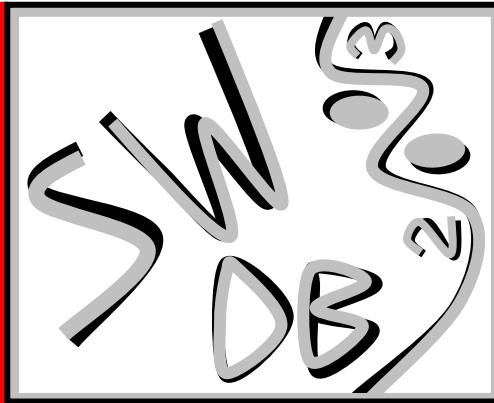


**Very
Large
Data
Bases**



**Proceedings of
SWDB' 03**

**The first International Workshop on
Semantic Web and Databases**

Co-located with VLDB 2003
Humboldt-Universität
Berlin, Germany
September 7-8, 2003

We appreciate the contributions from our sponsors:



OntoWeb Network

Organizers

Program Committee Chairs

Isabel F. Cruz
U. Illinois at Chicago, USA
(ifc@cs.uic.edu)

Vipul Kashyap
National Library of Medicine, NIH, USA
kashyap@nlm.nih.gov)

Proceedings and Publicity Chair

Stefan Decker
USC Information Sciences Institute, USA
stefan@isi.edu

Organization Chair

Rainer Eckstein
Humboldt University, Germany
Rainer.Eckstein@informatik.hu-berlin.de

PC Members

Karl Aberer, EPFL, Switzerland
Sibel Adali, Rensselaer Polytechnic I., USA
Paolo Atzeni, U. Rome Tre, Italy
Alex Borgida, Rutgers U., USA
Olivier Bodenreider, NLM-NIH, USA
Stéphane Bressan, National U. of Singapore
Christoph Bussler, Oracle, USA
Isabel Cruz, U. of Illinois at Chicago, USA
Umesh Dayal, HP Labs, USA
Stefan Decker, USC-ISI, USA
Max Egenhofer, U. Maine, USA
Rainer Eckstein, Humboldt U., Germany
Dieter Fensel, Institut für Informatik, Austria
Mary Fernandez, AT&T Labs - Research, USA
Susan Gauch, U. Kansas
Carole Goble, U. Manchester, UK

Rick Hull, Lucent Technology, USA
Vipul Kashyap, NLM-NIH, USA
Maurizio Lenzerini, U. Rome "La Sapienza", Italy
Ling Liu, Georgia Tech, USA
Robert Meersman, Vrije U., Belgium
John Mylopoulos, U. Toronto, Canada
Aris Ouksel, U. Illinois at Chicago, USA
Dimitris Plexousakis, U. Crete, Greece
Steve Ray, NIST, USA
Amit Sheth, U. Georgia and Semagix, USA
Surya Sripada, Boeing, USA
Munindar Singh, N. Carolina U., USA
V.S. Subrahmanian, U. Maryland, USA
Rudi Studer, U. Karlsruhe, Germany
Ram Sriram, NIST, USA
Clement Yu, U. Illinois at Chicago, USA

