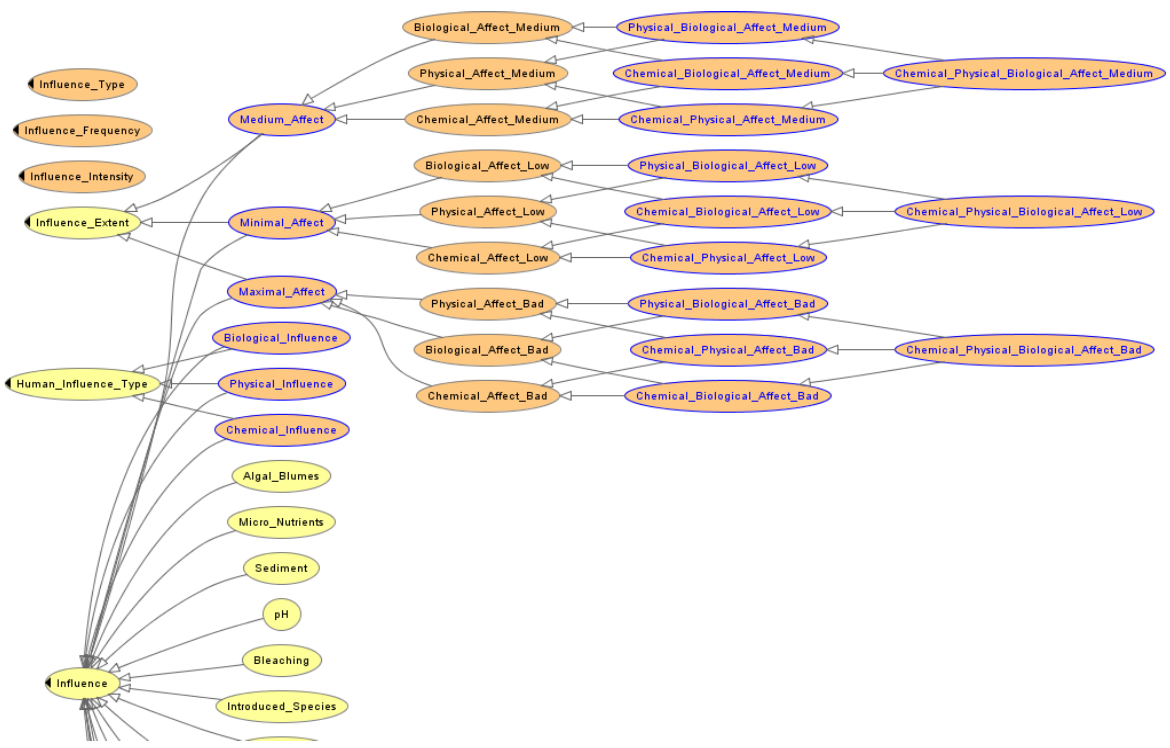


Figure 4 – The inferred relationships after classification of the 'human influence' ontology.

4.2.1 The light-weight ontologies

The bottom level reef ontology is designed for maximum reusability. It describes a taxonomy written in OWL-Lite that lists the vocabulary of the coral reef domain in both scientific and common names. The only constraints used for this hierarchical classification level are the owl:equivalentClass, which explicitly states the equality relationship between the phylum and family names with the common names. Adding in the structured scientific names as well as the common names will allow for flexibility in the disparate datasets used to populate this ontology with standing stock. That is, accessing prewritten species databases that are listed in different perspectives (i.e. common names versus phylum, etc.) will be inconsequential, as the reasoning engine will subsume instances to multiple superclasses. For example, 'Scaridae' in one database, such as the CSIRO Data Centre (CSIRO, 2008) might be 'parrotfish' in another, where upon classification, the instances will belong to both classes automatically. Bridging different schemas is imperative, as any conclusions should be based on all relevant, available data. In addition, the end users perspectives, needs and knowledge of the domain, and the types of queries and inferences required, could be vastly different or unknown, at this time.

The three main quantifiable environment elements are the temperature regime, water chemistry and light quanta. These elements are depicted in common ontologies for access by the Kepler workflow in order to populate with temporal instances, such as, the SST, nitrogen levels, or pH reading at a given timestamp taken from the GBROOS sensors and/or salinity from the Australian Bureau of Meteorology's BLUElink (GBROOS, 2007, BLUElink, 2008). Upon population, these lightweight ontologies can then all be imported into the more complex environmental ontology, which explicitly defines the relationships between the elements, for instance certain domain-specific metrics such as DHDs. This effectively separates the instance data from the logical descriptions, supporting the goal of reusability, as the environmental elements will be different dependant on time and location.

4.2.2 The heavy-weight ontologies

There are many principles within the Semantic technology standards that must be adhered to in order to employ all of the complex logic levels available within the languages and current tools. For example, in order for the ontology to remain straightforward, for reasons of reusability, without the added complexity of the logic systems, it is prudent to maintain a lightweight taxonomy (i.e. written using RDF or OWL-Lite). This limits the reasoning ability by the classifiers, to automate the inferred subsumption of classes and/or instances, as the richer descriptions of concepts using axiomisations, quantifications available in DL, are not available at this level. Therefore when describing the intricate, multi-scale relations involved in a coral reef ecosystem, one such example being the 'Human_Influence' ontology (Figure 4) or the Reef_Trophics' ontology, OWL-DL was used.

Engineering an ontology to describe human influences on coral reefs is not a trivial task, especially when designing for scalability and flexibility to allow for future unknown queries, propositions or additions. Coral reefs can be affected by a variety of human influences, which can be categorised as biological, physical or chemical. The 'Human_Influence' ontology was fashioned as a matrix consisting of occurrences and the levels of severity for each of these influence types. The main factors of each influence type are intensity, frequency and extent,

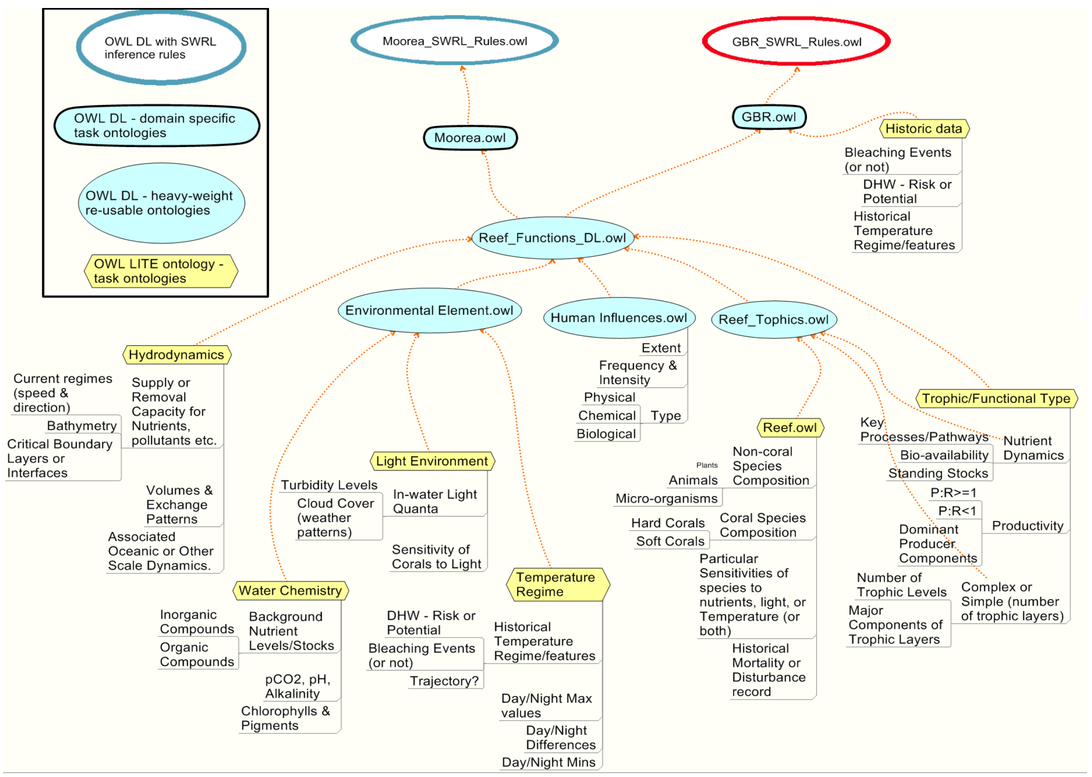

OWL DL with SWRL inference rules
OWL DL - domain specific task ontologies
OWL DL - heavy-weight re-usable ontologies
OWL LITE ontology - task ontologies
Moorea_SWRL_Rules.owl
GBR_SWRL_Rules.owl
Moorea.owl
GBR.owl
Historic data
Bleaching Events (or not)
DHW - Risk or Potential
Historical Temperature Regime/features
Reef_Functions_DL.owl
Environmental Element.owl
Human Influences.owl
Reef_Tophics.owl
Hydrodynamics
Current regimes (speed & direction)
Bathymetry
Critical Boundary Layers or Interfaces
Supply or Removal Capacity for Nutrients, pollutants etc.
Volumes & Exchange Patterns
Associated Oceanic or Other Scale Dynamics.
Light Environment
Turbidity Levels
Cloud Cover (weather patterns)
In-water Light Quanta
Sensitivity of Corals to Light
Water Chemistry
Inorganic Compounds
Organic Compounds
Background Nutrient Levels/Stocks
pCO2, pH, Alkalinity
Chlorophylls & Pigments
DHW - Risk or Potential
Bleaching Events (or not)
Trajectory?
Temperature Regime
Historical Temperature Regime/features
Day/Night Max values
Day/Night Differences
Day/Night Mins
Extent
Frequency & Intensity
Physical
Chemical
Biological
Type
Reef.owl
Plants
Animals
Micro-organisms
Non-coral Species Composition
Hard Corals
Soft Corals
Coral Species Composition
Particular Sensitivities of species to nutrients, light, or Temperature (or both)
Historical Mortality or Disturbance record
Trophic/Functional Type
Key Processes/Pathways
Bio-availability
Standing Stocks
Nutrient Dynamics
P:R>=1
P:R<1
Dominant Producer Components
Productivity
Number of Trophic Levels
Major Components of Trophic Layers
Complex or Simple (number of trophic layers)

Figure 5 – The hierarchical import methodology

which were expressed so each instance could be categorised and subsumed to a degree of extent from minimal to maximal severity.

Specifically, the ‘Intensity’, ‘Frequency’ and ‘Influence_Type’ classes of the ‘Human_Influence’ ontology were created as enumerated classes. Where ‘Intensity’ and ‘Frequency’ were filled with individuals from low to high, and the ‘Influence_Type’ class was filled with predefined individuals of biological, physical and chemical. Subsequently, the minimal to maximal affects were created as subclasses of Influence_Extent’, where class axioms of *necessary* and *necessary and sufficient* conditions were explicitly stated to allow for the automated reasoning (Figure 4). For example, to belong to the class ‘Chemical_Biological_Affect_Bad’, a subclass of ‘Maximal_Affect’ the conditions state:

Chemical_Biological_Affect_Bad ⊑
hasHumanInfluence ∃ Biological_Influence
hasHumanInfluence ∃ Chemical_Influence
hasHumanInfluence ∀ (Biological_Influence ⊔
Chemical_Influence)
hasInfluenceType ∋ Biological
hasInfluenceType ∋ Chemical
hasFrequencyOf ∋ High_Frequency
HasIntensityOf ∋ High_Intensity

On populating the Influence class with instances, properties such as ‘hasIntensityOf’ and ‘hasFrequencyOf’ are filled, dependant on the pre-processing triggers from the Kepler workflow (e.g. a highly frequent value can be determined as the data is presented in real-time). Therefore, following classification by the reasoning engine, all influence instances (such as a dredging, scientific experiment, pollution, flumes, etc) will be automatically categorised to the severity of the affect.

5 An Example Hypothesis

The capability of validating ecological hypotheses across wide geographic domains are still mostly in development stages. The KR system described here would enable such propositional testing by using Semantic Web Rules Language (SWRL) inference rules. The domain task ontologies are introduced, at the highest level, to perform tasks such as these types of queries or proposing hypotheses, for any reef, with any mix of standing stock composition, hydrodynamics, environmental elements or external human influences.

By importing all, or a subset of, the other ontologies, rules can be designed as inductive, deductive, reactive or production rules. For instance, exploring the unknown casual effects of a coral bleaching event could be researched through inductive reasoning. Where, by posing antecedents that are believed to be responsible for the occurrence would produce a conclusion which would allow for the proving or disproving of the hypothesis by observation (Figure 5). For example, the following basic inference rule checks for any instances that have a higher than average SST, salinity level and whether there is a human influence involved, which is dredging in this case:

```
        Coral_Reef(?z) ^
    Environmental_Element(?x) ^
     hasEnvironmentOf(?z, ?x) ^
 Reef:hasLongitudeOf(?z, ?ReefLong) ^
  Reef:hasLatitudeOf(?z, ?ReefLat) ^
    SST_anomaly(?x, ?anomalyTot) ^
 swrlb:greaterThan(?anomalyTot, 2.5) ^
swrlb:lessThanOrEqual(?anomTot, 3.5) ^
     Coral(?y) ^ hasPart(?z, ?y) ^
       isHermatypic(?y, true) ^
    hasSalinity(?x, ?salinityPcnt) ^
swrlb:greaterThan(?salinityPcnt, 42) ^
```

```
Human_Influence:Dredging(?a) ^
Influence:hasLongitudeOf(?z, ?HILong)^
Influence:hasLatitudeOf(?z, ?HILat) ^
swrlb:equal(?ReefLat, ?HILat) ^
swrlb:equal(?ReefLong, ? HILong) ^
hasIntensityOf(?a, HIGH)
→  Bleached_Watch(?z)
```

Any instance deduced to the bleach watch class would be observed *in situ* for a sign of a bleaching incident.