Semantic Web and Databases

September 7, 2003 (Sunday)	September 8, 2003 (Monday)
8:45-9:00 Welcome	9:00-10:10 Keynote Talk
9:00-10:10 Keynote Talk <u>Can we do better than Google? Using semantics to explore large heterogeneous knowledge sources</u> Anatole Gershman, Accenture Technology Labs	<u>From Semantic Search to Analytics and Discovery on Heterogeneous Content: Changing Focus from Documents and Entities to Relationships</u> Amit Sheth, University of Georgia and Semagix, Inc.
10:10-10:40 Semantic Web at Work Spatially Navigating the Semantic Web for User Adapted Presentations of Cultural Heritage Information in Mobile Environments <i>Marco Neumann, Dublin Institute of Technology, Ireland.</i>	10:10-10:40 Web Services ODE-SWS: A Semantic Web Service Development Environment <i>Oscar Corcho, Asunción Gómez-Pérez, Mariano Fernández-López, and Manuel Lama, Universidad Politécnica de Madrid, Spain, and Universidad de Santiago de Compostela, Spain.</i>
Text-Based Gene Profiling with Domain-Specific Views. Patrick Glenisson, Bert Coessens, Steven Van Vooren, Yves Moreau and Bart De Moor, Katholieke Universiteit Leuven, Belgium.	Applications of PSL to Semantic Web Services <i>Michael Gruninger, University of Maryland, College Park, USA.</i>
10:40-11:10 Coffee Break	10:40-11:10 Coffee Break
11:10-12:30 Context-Aware Systems Context-Aware Semantic Association Ranking <i>Boanerges Aleman-Meza, Chris Halaschek, I. Budak Arpinar, and Amit Sheth, University of Georgia, USA.</i>	11:10-12:30 Data Mining and Peer-to-Peer Systems H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. S. Castano, A. Ferrara, S. Montanelli, Università degli Studi di Milano, Italy.
I know what you mean: semantic issues in Internet-scale publish/subscribe systems <i>Ioana Burcea, Milenko Petrovic, and Hans-Arno Jacobsen, University of Toronto, Canada.</i>	A Collaborative Approach for Query Propagation in Peer-to-Peer Systems <i>Anne Doucet, Nicolas Lumineau, University of Paris 6, France.</i>
A Context-Oriented RDF Database <i>Mohammad-Reza Tazari, Computer Graphics Center, Dept. Mobile Information Visualization, Darmstadt, Germany.</i>	OntoMiner: Bootstrapping and Populating Ontologies from Domain Specific Web Sites <i>Hasan Davulcu, Srinivas Vadrevu, and Saravanakumar Nagarajan, Arizona State University, USA.</i>
An Adaptable Service Connector Model: <i>Gang Li, Yanbo Han, Zhuofeng Zhao, Jianwu Wang, Roland Wagner, Chinese Academy of Science, PRC, Fraunhofer, Germany</i>	Can Data Mining Techniques Ease The Semantic Tagging Burden? <i>Fabio Forno, Laura Farinetti¹, Sean Mehan, Politecnico di Torino, Italy, University of the Highlands and Islands, UK.</i>
12:30-2:00 Lunch (on your own)	12:30-2:00 Lunch (on your own)
2:00-3:10 Keynote Talk <u>Generic Model Management: A Database Infrastructure for Schema Manipulation</u> <i>Phil Bernstein, Microsoft Research, USA</i>	2:00-3:30 Formal Querying and Reasoning Formal aspects of querying RDF databases <i>Claudio Gutierrez, Carlos Hurtado, and Alberto Mendelzon, Universidad de Chile, Chile, and University of Toronto, Canada.</i>
3:10-3:40 Modeling Issues Building an integrated Ontology within SEWASIE system, <i>D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini, Università di Modena e Reggio Emilia, Italy and IEIIT-CNR, Italy.</i>	Event-Condition-Action Rule Languages for the Semantic Web. <i>George Papamarkos, Alexandra Poulouvassilis, Peter T. Wood, Birkbeck College, UK.</i>
Ontologies : A contribution to the DL/DB debate <i>Nadine Cullot, Christine Parent, Stefano Spaccapietra, and Christelle Vangenot, University of Burgundy, France, Swiss Federal Institute of Technology, Lausanne, Switzerland, University of Lausanne, Switzerland.</i>	Storing and Querying Ontologies in Logic Databases. <i>Timo Weithoener, Thorsten Liebig, and Guenther Specht, University of Ulm, Germany.</i>
Design Repositories for the Semantic Web with Description-Logic Enabled Services. <i>Joseph B. Kopena and William C. Regli, Drexel University, USA.</i>	Mediation of XML Data through Entity Relationship Models. <i>Irini Fundulaki and Maarten Marx, Bell Laboratories, USA, and University of Amsterdam, The Netherlands.</i>
3:40-4:10 Coffee Break	3:30-4:00 Coffee Break
4:10-5:30 RDF Storage and Implementation Issues Efficient RDF Storage and Retrieval in Jena2 <i>Kevin Wilkinson, Craig Sayers, and Harumi Kuno, HP Labs, USA.</i>	4:00-5:20 Integration and Interaction The ICS-FORTH SWIM: A Powerful Semantic Web Integration Middleware <i>V. Christophides, G. Karvounarakis, I. Koffina, G. Kokkinidis, A. Magkanaraki, D. Plexousakis, G. Serfiotis, and V. Tannen, University of Pennsylvania, USA, and Institute of Computer Science, FORTH, Greece.</i>
An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays. <i>Akiyoshi Matono, Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura, Nara Institute of Science and Technology, Japan, and Nagoya University, Japan.</i>	Semantic Representation of Contract Knowledge using Multi Tier Ontology <i>Vandana Kabilan, Paul Johannesson, Stockholm University and Royal Institute of Technology, Sweden.</i>
RDF Core: A component for effective management of RDF Models. <i>Floriana Esposito, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro, Università degli Studi di Bari, Italy.</i>	The Visual Semantic Web: Unifying Human and Machine Semantic Web Representations with Object-Process Methodology <i>Dov Dori, Technion, Israel and MIT, USA.</i>
Implementation of a Semantic Network Service (SNS) in the context of the German Environmental Information Network (gein®) <i>Thomas Bandholtz, Germany.</i>	Interaction and navigation for a document database: a concrete case study <i>Isabelle Berrien, François Laburthe, and Jean-David Ruvini, e-lab BOUYGUES SA, France.</i>
5:20-5:30 Closing	5:20-5:30 Closing

Table of Contents

Foreword	1
Invited Talks	3
Ontology and Ontology Maintenance	
<i>Spatially Navigating the Semantic Web for User Adapted Presentations of Cultural Heritage Information in Mobile Environments</i> Marco Neumann, Dublin Institute of Technology, Ireland.	9
<i>Text-Based Gene Profiling with Domain-Specific Views</i> Patrick Glenisson, Bert Coessens, Steven Van Vooren, Yves Moreau and Bart De Moor, Katholieke Universiteit Leuven, Belgium.	15
Context-Aware Systems	
<i>Context-Aware Semantic Association Ranking</i> Boanerges Aleman-Meza, Chris Halaschek, I. Budak Arpinar, and Amit Sheth, University of Georgia, USA.	33
<i>I know what you mean: semantic issues in Internet-scale publish/subscribe systems</i> Ioana Burcea, Milenko Petrovic, and Hans-Arno Jacobsen, University of Toronto, Canada.	51
<i>A Context-Oriented RDF Database</i> Mohammad-Reza Tazari, Computer Graphics Center, Dept. Mobile Information Visualization, Darmstadt, Germany.	63
<i>An Adaptable Service Connector Model</i> Gang Li, Yanbo Han, Zhuofeng Zhao, Jianwu Wang, Roland M. Wagner: Chinese Academy of Science, PRC., Fraunhofer Germany	79
Modeling Issues	
<i>Building an integrated Ontology within SEWASIE system</i> D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini, Università di Modena e Reggio Emilia, Italy and IEIIT-CNR, Italy.	91
<i>Ontologies : A contribution to the DL/DB debate</i> Nadine Cullot, Christine Parent, Stefano Spaccapietra, and Christelle Vangenot, University of Burgundy, France, Swiss Federal Institute of Technology, Lausanne, Switzerland, University of Lausanne, Switzerland.	109
RDF Storage and Implementation Issues	
<i>Efficient RDF Storage and Retrieval in Jena2</i> Kevin Wilkinson, Craig Sayers, and Harumi Kuno, Dave Reynolds, HP Labs	131
<i>An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays</i> Akiyoshi Matono, Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura, Nara Institute of Science and Technology, Japan, and Nagoya University, Japan.	151
<i>RDF Core: A component for effective management of RDF Models</i> Floriana Esposito, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro, Università degli Studi di Bari, Italy.	169
<i>Implementation of a Semantic Network Service (SNS) in the context of the German Environmental Information Network (gein®)</i> Thomas Bandholtz, Germany.	189