6 Future Works and Conclusions

The demand for automatic data analysis and hypothesis testing is emerging. This need for autonomy grows increasingly vital as the escalating range of data gathering devices and instruments are deployed. Presently it is becoming increasingly difficult for individual researchers to deal with the growing amount of data available to them. However, assimilating the mass data being gathered is vitally important for the management and research of our reef systems if they are to be resilient to, or even survive, the effects of climate change. Coral bleaching is one such impact of climate change and although the stress factor most commonly associated with bleaching is elevated sea temperature, for which there is a variety of instruments deployed to gather this spatially varied data, there are a number of other causal factors. These additional stressors, such as high light intensity, low salinity and pollutants, are known to exacerbate coral bleaching (Hughes et al., 2003), however the information to confirm these findings is limited by the finite historical data available. Hence the requirement for real time data in combination with an efficient processing tools.

The modular ontology design used within the project may assist streamlining the data analysis phase. Here, diverse backgrounds and expertise were combined effectively in a collaboration to answer domain-specific and locality-specific hypotheses. As user driven propositions change due to new findings, information or queries, and new data sources become available, the open world nature of Semantic technologies, in particular OWL, allows for additions and eliminations, such as new discoveries, to be trivially met.

The Semantic Reef project is a new approach to such data analysis and interpretation. The project will be extendable to hypothesise over many areas and issues in environmental monitoring and climate change. The importance of research on these issues is vital, as the sustainability and survival of our reef ecosystems are indicators to the future of the global ecosystem.

To achieve these important goals cross-discipline collaborations are imperative. In an eco-informatics application, which by definition implies the binding of diverse disciplines, such as discussed here, there are many hurdles in bridging the collective knowledge required. Within the scope of the Semantic Reef project, it was necessary to have an understanding of both Semantic Web tools and technologies (e.g. ontology engineering, logic systems, etc) combined with knowledge of the environmental systems of interest (i.e. reef ecosystems). The undertaking of learning both semantic and logic systems, as well as becoming conversant with an extremely complex ecological science, is demanding, and to attain the level of expertise required in either field, to realise the potential of this project, is difficult. The obvious solution to overcoming this multi-disciplinary dilemma is adopting and fostering the collaborative research model. Here, disparate backgrounds and expertise have been combined effectively and, in this case, one potential end user, the Great Barrier Reef Marine Park Authority was involved.

Using 'cutting-edge' technologies, by its very nature, has a degree of risk as the developers of the technologies work with the users of these technologies to overcome difficulties as they arise. The evolution of Semantic technologies, in particular ontology design and engineering, is very rapid, and openly supported by a highly active community that strive for continual improvement in functionality as use-case applications, such as described here, develop.

7 Acknowledgements

We thank Scott Bainbridge and Stuart Kininmonth from the Australian Institute of Marine Science (AIMS) and Sally Holbrook and Russell Schmitt from the Coral Reef Environmental Observatory Network (CREON) group for their interest, helpful discussions and feedback and Climate Change Response group at the GBRMPA for their role. We also thank the anonymous reviewers for their valuable comments and advise.

8 References

AIMS: Australian Institute of Marine Science. http://www.aims.gov.au/. Accessed June 2007.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B. and S. Mock (2004): Kepler: An Extensible System for Design and Execution of Scientific Workflows,. *16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04)*, Santorini Island, Greece, 21-23,

Antoniou, G. and van Harmelen, F. (2004): *A Semantic Web Primer (Cooperative Information Systems)*, The MIT Press.

Atkinson, I. M., du Boulay, D., Chee, C., Chiu, K., Coddington, P., Gerson, A., King, T., McMullen, D., Quilici, R., Turner, P., Wendelborn, A., Wyatt, M. and Zhang, D. (2007): Developing CIMA-based cyberinfrastructure for remote access to scientific instruments and collaborative e-research. *Proceedings of the fifth Australasian symposium on ACSW frontiers - Volume 68*, Ballarat, Australia, Australian Computer Society, Inc.

BLUElink: Australian Bureau of Meteorology - BLUElink Ocean Forecasting Australia. http://www.bom.gov.au/bluelink/. Accessed April 2008.

CSIRO: Data Centre. http://www.marine.csiro.au/datacentre/. Accessed April 2008.

Davey, M., Holmes, G. and Johnstone, R. (2008) High rates of nitrogen fixation (acetylene reduction) on coral skeletons following bleaching mortality. *Coral Reefs*, 27, 227-236.

FaCT++: Fast Classification of Terminologies Descriptioin Logic classifier. http://owl.man.ac.uk/factplusplus/. Accessed May 2008.

GBRMPA: Great Barrier Reef Marine Park Authority. Australian Government. http://www.gbrmpa.gov.au/. Accessed May 2006.

GBROOS: Great Barrier Reef Ocean Observing System. http://www.imos.org.au/nodes/great-barrier-reef-observing-system.html. Accessed June 2007.

Gleeson, M. W. and Strong, A. E. (1995) Applying MCSST to coral reef bleaching. *Advances in Space Research,*, 16**,** 151-154.

Gomez-Perez, A., Corcho, O. and Fernandez-Lopez, M. (2004): *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing)*, Springer.

Guarino, N. (1997) Understanding, building and using ontologies. *Int. J. Hum.-Comput. Stud.*, 46**,** 293-310.

Hey, A. J. G. and Trefethen, A. E. (2003): The Data Deluge: An e-Science Perspective. IN *Grid Computing - Making the Global Infrastructure a Reality*. Berman, F., Fox, G. C. and Hey, A. J. G. (Ed.). Wiley and Sons.

Horrocks, I., Patel-Schneider, P. F. and van Harmelen, F. (2003) From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics,*, 1**,** 7-26.

Hughes, T. P., Baird, A. H., Bellwood, D. R., Card, M., Connolly, S. R., Folke, C., Grosberg, R., Hoegh-Guldberg, O., Jackson, J. B. C., Kleypas, J., Lough, J. M., Marshall, P., Nyström, M., Palumbi, S. R., Pandolfi, J. M., Rosen, B. and Roughgarden, J. (2003) Climate Change, Human Impacts, and the Resilience of Coral Reefs. *Science*, 301**,** 929 - 933.

IMOS: Integrated Marine Observing System (IMOS). http://imos.org.au/. Accessed June 2008.

Jess: The Rule Engine for the Java Platform. http://herzberg.ca.sandia.gov/jess/. Accessed July 2006.

Jones, R., Hoegh-Guldberg, O. and Larkum, A. (1998) Temperature-induced bleaching of corals begins with impairment of the CO2 fixation mechanism in zooxanthellae. *Plant Cell and Environment*, 21**,** 1219-1230.

Kininmonth, S., Bainbridgea, S., Atkinson, I., Gilla, E., Barrald, L. and Vidaude, R. (2004) Sensor networking the Great Barrier Reef. *Spatial Sciences Qld***,** 34-38.

Marshall, P. and Schuttenberg, H. (2006): *A Reef Manager's Guide to Coral Bleaching*, Townsville, Australia, Great Barrier Reef Marine Park Authority.

McGuinness, D. and van Harmelen, F.: OWL Web Ontology Language overview. W3C. http://www.w3.org/TR/owl-features/. Accessed May 2006.

Mindswap: Pellet: The open source OWL DL reasoner. Maryland Information and Network Dynamics Lab Semantic Web Agents Project. http://pellet.owldl.com/. Accessed July 2007.

Myers, T., Atkinson, I. and Maynard, J. (2007a): The Semantic Reef: An eco-informatics approach for modelling coral bleaching within the Great Barrier Reef. *ERE Environmental Research Event* Carins, QLD, Environmental Research Event Organising Committee.

Myers, T. S., Atkinson, I. M. and Lavery, W. J. (2007b): The Semantic Reef: Managing Complex Knowledge to Predict Coral Bleaching on the Great Barrier Reef. *Fifth Australasian Symposium on Grid Computing and e-Research (AusGrid 2007)*, Ballarat, Australia, **68**:59-67, ACS.

O'Connor, M. J., Knublauch, H., Tu, S. W., Grossof, B., Dean, M., Grosso, W. E. and Musen, M. A. (2005): Supporting Rule System Interoperability on the Semantic Web with SWRL. *Fourth International Semantic Web Conference*, Galway, Ireland, ISWC-2005.

Protégé: The Ontology Editor and Knowledge Acquisition System. Stanford University. http://protege.stanford.edu/. Accessed July 2007.

RacerPRO: Racer Systems. http:/www.racer-systems.com. Accessed June 2006.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H. and Wroe, C. (2004): OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. *Proceedings of the European Conference on Knowledge Acquistion (EKAW 2004)*, Northampton, England, **3257**:63-81, Springer-Verlag.

Ontology Evolution for Customer Services

Tho T. Quan[1] and Thai D. Nguyen[2]
[1]Faculty of Computer Science and Engineering, Hochiminh City University of Technology
Hochiminh City
Vietnam

qttho@cse.hcmut.edu.vn
[2]Department of Information Technology, Nguyen Tat Thanh College
Hochiminh City
Vietnam

ndthai@ntt.edu.vn

Abstract

Customer service support has become an integral part of many multinational manufacturing companies which produce insertion and surface mount machines in the electronics industry. With the recent advancement of the Semantic Web, many attempts have been made to provide customer support over the Semantic Web environment.

Ontology is the major factor to represent knowledge on the Semantic Web. Therefore, a specific ontology so-called Machine Ontology is required to render customer service support with precise semantics. Machine Ontology can be generated either manually by human experts or automatically by intelligent programs. Even though manual ontology generation is highly accurate, it is tedious, time-consuming and requires high cost in terms of man power. On the other hand, automatic generation of ontology can provide high level of knowledge details and be effective when dealing with large-scaled dataset. However, automatically generated ontology may be not very semantically correct.

In order to tackle this problem, this paper proposes an ontology evolution technique that can enhance manual ontology with additional in-depth knowledge previously discovered in an automatic manner. The ontology evolution technique is based on the concept of ontology integration. The proposed technique has been applied to evolve Machine Ontology which is used to support customer service for industry manufacturers in Singapore. Some experimental results are also presented.

Acknowledment

The authors wish to acknowledge the helpful support from Nguyen Tat Thanh College, Hochiminh City, Vietnam for the work presented in this paper.

1 Introduction

Currently, information available on the Web has been designed for human to understand. Programs can be written to process, analyze and index web pages to help us to process the information. However, due to the lack of machine-readable structure and knowledge representation in web documents, programs are unable to comprehend web page contents precisely, and hence semantic information from web documents cannot be extracted. The Semantic Web is therefore proposed as an extension to the current Web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation (Berners-Lee *et. al.* 2001).

Ontology is used to represent knowledge on the Semantic Web. Generally, ontology is a conceptualization of a domain into a human understandable, but machine-readable format consisting of entities, attributes, relationships and axioms (Guarino and Giaretta 1995). As such, programs can use the knowledge from the Semantic Web for processing information in an intelligent manner.

Recently, the term ontology engineering has been used to imply ontology-related research in computer science (Mizoguchi and Ikeda 1996). The current issues on ontology engineering include ontology generation, ontology mapping (Maedche *et. al.* 2002, Omelayenko 2002), ontology integration (Gangemi *et. al.* 2002, Noy and Musen 2002, Stumme Maedche 2001), ontology versioning (Klein *et. al.* 2002) and ontology evolution. Particularly, when ontologies currently are more and more deployed in applications, the issue on ontology evolution techniques is attracting much attentions since it can help significantly ontology engineers to capture and reflect knowledge more precisely in certain domains (Stojanovic and Motik 2002, Noy and Klein 2003).

Basically, ontology evolution, first termed by Klein *et al.* (Klein *et. al.*, 2002), is a process which adapts the contents of a pre-defined ontology used in practical applications based on the environment in which the applications are deployed. In recent research, there are many techniques proposed for ontology evolution. Inspired from natural evolutions, biological concepts are suggested to be adopted and adapted for ontology evolution (O'Brien 2006). Another remarkable approach is using conceptual graphs combined with ontology editor tool such as Protégé (Blundell and Pettifer 2004). However, currently there are not much good practical results reported from research on ontology evolution, due to the following reasons:

- Most of proposed techniques on ontology evolution heavily rely on manual methods. Thus, ontology evolution become a tedious and complex task, especially when representing large-scaled and in-depth domain knowledge.

- Generally, ontology is not formally defined before the ontology evolution process is performed. Thus, it is difficult to conduct a mathematical model which can help to automate ontology evolution.

In previous research (Quan *et. al.* 2006), we have proposed an approach for automatic ontology generation, which is based on the Formal Concept Analysis (FCA) theory (Ganter and Wille 1999). The ontology constructed by our approach is formally defined as a mathematical model employing FCA-based concepts for knowledge representation. The ontology can also present uncertainty information if necessary. Therefore, it provides a solid theoretical foundation for further study of ontology evolution.

In this paper, we propose an approach for ontology evolution. The ontology which is supposed to be evolved could be preliminarily generated either manually or automatically using our technique as well as other automatic ontology generation techniques. In order to evolve the ontology, we first analyze the in-depth knowledge of the domain of the application deploying the ontology. Then, we integrate the analyzed domain knowledge with the preliminary ontology. In addition, in this paper we also particularly discuss applying our proposed ontology evolution technique on a so-called Machine Ontology to suppor customer service in industry manufacturers.

The rest of the paper is organized as follows. In Section 2 we discuss Machine Ontology for customer service. Section 3 presents our proposed ontology evolution technique. Section 4 evaluates the performance based on some experiments conducted. Section 5 remarks some conclusion. The paper ends with references presented in Section 6.

2 Machine Ontology for Customer Service

Customer service support (Hui and Jha 2000), which renders helps while necessary for customers, has become an integral part of many multinational manufacturing companies which produce insertion and surface mount machines in the electronics industry. A customer service department is usually setup to provide prompt responses to service requests from customers, which may be located worldwide. The customer services of faulty machines are typically supported by a help-desk of the customer service department via telephone calls. This method relies heavily on the service engineers' past experience of similar reported problems, which is stored in a customer service database. Thus, a pool of expert service engineers is required in order to support such online customer services. With the advancement of the Internet, many manufacturing companies have moved their customer service support online over the Web with web-based help-desk applications (Hui *et. al.* 2001). Suggested remedial actions to the reported faults can be generated automatically from a customer service database or through the interaction between a user and service engineer in which the Web is used as a medium for communication.

Recently, the rapid development of the Semantic Web and Semantic Web Services (Austin *et. al.* 2002) has prompted us to consider supporting online machine services over the Semantic Web, which helps users to deal with problems occuring on the purchased machines. In a typical insertion and surface mount machine, there exist many machine models. Thus, a customer usually possesses different types of models and machines for manufacturing purposes. In such cases, when a machine fault problem occurs, the customer might need to obtain service support from different manufacturers. In conventional web-based service support systems, the customer is required to access different web sites, and retrieve different service support suggestions in order to rectify the machine fault problem. This is troublesome and cumbersome, and there is also the integration problem to be tackled.

As previously discussed, ontology is adopted as a standard for knowledge representation in the Semantic Web. Programs can use the knowledge from the Semantic Web for processing information in a semantic manner. As such, the Semantic Web enables machine service knowledge from different machines or models produced by different manufacturers to be shared and integrated. Moreover, the generated machine ontology can also be used to provide an interpretation on the common faults that have occurred for a certain machine model from a concept hierarchy. Thus, supporting machine services utilizying Semantic Web technologies will likely improve customer satisfaction in terms of reducing machine down time as well as increasing productivity.

This vision cannot become reality without a specific ontology known as *Machine Ontology* capturing machine knowledge in a computer-understandable manner. Machine Ontology can be generated either manually by experts or automatically by computer programs. Figure 1 presents an example Machine Ontology that is generated manually by experts. In this ontology, the experts have categorized the machine faults into a concept hierarchy.

Basically, this kind of ontology, so-called *Preliminary Machine Ontology*, is highly precise when reflecting domain knowledge, however since manual generation of ontology is generally tedious, the preliminary ontology is not always of sufficient details when deployed in practical applications.

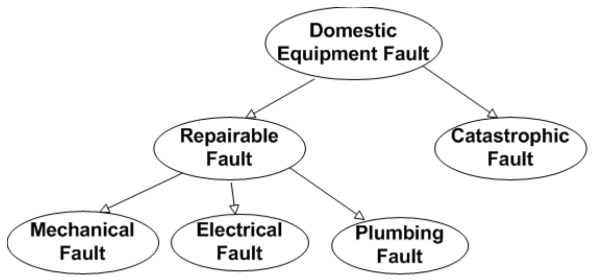


Figure 1: An example Preliminary Machine Ontology

Meanwhile, the initial knowledge stored in the ontology can be further processed using data mining to discover additional hidden information in the data. The discovered data can also be converted into ontological formalism to support more efficient retrieval tasks. Figure 2 illustratively depicts this kind of ontology, known as *In-depth Machine Ontology*. In this ontology, clustering technique is adopted to perform clustering on the dataset of machine faults and then a hierarchy is constructed based on the clusters generated.

The in-depth ontology should reflect specific knowledge of the domain. However, since the in-depth knowledge is quite extensive and profound, it lacks of a clear and precise structure that enables users with basic domain knowledge to easily follow, understand and deploy the knowledge.

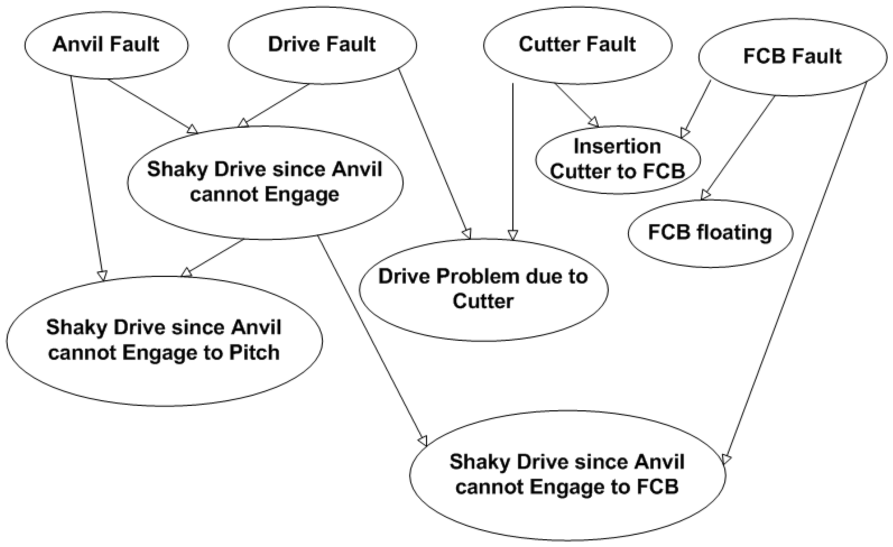

Anvil Fault
Drive Fault
Cutter Fault
FCB Fault
Shaky Drive since Anvil cannot Engage
Insertion Cutter to FCB
FCB floating
Drive Problem due to Cutter
Shaky Drive since Anvil cannot Engage to Pitch
Shaky Drive since Anvil cannot Engage to FCB

Figure 2: An example In-depth Machine Ontology

In this paper, we propose to use the in-depth ontology to make the preliminary ontology evolved accordingly into a final evolved ontology. The evolved ontology could be considered as an enhanced version of the preliminary ontology as it still preserves precisely the concept hierarchy inherited from the preliminary ontology but enriched by additional specific knowledge derived from the in-depth ontology. The evolved ontology therefore reflects the domain knowledge in a far deeper level but still maintaining the structural accuracy predefined by experts.

3 Ontology Evolution

In order to process ontology evolution, we perform ontology integration from the preliminary ontology with the in-depth ontology. In our proposed approach, ontology integration is considered as a process that integrates concepts in a preliminary ontology with those of a in-depth ontology in order to generate a new ontology. Thus, the preliminary ontology is extended with new concepts and relations derived from the in-depth ontology. Ontology integration consists of two sub-steps: (1) Hierarchical Concept Integration and (2) Consistency Checking and Refinemen

3.1 Hierarchical Concept Integration

Basically, an ontology includes classes, or *concepts*, and relations between these concepts; and the instances of the classes. Formally, an ontology can be defined as follows.

Definition 1 (Ontology). An ontology O consists of 4 elements (C, A^C, R, X), where C represents a set of concepts; A^C represents a collection of attributes sets, one for each concept; $R = (R_T, R_N)$ represents a set of relationships, which consists of 2 elements: R_N is a set of non-taxonomy relationships and R_T is a set of taxonomy relationships. Each concept c_i in C represents a set of objects, or instances, of the same kind. Each object o_{ij} of a concept c_i can be described by a set of attributes values denoted by $A^C(c_i)$. Each relationship $r_i(c_p,c_q)$ in R represents a binary association between concepts c_p and c_q, and the instances of such a relationship are pairs of (c_p,c_q) concept objects.

To integrate a Preliminary Ontology O_P and an In-depth Ontology O_I , each concept c in O_I will be mapped as concept c' in O_P, in which c' can be either an old concept already existing in O_P or a new concept of O_p. In case c' is a new concept in O_P, corresponding hierarchical relations will be accordingly constructed to make c' appropriate sub-concepts or super-concept of existing concepts in $O_{P.}$

Thus, in order to integrate concepts in two ontologies reasonably, it is necessary to develop a method to measure the *semantic similarities* of ontological concepts. In literature, there are two major trends of measuring semantic similarities for ontologies. The first approach is based on the structure of ontological concepts (Li *et. al.* 2003, Rada *et. al.* 1989), the second takes into account the internal information of the concepts for similarity measurement (Jiang 1997, Lin 1998, Resnik 1995).

In the context of this paper, we are about to integrate a preliminary ontology with an in-depth one. Thus, the conceptual structure of the two ontologies are quite distant and therefore not much informative for calculating concepts similarities. Hence, we will analyze and make use of the internal information (or *attributes*) of ontological concepts to measure their similarity. First, we will represent attributes of a certain ontological concept as a *fuzzy set*, known as *fuzzy representation* of the concept. (Note that to represent the concept precisely, fuzzy information is needed to reflect some uncertainty information that may occur in some concept attributes). Then we make use fuzzy logic to perform similarity measurement in the generated fuzzy sets. In addition, since a concept can be mapped not only as the same concept but also as a subconcept or superconcept of another concept, we also use fuzzy logic to evaluate the *subsethood* between the concepts.

The above discussion is formally represented as follows.

Definition 2 (Fuzzy Representation of Ontological Concepts). Given a fuzzy ontology (C,A^C,R,X), each concept $c \in C$ can be represented as a fuzzy set, or *fuzzy representation*, *FR(C)* as

$$FR(C)=\{A_1(\mu_1),A_2(\mu_2),...,A_m(\mu_m)\}$$

where $\{A_1, A_2,\ldots, A_m\}$ is the set of attributes in A^C and μ_i is the fuzzy membership A_i, calculated as follows:

$$\mu_i=\frac{n_i}{n}$$

where n_i is number of objects of C that have certain value on A_i and n is the number of the objects of C. It is easy to observe that if C have no uncertainty information, μ_i is 1 for any i.

Definition 3 (Ontological Concept Similarity). The similarity between two ontological concepts c_i and c_j is evaluated as

$$E(c_i,c_j)=\left|\frac{FR(c_i)\cap FR(c_j)}{FR(c_i)\cup FR(c_j)}\right|$$

Definition 4 (Ontological Concept Subsethood). The subsethood between two ontological concepts c_i and c_j is evaluated as

$$subsethood(c_i,c_j)=\left|\frac{FR(c_i)\cap FR(c_j)}{FR(c_j)}\right|$$

Thus, the concept similarity and concept subsethood defined above can significantly help to determine whether or not a concept in a In-depth Ontology should be mapped as a subconcept of or unified with an existing concept in a Preliminary Ontology. It is explained in more details in the following example.

Let's consider a Preliminary Ontology P_I as presented in Figure 3. P_I has three concepts, namely *Mechanical Fault*, *Anvil Fault* and *Drive Fault*. Basically, the concept *Mechanical Fault* describes all of possible mechanical problems on a certain machine model; whereas the concepts *Anvil Fault* and *Drive Fault* respectively concern faults specifically occurring in the anvil and drive components of the machine.

This preliminary ontology is supposedly constructed manually by an ontology engineer. When defining these ontological concepts, the ontology engeneer may as well define some concept attributes, which are some keywords related to the concepts extracted from the user manual of the model. Note that we assume that ontologies involved in our technique conform with concept definition in FCA theory. In this ontology, the *Mechanical Fault* concept is the most general concept. Thus it has theoretically no attributes, as stipulated by FCA theory. Meanwhile, the concepts *Anvil Fault* and *Drive Fault* are defined respectively by the set of attributes {*shaped, hammering, chiseling, forging, steel, plate*} and {*operate, control, force, repel, strike, throwing*}. For simplification purpose, we suppose that the membership values of all attributes are 1 (meaning all of them are not uncertain attributes). For convenience, these concepts are hereinafter referred to as *A*, *B* and *C* as noted in Figure 3.

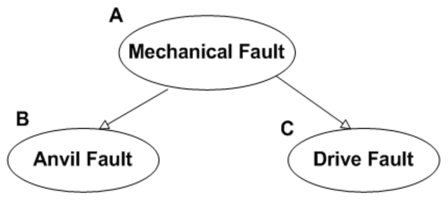


Figure 3: The Preliminary Ontology P_I

Suppose that we intend to integrate P_I with an ontology named In-depth Ontology E_I as depicted in Figure 4. E_I is constructed from real record of fault occuring in the model, collected from users. Basically, in E_I, the fault records are organized as a hierarchy of concepts, each of which is represented by a corresponding set of attributes as shown in Figure 4. The attributes here are once again keywords extracted from the reports of the corresponding fault cases. These concepts are also hereinafter referred to as *D*, *E*, *F*, *G* and *H*.

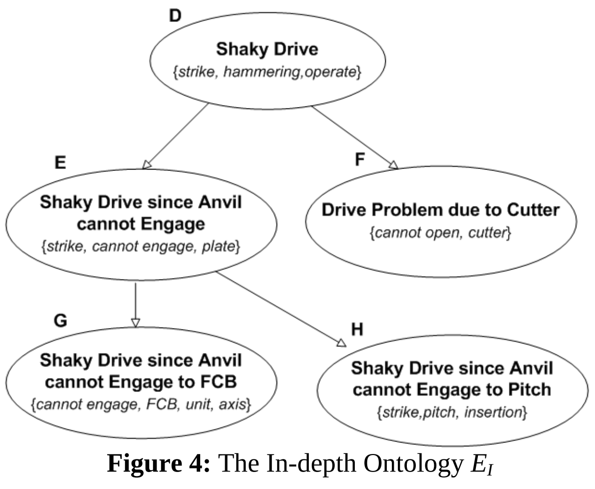

D
Shaky Drive
{strike, hammering,operate}
E
Shaky Drive since Anvil cannot Engage
{strike, cannot engage, plate}
F
Drive Problem due to Cutter
{cannot open, cutter}
G
Shaky Drive since Anvil cannot Engage to FCB
{cannot engage, FCB, unit, axis}
H
Shaky Drive since Anvil cannot Engage to Pitch
{strike,pitch, insertion}

Figure 4: The In-depth Ontology E_I

The ontology integration process, subsequently, is divided into two sub-steps, namely Subsethood-based Integration and Similartity-based Integration.

3.1.1 Subsethood-based Integration

In Subsethood-based Integration process, each concept in E_I will first be mapped as either a subconcept or a superconcept of appropriate concepts in P_I. The mapping is determined by evaluating the subsethood between concepts. For example, consider concepts *A* in P_I and *D* in E_I. We have *subsethood*(*C*,*D*) = 0.67. Since this subsethood value is considerably high, *D* then is mapped as a subconcept of *C*. In other hand, *subsethood*(*C*,*G*) = 0.25 and *subsethood*(*G*,*C*) = 0.125. Since *subsethood*(*G*,*C*) is too low, we do not consider mapping *C* as a subconcept of *G*. Instead, *G* can still be considered as a subconcept of *C*, due to the acceptably high value of the corresponding subsethood.

Thus, after fully mapped, the concepts on E_I will be initially integrated into P_I based on the corresponding subsethoods evaluated. As a result, an initial integration ontology is generated as illustrated in Figure 5. Note that in the initial ontology, *D* is mapped as concurrenly a subconcept as well as a superconcept of *C*. It is because both values of *subsethood*(*C*,*D*) and *subsethood*(*C*,*D*) are quite high (0.67 and 0.33 respectively). The ontology will be refined to solve this problem in the later steps.

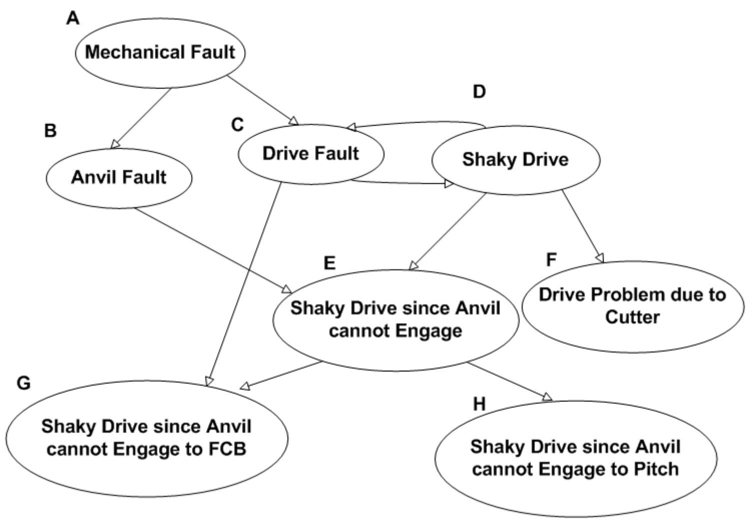

A
Mechanical Fault
B
Anvil Fault
C
Drive Fault
D
Shaky Drive
E
Shaky Drive since Anvil cannot Engage
F
Drive Problem due to Cutter
G
Shaky Drive since Anvil cannot Engage to FCB
H
Shaky Drive since Anvil cannot Engage to Pitch

Figure 5: Initial integration ontology based on subsethood.

3.1.2 Similarity-based Integration

The next step to be performed on the initial ontology is to evaluate the similarities between the concepts. Based on the similarities evaluated, we unify the concepts on the initial ontology if their similarities are considerably high. For example, on the initial ontology given in Figure 5, we have *E*(*B*,*E*) = 0.33. This similarity is considerably high, urging us to consider unifying these two concepts together as a new concept *BE*. As a result, the final integration ontology is generated as shown in Figure 6.

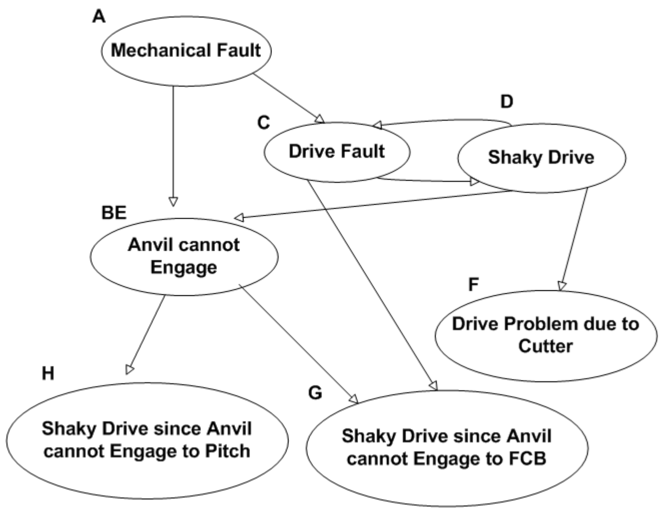

A
Mechanical Fault
C
Drive Fault
D
Shaky Drive
BE
Anvil cannot Engage
F
Drive Problem due to Cutter
H
Shaky Drive since Anvil cannot Engage to Pitch
G
Shaky Drive since Anvil cannot Engage to FCB

Figure 6: Integration ontology of P_I and E_I

3.1.3 Ontology Integration Algorithm

So far, we have introduced the key ideas for integrating ontologies, based on the definitions of ontological concept subsethoods and similarities. The above discussion on ontology integration can be formally presented as Algorithm 1 given in Figure 7.

```
Algorithm 1: Ontology Integration
Input: Preliminary Ontology $O_P$ and In-depth Ontology $O_I$
Output: Evolved ontology $O_E$
Process:
    1:  $O_E \leftarrow O_P$
    2:  for each concept $c$ concept of $O_i$ do
    3:  map c as a new concept $c_N$ of $O_E$
    4:  for each concept $c_E$ concept of $O_E$ do
                  if subsethood($c_N$,$c_E$) > $T_C$ then
                       make $C_E$ subconcept of $C_N$
                  endif
                  if subsethood($c_E$,$c_N$) > $T_C$ then
                       make $C_E$ superconcept of $C_N$
                  endif
    5:  endfor
    6:  find concept $c_{cand}$ of $O_E$ that is most similar to $c_N$
    7:  if E($c_{cand}$,$C_N$)> $T_C$ then
                  make every subconcept of $C_N$ subconcept of $c_{and}$
                  make every superconcept of $C_N$ superconcept of $c_{and}$
                  remove $c_N$ from $O_E$
    8:  endif
    9:  endfor
```

Figure 7. Ontology integration algorithm

3.2 Consistency Checking and Refinement

As previously discussed, the evolved ontology is generated mathematically without human control. Thus, there may be some inconsistency occuring in terms of concept relations. It is because there are some relations are accepted by a our mathematical model, but they do not make sense when concerned in the practical situation. For example, in the integration ontology given in Figure 5, concept *D* is at the same time subconcept and superconcept of concept *C*. Obviously, it makes the ontology unlogical and not natural. Such inconsistency should be refined.

In this research, when checking for inconsistency in the evolved ontology, we only focus on the inconsistency on hierarchical relations of concepts, defined as follows.

Definition 5 (Hierarchical Relation Inconsistency). Given an ontology *O*, a *hierarchical relation inconsistency* will occur in two concepts C_1 and C_2 of *O* if C_1 is concurrently a superclass and a subclass of C_2.

When a hierarchical relation inconsistency occurs in two concepts C_1 and C_2, we will refine the ontology using the information on subsethood of the concepts. That is, if *subsethood*(C_1,C_2) > *subsethood*(C_2,C_1), the taxonomy relation which specifies that C_1 is superclass of C_2 will be removed from the ontology (meaning that C_1 now is only a subclass and no longer superclass of C_2) and vice verca. For example, in the integration ontology mentioned above, we have *subsethood*(*C*,*D*) > *subsethood*(*D*,*C*). Therefore, we refine the ontology to make *D* no longer superconcep of *C*. As a result, we have the refined ontology as presented in Figure 8.

After fully refined, the ontology evolution process is considered complete. As shown in Figure 8, the Preliminary Ontology is now evolved with new concepts raised from In-depth Ontology. Moreover, the initial relations between original concepts have also been refined to adapt with new knowledge obtained.

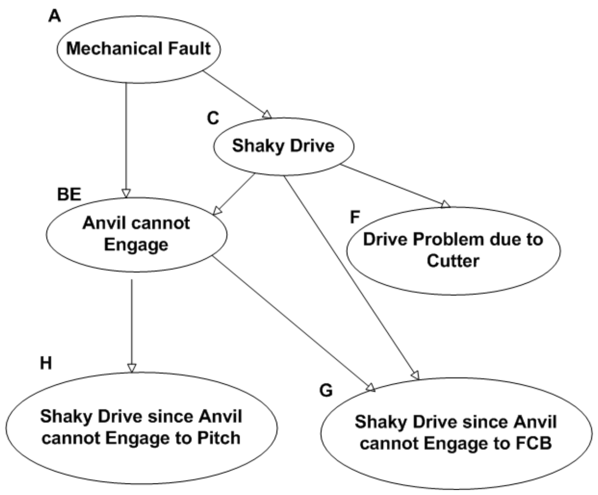

A
Mechanical Fault
C
Shaky Drive
BE
Anvil cannot Engage
F
Drive Problem due to Cutter
H
Shaky Drive since Anvil cannot Engage to Pitch
G
Shaky Drive since Anvil cannot Engage to FCB

Figure 8. Final evolved ontology

4 Performance Evaluation

In order to evaluate the peformance of the proposed technique, we have used a real *Customer Service Database* for ontology generation and evolution. The

Customer Service Database records fault occuring on products of a multinational corporation in Singapore that manufactures insertion and surface mount machines in the electronics industry. In the database, fault records (or reports) are currently defined and stored in the customer service database to keep track of all reported machine problems and remedial actions. Each service record consists of customer account information and service details.

Customer service details contain two types of information: *fault-condition* and *checkpoint information*. Fault-condition contains the service engineer's description of the machine fault. Checkpoint information indicates the suggested actions or services to be carried out to repair the machine into normal condition based on the occurred fault-condition given by the customer. Checkpoint information contains checkpoint group name, and checkpoint description with priority and an optional help file. The checkpoint group name is used to specify a list of checkpoints defined under the group. Each checkpoint is associated with a priority, which determines the sequence in which it can be exercised, and a help file that gives visual details on how to carry out the checkpoint. An example of fault-condition and its checkpoint information for a service record is given in Figure 9.

There are over 70,000 service records in the customer service database. Since each of the fault-conditions has several checkpoints, there are over 50,000 checkpoints. In addition, information on over 4,000 employees, 500 customers, 300 different machine models and 10,000 sales transactions are also stored.

Fault-condition	3008 PCB CARRY MISS ERROR. PCB WAS NOT TRANSFERRED BY THE CARRIER DURING LOADING BUT STAYED AT THE DETECTION POSITION OF PCB DETECTION SENSOR 2.	
Checkpoint group: AVF_CHK007		
Priority	Checkpoint description	Help file
1	CONFIRM WHETHER THE CARRY GUIDE PINS ARE IN LINE WITH PCB.	AVF_CHK007-1.GIF
2	CONFIRM WHETHER THE PCB IS IN CORRECT DIRECTION.	AVF_CHK007-2.GIF
3	CONFIRM THE POSITION OF THE GUIDE LOWER LIMIT SENSOR. (I/O 0165)	AVF_CHK007-3.GIF
4	CONFIRM THE TIMING FOR PCB 2 DETECT SENSOR.	AVF_CHK007-4.GIF
5	CONFIRM THE TIMING FOR THE CARRIER START TIMING.	AVF_CHK007-5.GIF

Figure 9: Customer Service Database

Based on the relational schema of the Customer Service Database, we define the Preliminary Machine Ontology, as illustratively depicted in Figure 1. Subsequently, clustering techniques that can be used to cluster machine fault conditions into groups based on their similarities are traditionally used for this purpose.

In this research, we have applied one of the most effective clustering techniques known as Kohonen Self-Organizing Map (KSOM) Neural Networkfor knowledge discovery. From the clusters generated, we automatically generated the In-depth Machine Ontology, as seen in Figure 2. The ontology generation technique is based on Formal Concept Analysis (FCA) technique.

From the Preliminary Machine Ontology and In-depth Machine Ontology, we generate the Evolved Machine Ontology, as illustrated in Figure 10. The purpose of the evolved ontology is to support customer services of past common fault information and current fault diagnosis. Therefore, an experiment has been conducted to evaluate the system performance of information retrieval.

There are two testing datasets conducted for retrieval accuracy evaluation. The first dataset, denoted as *DS1*, is a set of 15,850 fault-conditions, used as input strings for retrieval. As the fault descriptions in the fisrt set were manually created by experts, the second set, denoted as *DS2,* consists of 50 fault descriptions formed from non-expert users. It is used to test the performance when the input is less technically precise.

KSOM is an unsupervised learning technique, therefore it can automatically discover hidden knowledge in data without using prior knowledge given by human experts. For experimental comparison purpose, a supervised neural network, Learning Vector Quantization Version 3 (or LVQ3), is also adopted in the experiment. In addition, *k*-nearest neighbor (kNN) technique has been typically used for retrieval purpose in traditional customer service system. Therefore, apart of two mentioned neural networks, we also made use of kNN for retrieval evaluation. There are two popular variations of kNN techniques were adopted. The first variation, denoted as kNN1, bases on vector's normalized Euclidean distance to find the fault-conditions closest to the input string. The second, denoted as kNN2, makes use of fuzzy-trigram technique to do so. Then, we evaluate the performance of adopted techniques using two criteria: speed and accuracy.

4.1 Retrieval Speed Evaluation

We evaluate the speed performance based on average on-line retrieval. For KSOM and LVQ3 neural networks, training time was also measured and compared. As a unsupervised learning technique, KSOM can automatically make use the data in the Customer Service Database for training. However, LVQ3 needs human interpretation to determine target fault-condition corresponding to each given training input.

The dataset *DS1* was used for speed evaluation. The statistics information of training and online-average retrieval time of four mentioned techniques is given in Table 1. It states that even though kNN techniques do not need to be trained before performing retrieval, their performance was outperformed by neural networks techniques in terms of speed. As compared to LVQ3, KSOM requires longer time for training, but it performs more efficiently in terms of on-line retrieval speed. Since the training process can be carried out in off-line mode, neural networks, particularly KSOM, are more suitable for an on-line retrieval system than the typical kNN technique.

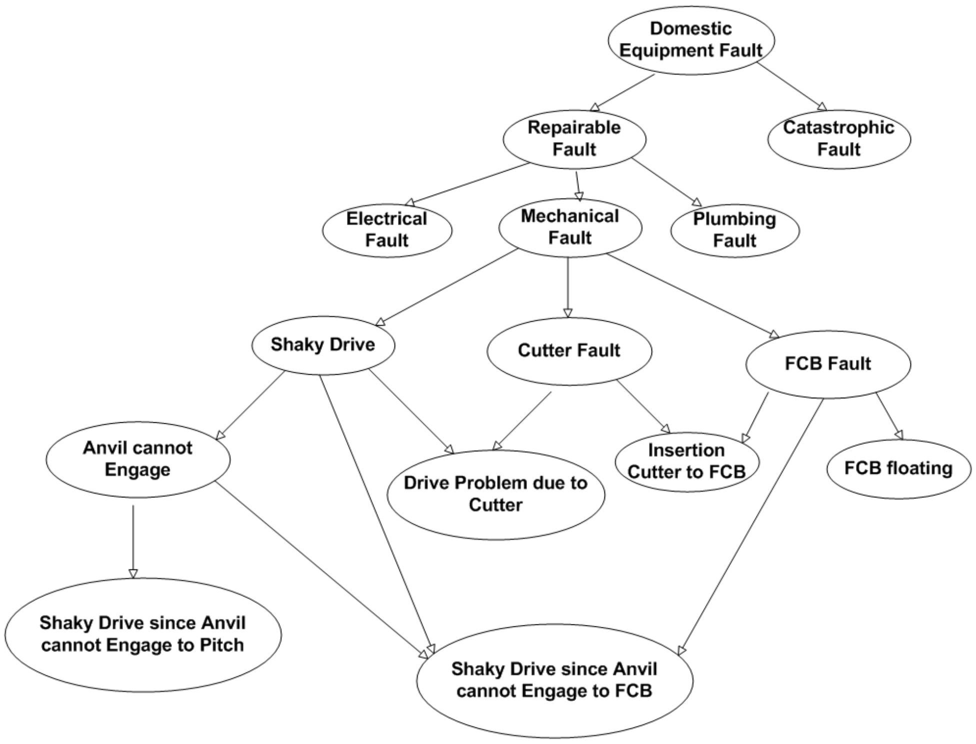


Figure 10: Final evolved Machine Ontology

Description	Time			
	LVQ3	KSOM	kNN1	kNN2
Training	96 min 44 sec	264 min 35 sec	N/A	N/A
Average on-line retrieval	1.9 sec	0.8 sec	15.3 sec	16.7 sec

Table 1: Retrieval Speed Evaluation

Retrieval Technique	Retrieval Accuracy	
	Based on *DS1*	Based on *DS2*
kNN1	81.4%	72%
kNN2	77.6%	76%
LVQ3	93.2%	88%
KSOM	90.3%	86%

Table 2: Retrieval Accuracy Evaluation

4.2 Retrieval Accuracy Evaluation

Retrieval accuracy was evaluated based on the accuracy of ault-conditions retrieved. The way to evaluate retrieval result of KSOM has been discussed earlier. The retrieval accuracy of LVQ3 and kNN techniques could be determined directly through the correctness of fault-conditions retrieved. Table 2 gives the accuracy of retrieval measured on two datasets *DS1* and *DS2*. As can be seen in the table, the accuracies of two neural networks KSOM and LVQ3 were better than those of kNN technique variations. The retrieval accuracy on the dataset *DS2* set was lower than on *DS1*. It is expected due to the fact that technical terms might be not properly used by non-expert users when conducting fault-conditions in *DS2*. In both data sets, LVQ3 achieved better performance than KSOM. It is also expected since a supervised learning technique often obtains better accuracy than an equivalent unsupervised one when applied in a same domain. However, since LVQ3 always needs human interpretation for training, it may encounter problem when dealing with large dataset.

5 Conclusion

This paper presents an ontology evolution technique. This technique allows an initial preliminary ontology to be evolved with new information that is generated automatically from large datasets. Thus, whereas the evolved ontology still maintains the original structural correctness, it is enriched reasonably with new in-depth knowledge which is generally hard for human being to extract manually.

In particular, the proposed ontology evolution technique has been applied to evolve Machine Ontology generated from the real Customer Service Database which records experienced faults occuring in industry manufacturers in Singapore. The evolved Machine Ontology is therefore potential to support customer service for e-commerce in the coming Semantic Web environment.

6 References

Austin D., Barbin A., Ferris C. and Garg S. (2002), Web Services Architecture Requirements. Available at: <http://www.w3c.org/TR/wsa-reqs>, 2002.

Berners-Lee, T., & Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*. Retrieved from http://www.sciam.com/2001/0501issue/0501berners-lee.html, 2001.

Blundell B. and Pettifer S. (2004): Graph Visualization to Aid Ontology Evolution in Protégé, In *Proceedings of 7th International Protégé Conference*, July 2004.

Gangemi A., Fisseha F., Pettman I., Pisanelli D., Taconet M, and Keizer J. (2002): A Formal Ontological Framework for Semantic Interoperability in the Fishery Domain, In *Proceedings of ECAI02 Workshop on Semantic Interoperability*, 2002.

Ganter B. and Wille R. (1999): *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin - Heidelberg, 1999.

Guarino N. and Giaretta. P. (1995) Ontologies and Knowledge Bases: Towards a Terminological Clarification. Toward Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. Amsterdam, Holland: IOS Press, 1995.

Hui S.C., Fong A.C.M. and Jha G. (2001): A Web-based Intelligent Fault Diagnosis System for Customer Service Support, *Engineering Applications of Artificial Intelligence*, **14**: 537-548, 2001.

Hui S.C. and Jha G. (2000): Data Mining for Customer Service Support, *Information & Management*, **38**:1-13, 2000.

Jiang, J.J, and Conrath, D.W (1997). Semantic Similarity Based on Corpus Statistics and Lexical Ontology. *In Proceedings on International Conference on Research in Computational Linguistics*, 19–33,1997.

Klein M., Fensel D., Kiryakov A., and Ognyanov D. (2002): Ontology Versioning and Change Detection on the Web, In *Proceedings of the OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management*, October 2002.

Li, Y., Bandar, Z. A. and McLean D. (2003): An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. *IEEE Transactions on Knowledge and Data Engineering,* **15**(4): 871-882, 2003.

Lin, D (1998). An Information-theoretic Definition of Dimilarity. In *Proceedings of the Int'l Conference on Machine Learning*, 1998.

Maedche A., Motik B., Silva N., and Volz. R. (2002): MAFRA - An Ontology Mapping Framework in the Semantic Web, In *Proceedings of the ECAI Workshop on Know-ledge Transformation, Lyon, France*, 2002.

Mizoguchi R. and Ikeda M. (1996): Towards Ontology Engineering, *technical report*, Osaka University, 1996.

Noy N. and Musen M. (2002): PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions, in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, (Alberta, Canada), 2002.

Noy N. and Klein M. (2003): Ontology Evolution: Not the same as schema evolution, *Knowledge Information Systems*, **5**, 2003.

Omelayenko (2002): RDTF: a Mapping Meta-ontology for Business Integration, In *Proceedings of the Workshop on Knowledge Conference on Artificial Intelligence for the Semantic Web at the 15th European Conference on Artificial Intelligence(KTSW2002),* Lyon, France, 2002.

O'Brien P. (2006): Modeling Intelligent Ontology Evolution Using Biological Evolutionary Processes, *Engineering of Intelligent Systems, IEEE International Conference*, 2006.

Quan T.T., Hui S.C., Fong A.C.M. and Cao H.T. (2006) : Automatic Fuzzy Ontology Generation for Semantic Web, *IEEE Transactions on Knowledge and Data Engineering*, **18**(6): 842- 856, 2006.

Rada, R., Mili, H. Bicknell, E. and Blettner, M (1989). Development and Application of a Metric on Semantic Net. *IEEE Transactions on Systems, Man and Cybernetics*, **19**(1):17-30, 1989.

Resnik, P. (1995): Using Information Content to Evaluate Semantic Similarity in Ontology. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*,448–453,1995.

Stojanovic L. and Motik B. (2002): Ontology Evolution within Ontology Editor, In *Conference on the Evaluation of Ontology-based Tools*, September 2002.

Stumme G. and Maedche A. (2001): FCA-merge: Bottom-up Merging of Ontologies, In *Proceedings of the 17th International Conference on Artificial Intelligence (IJCAI '01),* (USA): 225-230, 2001.

Creating and Querying Linguistically Motivated Ontologies

Rolf Schwitter

Centre for Language Technology
Macquarie University
Sydney NSW 2109, Australia
Email: `schwitt@ics.mq.edu.au`

Abstract

This paper argues that a formal ontology (in our case a description logic knowledge base) should be created in a linguistically motivated way so that it can be queried easily by non-specialists. This can best be achieved by using a strict naming convention that is based on those linguistic expressions that occur in the application domain for which the ontology will be created. We will see that ABox and TBox statements that closely follow this naming convention can be written directly in a controlled natural language and that the same controlled natural language can be used to query the description logic knowledge base. Both ABox and TBox statements written in controlled natural language are translated automatically into the Knowledge Representation System Specification (KRSS) syntax and questions are translated into RacerPro's new query language nRQL and answered over the description logic knowledge base. Using a controlled natural language as a high-level interface language abstracts away from any formal notation and allows for true collaboration between humans and machines.

Keywords: ontology design, controlled natural languages, question answering, human-computer interfaces

1 Introduction

Formal languages are difficult to understand and use by non-specialists – ontologies are no exception in this respect. However, ontologies have a rather special status since their role is to specify a vocabulary with which assertions and queries are exchanged not only between machines but also between humans and machines. In the ideal case, ontologies should be human-readable as well as machine-processable since they are agreements among all participants to use a common vocabulary in a specific domain (Gruber 1993).

Recently, the use of machine-oriented controlled natural languages has been suggested to create ontologies in a human-readable and machine-processable way (Schwitter & Tilbrook 2004). These controlled natural languages look like natural language but they are in fact formal languages "in disguise". They have a formal syntax and semantics and can be unambiguously translated into an existing formal language, for example into a version of description logics or into first-order logic.

There exists an entire stream of research that investigated the usefulness of controlled natural languages for authoring and verbalising description logic-based ontologies. For example, (Schwitter & Tilbrook 2004, 2006) discuss the bidirectional mapping between the controlled natural language *PENG* and various subsets of the web ontology language OWL. This work built the foundation for *Sydney OWL Syntax* – a proposal of a controlled natural language syntax for OWL 1.1 (Cregan et al. 2007). In (Bernardi et al. 2007) a categorial grammar is introduced that translates the controlled natural language *Lite Natural Language* into the description logic DL-Lite, a tractable fragment of OWL 1.1. This fragment is expressive enough to deal with UML diagrams and relational databases. In (Hart et al. 2007) the controlled natural language *Rabbit* is presented which focuses on the knowledge acquisition process and requires that a domain expert and a knowledge engineer work in cooperation to create an OWL ontology. In (Kaljurand 2007) a bidirectional interface to OWL is discussed where a subset of the controlled natural language *Attempto Controlled English* is used for the purpose of authoring and verbalising OWL ontologies. The three controlled natural languages *Sydney OWL Syntax*, *Rabbit*, and *Attempto Controlled English* are compared in (Schwitter et al. 2008) and a number of requirements to an OWL compatible controlled natural language are put forward.

There exists another stream of research that studied the usefulness of controlled natural language and unrestricted natural language as ontology query languages. For example, in (Bernstein et al. 2004) a controlled language-based query interface is introduced that guides the writing process of questions using predictive interface techniques, and in (Bernstein et al. 2005) the authors show that this type of interface performs better than formal query languages. There exist also a number of systems that support the use of unrestricted natural language for answering questions over ontologies (Lopez et al. 2006, Kaufmann et al. 2006, Mithun et al. 2007). However, these unrestricted approaches suffer from similar problems to database systems with natural language interfaces (Copestake & Sparck-Jones 1990, Androutsopoulos et al. 1995, Popescu et al. 2004) because the end-users both over- and undershoot the system's capabilities since they don't know where the system's boundaries are and what kind of queries are supported. It is an ongoing debate which approach is most convenient for end-users (Reichert et al. 2005, Kaufmann et al. 2007), and it is not surprising that the performance of a natural language query interface to a description logic knowledge base depends highly on the quality and choice of the vocabulary in the knowledge base (Kaufmann et al. 2007).

In this paper, we will bring these two research streams closer together and argue that a description logic knowledge base should be created in a **linguistically motivated** way in order to support question answering in an optimal way. We will show that a controlled natural language can be used for making terminological and assertional statements and that the structure of these statements is related in a systematic way to the controlled natural language used for expressing questions.

The rest of this paper is structured as follows: In Section 2, we will give a brief introduction to description logics and introduce the main constructors that are usually used to build a description logic knowledge base. In Section 3, we will study the relationship between the terms used in existing description logic ontologies and natural language expressions and make some recommendations about how to use linguistic expressions in an optimal way in ontologies. In Section 4, we will show that a description logic knowledge base can be specified in controlled natural language and introduce a notation that uses reification of relations in order to deal with n-ary relations in description logics. In Section 5, we look at the kind of questions that are supported by a state of the art description logic reasoner and suggest controlled natural language renderings for these questions. In Section 6, we will discuss the architecture of a controlled natural language processor that relates the structures of statements and questions written in controlled natural language in a systematic way to each other and translates these sentences into the input format of a description logic reasoner; we will also show that the same language processor can be used to generate answers to questions in controlled natural language. Finally, in Section 7, we will summarise the benefits of our approach and point to a number of further research challenges.

2 Description Logics (DLs)

DLs are a family of knowledge representation languages that can be understood as decidable fragments of first-order logic (Baader et al. 2003). DLs – such as the web ontology language OWL DL (Horrocks et al. 2003) – play an important role in the Semantic Web architecture because they are decidable and allow for a clear separation between the terminological information and the data.

A DL knowledge base usually consists of two components: a terminological component (= TBox) and an assertional component (= ABox). The TBox asserts general facts about concepts (= unary predicates) and roles (= binary relations) through declarations while the ABox asserts specific facts about individuals in an application domain. For example, a TBox might contain concept definitions like (1) and general inclusion axioms like (2):

```
1. (equivalent academic_staff (or
    professor senior_lecturer lecturer))

2. (implies student (or graduate_student
    undergraduate_student))
```

and the ABox might contain concept assertions like (3) and role assertions like (4):

```
3. (instance david_miller
    undergraduate_student)

4. (related david_miller comp101 attend)
```

Note that we use the Knowledge Representation System Specification (KRSS) syntax (Patel-Schneider & Swartout 1993) here in order to express these and all subsequent axioms in a compact way.

The expressivity of a DL depends on the set of constructors that can be used for defining concept terms from concept names and role names. Some common constructors include logical constructors: *intersection*, *union*, and *complement*; and *quantified role restriction* like (5), and *number restriction* like (6):

```
5. (instance kylie_jones (some attend
    unit))

6. (instance comp101 (exactly 22 student))
```

Other constructors are used to characterise roles and for enhanced reasoning with roles, for example: *inverse*, *transitivity*, and *functionality*. For instance, (7) defines a role with a domain and a range restriction and provides a name for the inverse of the defined role:

```
7. (define-primitive-role teach
    :inverse is_taught_by
    :domain academic_staff
    :range unit)
```

In addition to these features, some DLs provide extensions for algebraic reasoning over concrete domains, for example:

```
8. (instance david_miller (= age 21))
```

Here the age of an individual is specified with the help of a concrete domain predicate (`=`) and a concrete domain attribute (`age`) that is of type `cardinal`.

3 Linguistic Structures in Ontologies

A DL ontology is essentially a logical theory that specifies a conceptualisation of a specific part of the world – in our case of an university. Since the aim of an ontology is to make domain assumptions explicit and to establish a common understanding of the structure of information in a particular domain, it is important that all participants are able to use the same vocabulary in order to make assertions and ask queries in a way that is consistent with the logical theory. It should be possible for a non-specialist to relate the terms used in the ontology to the entities in the application domain. That means the terminology should be easy to use and understand by humans so that humans and machines can cooperate in the best possible way.

It seems intuitive to choose a naming convention for creating an ontology that is close to the linguistic expressions that occur in a particular domain. However, this is not always the case and a wide diversity and fragmentation of naming schemes can be found in real-world ontologies (Schober 2007). This is because knowledge engineers can use in principle any character sequence for naming entities and often make cost/accuracy trade-offs when creating an ontology. For a machine it does not make any difference whether a concept or a role is labeled in a way that can be understood by a human or not. Here is an extreme example:

```
9. (implies C1 (and C2 (some R (or C3
    C4))))
   (or C3 C4)
   (implies C5 (and C1 (some R C3)))
   (implies C6 (and C1 (some R C4)))
```

The labels that occur in this excerpt make it difficult for a human reader to understand what the single concept and role names stand for, how they relate to each other, and how they refer to the entities in the application domain for which the ontology is designed. It is much easier to understand what is going on if the terms in the ontology communicate the intended meaning using natural language-like expressions, for example:

```
10. (implies person (and human (some
     has_gender (or female male))))
    (disjoint female male)
    (implies woman (and person (some
     has_gender female)))
    (implies man (and person (some
     has_gender male)))
```

Existing ontologies sometimes combine natural language expressions with non-linguistic artifacts. But there is a clear tendency to use linguistic patterns in ontologies. (Mellish & Sun 2005) collected 882 ontology files encoded in OWL and analysed the linguistic structure of concept and role names. They found that 72% of concept names ended with an English noun, 30% of concept names consisted entirely of noun sequences (various forms of compound nouns), and 14% contained no recognised word. The structure of role names showed a broader variety of patterns but 65% of these patterns started with a verb.

Given these results, we suggest using those patterns that occur most frequently in existing ontologies in order to establish a linguistically motivated naming convention. In particular, we suggest using

- nouns and compound nouns as concept names (e.g. `student` and `undergraduate_student`);
- transitive verbs and auxiliary verb-noun constructions as role names (e.g. `take`, `consist_of`, `is_student_of`, `has_student`);
- normalised forms of proper nouns as identifiers for individuals (e.g. `david_miller`, `comp101`).

Following (Schober 2007), we additionally recommend to use an underscore (_) to delimit words in compound terms since this separator is closer to natural language than CamelCase; to replace homonyms by an alternative word form since they create confusion; to resolve abbreviations and acronyms in names and include them as synonyms in the ontology.

4 Controlled Natural Languages (CNL)

In the last section, we have argued that using a linguistically motivated naming convention can improve the readability and usability of an ontology for non-specialists. But also knowledge engineers who have to maintain and extend an ontology for new business needs can benefit from naming conventions since well-chosen naming conventions can enhance clarity, avoid the introduction of faults, and make it easier for subsequent generations of analysts to understand what the ontology is designed for. A linguistically motivated naming convention can directly be applied when creating an ontology with an ontology editor although current ontology authoring tools do not actively enforce naming conventions.

Adopting a common naming convention is a good strategy to improve the quality of an ontology but we can even go a step further towards natural language and express ABox and TBox statements of an ontology completely in a controlled natural language (CNL) and then translate these statements directly into the input language of a DL reasoner. That this can be done has been shown in previous work (Schwitter et al. 2008). Therefore, we will give here only a brief overview in order to discuss the underlying design principles and illustrate how such CNL statements look like before we focus on the structure of CNL questions and the relationship between CNL assertions and questions.

4.1 Terminological Statements in CNL

TBox statements express the intensional knowledge about a domain in form of a terminology. This terminological information is static, and it is more likely that a knowledge engineer will write and modify such statements than a domain specialist. Nevertheless, domain specialists need to be able to read, understand and validate this information with respect to an application domain. A CNL can help in this respect since it provides a high-level interface to a formal language that is potentially difficult to understand by an end-user. For example, a concept definition like (1) – see Section 2 – can be expressed as (11) in CNL, and a general inclusion axiom like (2) as (12):

```
11. Every academic staff is defined as
    a professor or a senior lecturer or
    a lecturer.

12. Every student is a graduate student
    or is an undergraduate student.
```

The TBox statement (11) makes it explicit that this statement is a definition that consists of a set of necessary and sufficient conditions. In contrast to (11), the TBox statement (12) does not speak about a definition since it only states necessary conditions for a primitive concept.

The definition of primitive roles in CNL requires the use of meta-information in order to speak about the features of a role. The use of variables makes it possible to speak about various features of a role in a very compact way in CNL. Note that variables are not a specific characteristic of a formal language; variables are also used in natural language texts as a look into any undergraduate maths textbook proves. Here is the CNL rendering of the primitive role definition (7) that uses two variables to signal the domain and range restrictions of the role:

```
13. X teaches Y is defined as Y is taught
    by X and X is an academic staff and Y
    is a unit.
```

Note that there is no need to use additional constructors such as *inverse, domain*, and *range* as in (7) since the CNL language processor can figure out the type of these restrictions from the linguistic structure.

4.2 Assertional Statements in CNL

In contrast to TBox statements, ABox statements are much more dynamic in nature and more likely to be used by non-specialists. ABox statements rely on the vocabulary defined in the TBox. In the simplest case, ABox statements can be expressed via an auxiliary or a transitive verb in CNL, for example, the concept assertion (3) can be expressed as (14) and the role assertion (4) as (15):

```
14. David Miller is an undergraduate
    student.

15. David Miller attends COMP101.
```

The following two examples (16) and (17) illustrate how the quantified restriction in (5) and the number restriction in (6) can be expressed in CNL:

```
16. Kylie Jones attends some unit.
```

```
17. COMP101 has exactly 22 students.
```

The subsequent rendering (18) shows how the concrete domain example introduced in (8) can be expressed in CNL:

```
18. David Miller's age is 21.
```

In summary: all these ABox statements are based on the following simple functional structure:

```
19. Subject Verb Object
```

As we will see later, coordination is allowed in object position but not in subject position since this would introduce ambiguity. However, nouns that occur in the subject position can be modified as the example in (18) illustrates.

4.3 From Binary to N-ary Relations

Most DL languages only support binary relations and at first glance it looks like only very simple CNL statements can be captured by this logical framework. However, it is often necessary (and more natural) to describe relations among more than two individuals in one statement. This can be achieved via reification of binary relations (Noy & Rector 2006). For example, the ABox statement:

```
20. David Miller takes COMP101 on Monday
    at 11am in the Lincoln Building.
```

can be represented in DL via a concept assertion consisting of a new individual that stands for a reified relation (= verbal event) and a set of role assertions that link the individual of the reified relation with its participants:

```
21. (instance e1 take)
    (related e1 david_miller has_agent)
    (related e1 comp101 has_theme)
    (related e1 monday has_day)
    (related e1 1100 has_hour)
    (related e1 lincoln_building
     has_location)
```

Here the new individual `e1` stands for the reified relation and is an instance of the `take` concept. The roles (`has_agent`, `has_theme`, `has_day`, `has_hour`, and `has_location`) link this new individual to their corresponding individuals and values. Of course, this approach requires that the `take` concept is available in the TBox, for example as a defined concept:

```
22. (equivalent take (or attend select))
```

and that all the relevant roles are also defined. For example, the declaration of the `has_theme` role looks as follows in our context:

```
23. (define-primitive-role has_theme
     :domain take :range unit)
```

Note that the concept term `take` occurs in this definition as a domain restriction and the concept term `unit` as a range restriction.

5 DL Reasoners

There exist a number of state of the art DL reasoners for querying a DL knowledge base, for example: FaCT^{++} (Tsarkov & Horrocks 2006), Pellet (Sirin et al. 2007), and RacerPro (Haarslev & Möller 2003, Wessel & Möller 2006). These reasoners usually implement a tableau-based decision procedure for general TBox reasoning (subsumption, satisfiability, and classification) and offer support for ABox reasoning (retrieval and conjunctive queries). Pellet and RacerPro both support a subset of SPARQL (Prud'hommeaux & Seaborne 2008) for answering conjunctive ABox queries.

5.1 RacerPro and nRQL

RacerPro is a knowledge representation system that implements the expressive description logic $\mathcal{ALCQHIR}_{\mathcal{R}}+(\mathcal{D}^-)$. This is the basic description logic $\mathcal{ALC}$ augmented with qualifying number restrictions, role hierarchies, inverse roles, and transitive roles. In addition to these basic features, RacerPro also provides facilities for algebraic reasoning including concrete domains and implements most of the functions specified in the KRSS specification (see (Racer Systems 2007) for details).

The new RacerPro query language (nRQL) is a query language for RacerPro's concept language and supports – among other things – strong negation, negation as failure, and numeric constraints. RacerPro translates SPARQL queries into nRQL queries. nRQL is a more expressive query language than SPARQL. In contrast to SPARQL, nRQL uses description logic reasoning and does not work on the syntactic level of triples but on the level of semantic models (Racer Systems 2007).

In the following, we will first show how the syntactic structure of nRQL queries looks like and discuss then the main types of queries that are supported by RacerPro and provide CNL renderings for these queries.

5.2 Queries in nRQL

An nRQL query consists of a query head and a query body. The query head specifies the format of the answer and the query body contains (unary or binary) query atoms that are used to specify retrieval conditions on the bindings of query variables. nRQL queries are either simple or complex. A simple nRQL query consists of a single query atom in the query body. A complex nRQL query consists of two or more query atoms that are combined with the help of query body constructors (e.g. `and`, `union`, `neg`). For example, the simple nRQL query:

```
24. (retrieve (?1) (?1 comp101 take))
```

has a query variable (`?1`) as head and a binary query atom (`?1 comp101 take`) as body. This nRQL query can be expressed as a *wh*-question in CNL:

```
25. Who takes COMP101?
```

Note that the nRQL query in (24) can not be answered over a DL knowledge base that contains reified relations as introduced in (21). In order to answer the CNL question (25) over a DL knowledge base that contains reified relations, the verbal relation of the question needs to be reified too and this results in a complex (conjunctive) nRQL query of the form:

```
26. (retrieve (?2) (and (?1 ?2 has_agent)
     (?1 take) (?1 comp101 has_theme)))
```

The body of this complex query is composed of a query body constructor (`and`) that takes an unary query atom ((`?1 take`)) and two binary query atoms ((`?1 ?2 has_agent`) and (`?1 comp101 has_theme`)) as arguments.

5.3 nRQL Queries in CNL

The query language nRQL distinguishes a number of different types of queries that can be answered over a DL knowledge base. In the following, we will discuss CNL renderings for the most important types of these queries.

5.3.1 Queries about Concepts

In nRQL, queries about concepts can be used to retrieve all instances of a concept from an ABox. This type of queries can be expressed in CNL via a *wh*-question (27) or as an imperative construction (28):

```
27. Who is a student?
28. Find all students.
```

The nRQL query processing engine returns a set of tuples of the form `(((?1 david_miller)) ((?1 eva_barth)))` as answer to these questions. Note that the CNL question (28) makes the expected answer set explicit using the universal quantifier *all* in contrast to (27). A partial answer such as `((?1 eva_barth))` would be an acceptable answer for question (27) but not for question (28).

5.3.2 Queries about Roles

In nRQL, queries about roles can be used to retrieve role fillers from an ABox. Since these queries extract information from binary relations, there are in principle three different queries one might want to ask about the arguments of a binary relation. This corresponds to the following three CNL questions:

```
29. Who takes what?
30. Who takes COMP101?
31. What does David Miller take?
```

In (29) we are asking for information about the subject as well as the object of a relation, in (30) we are only asking for information about the subject, and in (31) only for information about the object. Note that the structure of (31) is different from (29) and (30). The query word *what* has been moved from the object position to the front of the sentence and is used there together with a *do*-operator.

5.3.3 Boolean Queries

In nRQL, simple Boolean queries can be used to check whether or nor at least one individual exists for a specific concept, whether or not a specific individual exists for a particular concept, and whether or not two specific individuals stand in a particular relation. Boolean queries return either true or false.

In CNL, we can express Boolean queries via a *yes/no*-question. The following two questions (32) and (33) are equivalent and ask whether or not at least one individual exists that is a student:

```
32. Is there a student?
33. Does a student exist?
```

However, these two CNL questions are constructed in a different way: question (32) uses the verb *be* and an existential *there*, and question (33) uses a *do*-operator and the intransitive verb *exist*.

The following CNL question checks whether or not a specific individual belongs to a specific concepts:

```
34. Is David Miller a student?
```

and finally the subsequent CNL question (35) checks whether or not two specific individuals stand in a particular relation:

```
35. Does Lisa Brown teach Kylie Jones?
```

As we will see later, the answer to these questions is *yes* or *no* but one can also include the focus of the question in the answer, for example: *Yes, Lisa Brown does.*

5.3.4 Classical Negated Concepts and Roles

The nRQL query language allows for classical negated concepts and roles. For example, RacerPro can prove that somebody is not a student or that a student cannot teach a unit. Note that the negation of roles is only available in the nRQL query language but **not** in the concept language of RacerPro (in order to guarantee decidability). Here are two examples: in question (36) a concept is negated and in question (37) a role:

```
36. Who is not a student?
37. Who does not teach a unit?
```

The first question returns all instances that are not part of a concept and the second question returns all instances that are not a filler of a specific role with an existential restriction.

5.3.5 Implied Role Fillers

In nRQL only explicitly modeled role fillers are retrieved by default. However, a DL knowledge base can have logically implied role fillers whose presence is enforced in the logical models of the knowledge base. Let us assume that we asserted at some point that *Anna Grau is a professor* then Anna must have a PhD but this information might not be explicitly present in the DL knowledge base. In nRQL, it is possible to identify those individuals which have a certain role filler that is not explicitly modeled in the knowledge base using a negation as failure (`neg`) and a `project-to` operator. In CNL, we use the specific keyword *show* to trigger the same functionality:

```
38. Show which professor has a PhD.
```

This imperative construction returns individuals which have an implicit role filler which is not explicitly modeled in the knowledge base. Note that the answer set of (38) is different from that of question (39):

```
39. Find all professors who have a PhD.
```

The answer to (38) returns only the implicit instances while (39) returns all instances.

5.3.6 Concrete Domains

In nRQL, the concrete domain part of an ABox can be queried using concrete domain predicates. In CNL, we can use questions such as:

```
40. Who's age is 18?
41. Who is the oldest student?
```

for this purpose. Note that the relation between the adjective *oldest* and the concrete domain attribute `age` needs to be handled in the linguistic lexicon of the CNL processor.

5.3.7 Synonyms

In nRQL it is possible to check whether two instances are individual synonyms or not. In CNL we can check this using the expression *same as* in a question, for example:

```
42. Is Kylie Johns the same as Kylie
    Johns-Pedersen?
43. Is Kylie Johns not the same as Kylie
    Johns-Pedersen?
```

That means synonyms need not to be modeled in the linguistic lexicon. As we will see in the next section, only orthographic variants of content words are modeled in the linguistic lexicon but not synonyms.

5.3.8 Conjunctive Queries

A conjunctive query is a complex nRQL query that is constructed from unary and binary query atoms with the help of the query body constructor `and`. As soon as we work with a DL knowledge base that relies on reification of relations we will end up with conjunctive queries in the translation. The following CNL questions are complex (conjunctive) questions:

```
44. On what day does Kylie Johns take
    COMP101?

45. Where does Kylie Johns take COMP101?

46. Which student who takes COMP101 does
    not take MATH102?
```

Note that the processing of the two questions (44) and (45) relies on a DL knowledge base that reifies relations as discussed in Section 4.3 while (46) does not. However, all three CNL questions are complex questions and their translation results in three conjunctive nRQL queries.

6 The CNL Processor

The task of the CNL processor is to translate ABox and TBox statements as well as questions written in controlled natural language into the input format of the DL reasoner, and to generate answers in CNL from the output of the reasoner.

The CNL processor uses for this task a bidirectional unification-based grammar and a linguistic lexicon that contains syntactic constraints and information about the mapping between the linguistic expressions and the terms of the DL.

The kernel of the grammar is implemented as a definite clause grammar that can either run independently or be processed by a chart parser (Kay 1980) that stores processed substrings and hypotheses about substrings. Storing processed substrings reduces redundancies during the parsing process, and the maintained hypotheses can be used to predict the next possible processing steps. As we will see later, the chart parser implements a meta-interpreter that generates lookahead information that can be used to support the writing process of the user in a predictive way.

The CNL processor first translates the input sentences into TPTP notation (Sutcliffe & Suttner 1998) and then further into the DL reasoner's target format. TPTP is a widely used notation for representing problems for automated theorem proving in first-order logic. The use of TPTP as an intermediate representation language has the advantage that we can extend the grammar of the CNL processor later and interface the processor easily with other (first-order) reasoning services. TPTP gives us the necessary flexibility to translate statements and questions into a suitable target format.

In our case, the CNL processor translates ABox and TBox statements via TPTP notation into RacerPro's KRSS format and adds the resulting formulas to the DL knowledge base. Questions too are first translated into TPTP notation but then transformed into nRQL syntax, and finally answered over the DL knowledge base using RacerPro's reasoning engine. The TPTP representation of a question is stored by the language processor and used as a template for answering questions. This is possible because the syntactic structure of questions and statements is related in a systematic way in CNL. That means that the same CNL grammar can be used as a processor and as a generator.

6.1 The CNL Lexicon

The lexicon of the CNL processor distinguishes between two main categories of words: content words and function words. Content words (nouns, adjectives, verbs, prepositions, and proper nouns) are closely related to concept names, role names, and names for individuals; while function words (conjunction, disjunction, negation, quantifiers, cardinals, and operators) are closely related to DL constructors – provided that we model the DL knowledge base in a linguistically motivated way. Function words are predefined in the lexicon and build the scaffolding of the CNL while content words can be added by the user to the lexicon.

In order to exclude ungrammatical sentences in CNL, the lexicon contains syntactic information that enforces, for example, number agreement between the subject and the verb of a sentence like (47) and the subject and the auxiliary verb of a question like (48):

```
47. David Miller takes COMP101.

48. When does David Miller take?
```

Below are the lexical entries for the proper nouns *David Miller* and *COMP101*:

```
49. lex(cat:[pn],wf:['David','Miller'],
         sn:[],
         syn:[third,sg],sort:[person],
         fol:X^named(X,john_miller)).

50. lex(cat:[pn],wf:['COMP101'],
         sn:[['COMP',101],['COMP-101'],
             [comp,101],[comp-101]],
         syn:[third,sg],sort:[entity],
         fol:X^named(X,comp101)).
```

These two entries contain syntactic information, sortal information and information that is required to generate the TPTP representation. The syntactic information deals with number agreement between the subject and the verb as explained above. There is also orthographic information available that deals with approved variants of the input, for example: *COMP-101* instead of *COMP101*, etc.

In order to resolve the ambiguity of prepositional phrases in CNL, the lexicon includes sortal information for nouns, proper nouns and prepositions so that the correct role name can be derived for a prepositional phrase, for example in:

```
51. David Miller takes COMP101 on Monday
    on South Campus.
```

the preposition *on* is ambiguous on the surface level of the CNL. The sortal information in the lexicon for the proper noun and for the preposition disambiguates the two prepositional phrases *on Monday* and *on South Campus*. This results in two different role names in the DL representation:

```
52. (related e2 monday has_day)

53. (related e2 south_campus has_location)
```

Please note that the sortal information is not required, if we do not allow for this kind of prepositional phrases in the CNL – it is in theory possible to construct a reified version of a CNL sentence such as (51) that does not use prepositional phrases but offers the same expressivity, for example in the following way:

```
54. E1 is a take relation.
    E1 has David Miller as an agent.
    E1 has COMP101 as a theme.
    E1 has Monday as a day.
    E1 has South Campus as a location.
```

Let us assume that an ontology has already been constructed with the help of an ontology editor in a linguistically motivated way and that we are only interested in building an interface for querying the DL knowledge base. If that it the case, then we can automatically extract all concept names, role names, and names for individuals from the ontology using the following RacerPro functions:

```
55. (all-atomic-concepts)

56. (all-roles)

57. (all-individuals)
```

and populate the lexicon semi-automatically.

6.2 The CNL Grammar

The CNL grammar distinguishes three different modes: a TBox mode for processing terminological statements, an ABox mode for processing assertional statements and for generating answers to questions, and a query mode for processing questions. The grammar rules of the ABox mode are bidirectional (that means they can be used to analyse and generate statements) and some of these grammar rules can also used to process questions.

6.3 Processing Statements

Below is an example of an ABox grammar rule that gives a high-level overview about how sentences written in CNL are parsed and translated during the parsing process into TPTP formulas:

```
58. s0(mode:M,
       fol:LF1,
       gap:G1-G3,
       para:P1-P3,
       ant:A1-A3) -->
    n3(mode:M,
       syn:[subj,Per,Num],
       sort:_,
       fol:LF2^LF1,
       gap:[]-G2,
       para:P1-P2,
       ant:A1-A2),
    v3(mode:M,
       syn:[fin,Per,Num,_,_,_],
       fol:_E^LF2,
       gap:G1-G3,
       para:P2-P3,
       ant:A2-A3).
```

This top-level grammar rule takes part in the translation of the following ABox statement:

```
59. David Miller takes COMP101 on Monday.
```

The grammar rule (58) breaks this statement into a noun phrase *David Miller* and into a verb phrase *takes COMP101 on Monday* and combines the logical form of the noun phrase with the logical form of the verb phrase in a compositional way. Each word form in this sentence is associated with a partial logical form in the lexicon; these partial logical forms are merged (via unification) during the parsing process into a complete TPTP formula for the input sentence. The grammar rule contains various feature structures that are used for syntactic, semantic, and pragmatic purposes; one of these feature structures (`syn`) guarantees for example that the noun phrase agrees in person and number with the verb phrase; other feature structures are used to deal with the different types of sentences (`mode`), the composition of the logical form (`fol`), discontinuous constituents in the input string (`gap`), the construction of a paraphrase (`para`) that makes the interpretation of the sentence clear, and the collection of noun phrase antecedents (`ant`) for anaphora resolution.

The output of the grammar is a logical formula in TPTP format, in our case the following one:

```
60. input_formula(university,axiom,
     (? [A]: (named(A,david_miller) &
        (? [B]: (property(B,has_agent,A) &
              (event(B,take) & contemp(B,u)) &
          (? [C]: (named(C,comp101) &
                property(B,has_theme,C) &
          (? [D]: (timex(D,monday) &
                property(B,has_day,D)))))))))).
```

This logical formula is then further translated by the CNL processor into RacerPro's KRSS notation:

```
61. (instance e1 take)
    (related e1 david_miller has_agent)
    (related e1 comp101 has_theme)
    (related e1 monday has_day)
```

Note that the TPTP representation (60) contains additional information (`contemp(B,u)`) about the time when the utterance occurred since the CNL processor can handle tense (e.g. present tense and past tense). This information does not appear in the KRSS representation since DL can not deal with this kind of information and therefore all verb forms must be in present tense in a strict DL setting. However, the information about tense is potentially useful in other logical frameworks.

6.4 Processing Questions

The CNL processor distinguishes between *wh*-questions, *yes/no*-questions, and imperative constructions. It is well-known that the structure of questions in natural language is related in a systematic way to declarative sentences. For example, the declarative sentence (59) has the following functional structure:

```
62. Subject:  (David Miller)
    Verb:     (takes)
    Object:   (COMP101)
    Modifier: (on Monday)
```

In the case of *wh*-questions, the query word is related to one of these functional elements. For example in:

```
63. On what day does David Miller take
    COMP101?
```

a constituent in modifier position has been replaced by a query expression (*on what day*) and this query expression has been moved to the front of the sentence where it functions as a filler and is used together with a *do*-operator letting a gap behind:

```
64. [On what day]_filler does David Miller
    take COMP101 [ ]_gap?
```

When the CNL grammar processes (63), the query expression which acts as a filler for this gap is moved back into its original position. Schematically, the result of this movement process looks as follows:

```
65. Subject:  (David Miller)
    Verb:     (takes)
    Object:   (COMP101)
    Modifier: (on what day)
```

The TPTP representation for the question is constructed during the parsing process. That means at the same time when the syntactic movement of the query expression takes place. As already explained in the previous section, each word form is related to a partial logical form in the linguistic lexicon. For example, the processing of the query expression *on what day* triggers the following partial logical form:

```
66. (timex(F,G) & property(C,has_day,F))
```

and the processing of the entire question (63) results in the subsequent conjunctive query which contains the logical form for the query expression as a part:

```
67. input_formula(university,conjunctive_query,(
     (? [A]: (named(A,david_miller) &
       (? [B]: (named(B,comp101) &
        (? [C]: ((property(C,has_agent,A) &
                 (event(C,take) &
                 (property(C,has_theme,B) &
                  contemp(C,u)))) & (timex(F,G) &
                  property(C,has_day,F))))))))
      => answer(F))).
```

The CNL processor takes this TPTP formula and transforms it into a conjunctive nRQL query:

```
68. (retrieve (?2) (and (?1 david_miller
     has_agent) (?1 take) (?1 comp101
     has_theme) (?1 ?2 has_day)))
```

Note that *yes/no*-questions are treated in a similar way as *wh*-questions. The CNL processor first generates a normalised TPTP representation that looks similar to the representation of a declarative sentence, and then translates this representation into a Boolean conjunctive query, for example (69) into (70):

```
69. Does David Miller take COMP101?
```

```
70. (retrieve nil (and (?1 david_miller
     has_agent) (?1 take) (?1 comp101
     has_theme)))
```

The TPTP representation of a question is stored by the CNL processor and can be used as a starting point to answer questions as we will see in the next section.

6.5 Generating Answers

The relationship between declarative sentences and questions can be used for generating answers to questions in CNL. Internally, all TPTP formulas are annotated with syntactic information during the parsing process. For example, the TPTP formula (67) contains additional annotations (#) for all predicates that have been derived from content words, for example:

```
71. event(B,take)#[inf,_,_,pres,no,no]
```

Once the DL reasoner comes back with an answer for a question, the stored TPTP representation of a question is transformed into a TPTP representation for a declarative sentence which contains an update of the relevant syntactic annotations, for example:

```
72. event(B,take)#[fin,3rd,sg,pres,no,no]
```

That means the CNL processor can now generate the correct verbal form (`takes - fin`) for the answer string:

```
73. David Miller takes COMP101 on Monday.
```

Since the DL reasoner deals with subsumption hierarchies and concept definitions, there is no need to encode this ontological information in the linguistic lexicon. We get this terminological information for "free" from the DL reasoner during question answering. For example, the following questions:

```
74. Who studies COMP101?
75. Which person studies COMP101?
76. Which student studies COMP101?
```

return all the same answers because the query word *who* is more general than *person* and *person* subsumes *student*.

7 Writing in CNL

The writing of DL statements and questions in controlled natural language can be supported with the help of a predictive text editor that guides the writing process in CNL (see (Thompson et al. 2005, Chintaphally et al. 2007) for an introduction). We have implemented such editing techniques in the past (Schwitter et al. 2003) which have some similarities to editing techniques used in programming language environments. The basic idea here is to process the grammar rules with the help of a chart parser. A chart parser stores information about well-formed substrings as well as information about hypotheses of substrings in a chart (Kay 1980, Gazdar & Mellish 1989) and avoids repetition of work by looking up substrings in the chart instead of recomputing them. Complete and incomplete analyses are stored as so-called edges in the chart. Edges that correspond to partially recognized substrings are said to be active, while inactive edges represent completely recognised substrings. In our case, an active edge is a term of the form:

```
77. edge(Number,Pos1,Pos2,Category1,
          Found,[Category2|Categories]).
```

This term consists of the actual sentence number (`Number`) which serves as an index, a start position (`Pos1`) and an end position (`Pos2`) of a substring, the category (`Category1`) on the left-hand side of the grammar rule, a list of categories that have been found (`Found`) on the right-hand side of the grammar rule, and a list of remaining categories to be found (`[Category2|Categories]`). Chart parsing allows us – after processing of each new word form – to search through the active edges in the chart in order to collect the categories of those word forms that can follow the current input string.

Let us assume that the user is working in the query mode and plans to enter the following question into the text editor:

```
78. Where does David Miller take COMP101?
```

At the beginning, the text editor will display the initial lookahead information for questions. This information is grouped into three categories since the number of possible lexical entries for these categories is already large:

```
79. [ wh-question | yes/no-question |
       imperative construction ]
```

After clicking on the hypertext link for the `wh-question`, the editor will display the lookahead information that falls under this category:

```
80. [ Who | What | Which | Where | ... ]
```

The user can type (or select) one of these query words – in our case *where* – that is then sent to the CNL processor. The chart parser of the CNL processor initialises the chart, adds edges into the chart, and activates rules in order to expand the chart. This results in a set of new hypothesis that can be harvested for new lookahead information, in our case the editor will display the following function words:

```
81. [ does | is ]
```

After entering the operator *does*, the editor will display the following lookahead categories:

```
82. [ determiner | proper noun ]
```

At this point, the user has to make a decision whether he or she wants to enter a `determiner` or a `proper noun`. After entering the proper noun *David Miller* (or selecting it from a list of approved proper nouns) more lookahead information is displayed and this process continues until the structure of the question is complete.

Of course, an experienced user can switch off this lookahead facility and rely on the error messages that the parser generates if the user does not stick to the rules of the CNL. These predictive interface techniques guarantee that only sentences are added to the DL knowledge base and that only questions are used that conform 100% to the rules of the CNL.

8 Conclusions

This paper argued that a DL knowledge base should be constructed in a linguistically motivated way in order to support question answering in an optimal way. To achieve this the naming conventions used for constructing the ontology should be based on a set of well-defined linguistic patterns and on a terminology that occurs naturally in the application domain.

I showed that such a linguistically motivated naming convention makes it easier to create an ontology on the level of a machine-oriented controlled natural language. The controlled natural language can be used to express ABox and TBox statements as well as a query and feedback language. The presented approach has a number of attractive features: the controlled natural language looks like English and is easy to understand by humans and easy to process by machines; furthermore, the language is so precisely defined that it can be translated unambiguously into DL and thus is in fact a formal language. A logic-based grammar is used to process statements and questions, and the same grammar can run "backwards" taking logical formulas in TPTP notation as input and generate answers to questions in controlled natural language. The user does not need to learn and remember the rules of the controlled natural language since the writing process is supported with the help of predictive interface techniques.

It is important to note that not all parts of a DL knowledge base need to be constructed in controlled natural language. The aim of this paper was to illustrate that this is in principle possible. However, it might be the case that a knowledge engineer feels more comfortable using an ontology editor such as Protégé (Horridge et al. 2004) to develop the terminological knowledge for an application domain. As long as the knowledge engineer develops the TBox of this knowledge base in a linguistically motivated way – as sketched in this paper –, then this terminological knowledge can be combined with assertional knowledge expressed in controlled natural language. For some applications, it might be possible to extract this assertional knowledge from existing textual information and represent it in controlled natural language so that it can be checked easily and modified by a human (and still be processed by a machine).

I am convinced that controlled natural languages are a promising approach for creating DL knowledge bases and that many applications can benefit from such a high-level interface language. I am currently investigating how controlled natural languages can be used as an interface language to the Semantic Web, as a language for expressing business rules, and as a query and alert language in a decision-support system.

Acknowledgements

I would like to thank Anne Cregan, Alfredo Gabaldon, and Kevin Lee of NICTA's Knowledge Representation and Reasoning program at the Kensington laboratory in Sydney – where I spent my sabbatical – for many fruitful discussions related to this work. I would also like to thank three anonymous reviewers for their constructive and valuable comments.

References

Androutsopoulos, I., Ritchie, G., Thanisch, P. (2007), Natural Language Interfaces to Databases – an Introduction, in: Journal of Language Engineering, **1**(2), pp. 29–81.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (2002), The Description Logic Handbook, Cambridge University Press.

Bernardi, R., Calvanese, D., Thorne, C. (2007), Lite Natural Language, in: Proceedings of IWCS-7.

Bernstein, A., Kaufmann, E., Fuchs, N.E. (2004), Talking to the semantic web – a controlled English query interface for ontologies, in: 14th Workshop on Information Technology and Systems, pp. 212–217.

Bernstein, A., Kaufmann, E., Göhring, A., Kiefer, C. (2005), Querying ontologies: A controlled English interface for end-users, in: Proceedings of the 4th International Semantic Web Conference, pp. 112–126.

Chintaphally, V.R., Neumeier, K., McFarlane, J., Cothren, J., Thompson, C.W. (2007), Extending a Natural Language Interface with Geospatial Queries, in: IEEE Internet Computing, pp. 82–85.

Copestake, A., Sparck-Jones, K. (1998), Natural language interfaces to databases, in: Knowledge Engineering Review 5, pp. 225–249.

Cregan, A., Schwitter, R., Meyer, T. (2007), Sydney OWL Syntax – towards a Controlled Natural Language Syntax for OWL 1.1, in: C. Golbreich, A. Kalyanpur, and B. Parsia (eds.), 3rd OWL Experiences and Directions Workshop (OWLED 2007), Vol. 258, CEUR Proceedings.

Gazdar, G., Mellish, C. (1989), Natural Language Processing in Prolog. An Introduction to Computational Linguistics, Addison-Wesley.

Gruber, T. (1993), A translation approach to portable ontologies, in: Knowledge Acquisition, **5**(2), pp. 199–220.

Haarslev, V., Möller, R. (2003), Racer: A Core Inference Engine for the Semantic Web, in: Proceedings of EON2003, pp. 27–36.

Hart, G., Dolbear, C., Goodwin, J. (2007), Lege Feliciter: Using Structured English to represent a Topographic Hydrology Ontology, in: C. Golbreich, A. Kalyanpur and B. Parsia (eds), 3rd OWL Experiences and Directions Workshop (OWLED 2007), Vol. 258, CEUR Proceedings.

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, http://www.co-ode.org/resources/tutorials.

Horrocks, I., Patel-Schneider, P.F., van Harmelen, F. (2003), From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language, in: Journal of Web Semantics, **1**(1), pp. 7–26.

Kaljurand, K. (2007), Attempto Controlled English as a Semantic Web Language, PhD, Faculty of Mathematics and Computer Science, University of Tartu.

Kaufmann, E., Bernstein, A., Zumstein, R. (2006), Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs, in: Proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, pp. 980–981.

Kaufmann, E., Bernstein, A., Fischer, L. (2007), NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies, 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria, pp. 1–2.

Kaufmann, E., Bernstein, A. (2007), How Useful are Natural Language Interfaces to the Semantic Web for Casual End-users?, in: Proceedings of the 6th International Semantic Web Conference (ISWC 2007), Busan, Korea, pp. 281–294.

Kay, M. (1980), Algorithm Schemata and Data Structures in Syntactic Processing, CSL-80-12, Xerox Parc, Palo Alto, California.

Lopez, V., Motta, E., Uren, V. (2006), PowerAqua: Fishing the semantic web, in: The Semantic Web: Research and Applications, Lecture Notes in Computer Science, pp. 393–410.

Mellish, C., Sun, X. (2005), The Semantic Web as a Linguistic Resource, in: 26th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Peterhouse College, Cambridge, UK, December 12-14th.

Mithun, S., Kosseim, L., Haarslev, V. (2007), Resolving Quantifier and Number Restriction to Question OWL Ontologies, in: Proceedings of the 3rd International Conference on Semantic, Knowledge and Grid, IEEE Computer Society, pp. 218–223.

Noy, N., Rector, A. (2006), Defining N-ary Relations on the Semantic Web, W3C Working Group Note 12 April 2006, http://www.w3.org/TR/swbp-n-aryRelations/.

Patel-Schneider, P.F., Swartout, B. (1993), Description-Logic Knowledge Representation System Specification from the KRSS Group of the ARPA Knowledge Sharing Effort, 1 November.

Popescu, A.-M., Armanasu, A., Etzioni, O., Ko, D., Yates, A. (2004), Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability, in: Proceedings of the 20th International Conference on Computational Linguistics (COLING), Article No. 141.

Prud'hommeaux, E., Seaborne, A. (2008), SPARQL Query Language for RDF, W3C Recommendation 15 January 2008, http://www.w3.org/TR/rdf-sparql-query/.

Racer Systems (2007), RacerPro User's Guide, Version 1.9.2, Technical report, http://www.racer-systems.com.

Reichert, M., Linckels, S., Meinel, C., Engel, T. (2004), Student's Perception of a Semantic Search Engine, in: Proceedings of IEEE IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA), Porto, Portugal, pp. 139–147.

Schober, D., Kusnierczyk, W., Lewis, S.E., Lomax, J., Members of the MSI, PSI Ontology Working Groups, Mungall, C., Rocca-Serra, P., Smith, B., Sansone, S.-A. (2007), Towards naming conventions for use in controlled vocabulary and ontology engineering, in: Proceedings of Bio-Ontologies SIG Workshop 2007.

Schwitter, R., Ljungberg, A., Hood, D. (2003), ECOLE – A Look-ahead Editor for a Controlled Language, in: Proceedings of EAMT-CLAW03, May 15-17, Dublin City University, Ireland, pp. 141–150.

Schwitter, R., Tilbrook, M. (2004), Controlled Natural Language meets the Semantic Web, in: S. Wan, A. Asudeh, C. Paris (eds.), Australasian Language Technology Workshop 2004, pp. 55–62.

Schwitter, R., Tilbrook, M. (2006), Let's Talk in Description Logic via Controlled Natural Language, in: Logic and Engineering of Natural Language Semantics 2006, (LENLS2006), Tokyo, Japan, June 5-6th.

Schwitter, R., Kaljurand K., Cregan, A., Dolbear, C., Hart, G. (2008), A Comparison of three Controlled Natural Languages for OWL 1.1, 4th OWL Experiences and Directions Workshop (OWLED 2008 DC), Washington, 1-2 April.

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y. (2007), Pellet: A practical OWL-DL reasoner, in: Journal of Web Semantics, **5**(2), pp. 51–53.

Sutcliffe, G., Suttner, C.B. (1998), The TPTP Problem Library: CNF Release v1.2.1., in: Journal of Automated Reasoning, **21**(2), pp. 177–203.

Thompson, C.W., Pazandak, P., Tennant, H.R. (2005), Talk to Your Semantic Web, in: IEEE Internet Computing, **9**(6), pp. 75–79.

Tsarkov, D., Horrocks, I. (2006), FaCT^{++} Description Logic Reasoner: System Description, in: Proceedings of IJCAR 2006, LNAI 4130, Springer, pp. 292–297.

Wang, C., Xiong, M., Zhou, Q., Yu, Y. (2007), PANTO: A Portable Natural Language Interface to Ontologies, in: Proceedings of the 4th European Semantic Web Conference (ESWC 2007), Innsbruck, Austria, pp. 473–487.

Wessel, M., Möller, R. (2006), A Flexible DL-based Architecture for Deductive Information Systems, in: Proceedings of the FLoC'06 Workshop on Empirically Successful Computerized Reasoning, Seattle, USA, August 22, pp. 92–111.

Author Index

Atkinson, Ian, 51

Chamiel, Gil, 1
Chesñevar, Carlos Iván, 11

Gómez, Sergio Alejandro, 11

Homola, Martin, 21
Hooijmaijers, Dennis, 31

Johnstone, Ron, 51

Kutz, Oliver, 41

Lücke, Dominik, 41

Meyer, Thomas, iii
Mossakowski, Till, 41
Myers, Trina S., 51

Nguyen, Thai D., 61

Orgun, Mehmet A., iii

Pagnucco, Maurice, 1

Quan, Tho T., 61

Schwitter, Rolf, 71
Serafini, Luciano, 21
Simari, Guillermo Ricardo, 11
Stumptner, Markus, 31

Recent Volumes in the CRPIT Series