Web Services

ODE-SWS: A Semantic Web Service Development Environment
Oscar Corcho, Asunción Gómez-Pérez, Mariano Fernández-López, and Manuel Lama,
Universidad Politécnica de Madrid, Spain, and Universidad de Santiago de Compostela, Spain. **203**

Applications of PSL to Semantic Web Services
Michael Gruninger, University of Maryland, College Park, USA. **217**

Web Services

H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems
S. Castano, A. Ferrara, S. Montanelli, Università degli Studi di Milano, Italy. **231**

A Collaborative Approach for Query Propagation in Peer-to-Peer Systems
Anne Doucet, Nicolas Lumineau, University of Paris 6, France. **251**

OntoMiner: Bootstrapping and Populating Ontologies from Domain Specific Web Sites
Hasan Davulcu, Srinivas Vadrevu, and Saravanakumar Nagarajan, Arizona State University, USA. **259**

Can Data Mining Techniques Ease The Semantic Tagging Burden?
Fabio Forno, Laura Farinetti, Sean Mehan, Politecnico di Torino, Italy,
University of the Highlands and Islands, UK. **277**

Formal Querying and Reasoning

Formal aspects of querying RDF databases
Claudio Gutierrez, Carlos Hurtado, and Alberto Mendelzon, Universidad de Chile, Chile, and
University of Toronto, Canada. **293**

Event-Condition-Action Rule Languages for the Semantic Web
George Papamarkos, Alexandra Poulouvassilis, Peter T. Wood, Birkbeck College, UK. **309**

Storing and Querying Ontologies in Logic Databases
Timo Weithoener, Thorsten Liebig, and Guenther Specht, University of Ulm, Germany. **329**

Design Repositories for the Semantic Web with Description-Logic Enabled Services.
Joseph B. Kopena and William C. Regli, Drexel University, USA. **349**

Mediation of XML Data through Entity Relationship Models
Irina Fundulaki and Maarten Marx, Bell Laboratories, USA, and University of Amsterdam,
The Netherlands. **357**

Integration and Interaction

The ICS-FORTH SWIM: A Powerful Semantic Web Integration Middleware
V. Christophides, G. Karvounarakis, I. Koffina, G. Kokkinidis, A. Magkanaraki,
D. Plexousakis, G. Serfiotis, and V. Tannen, University of Pennsylvania, USA,
and Institute of Computer Science, FORTH, Greece. **381**

Semantic Representation of Contract Knowledge using Multi Tier Ontology
Vandana Kabilan, Paul Johannesson, Stockholm University and
Royal Institute of Technology, Sweden. **395**

*The Visual Semantic Web: Unifying Human and Machine Semantic Web
Representations with Object-Process Methodology*
Dov Dori, Technion, Israel and MIT, USA. **415**

Interaction and navigation for a document database: a concrete case study
Isabelle Berrien, François Laburthe, and Jean-David Ruvini, e-lab BOUYGUES SA, France. **435**

Foreword

The Semantic Web is a key initiative being promoted by the World Wide Web Consortium (W3C) as the next generation of the current web. Machine-understandable metadata is emerging as a new foundation for component-based approaches to application development. Within the context of reusable distributed components, Web services represent the latest architectural advancement. Such concepts can be synthesized providing powerful new mechanisms for quickly modeling, creating and deploying complex applications that readily adapt to real world need.

The objective of this workshop is to present database and information system research as they relate to the Semantic Web and more broadly, to gain insight into the Semantic Web technology as it relates to databases and information systems.

Isabel F. Cruz
U. Illinois at Chicago
USA

Vipul Kashyap
National Library of
Medicine, NIH, USA

Stefan Decker
USC Information
Sciences Institute, USA

Rainer Eckstein
Humboldt University,
Germany

Invited Talks

Can we do better than Google? Using semantics to explore large heterogeneous knowledge sources

Anatole Gershman
Accenture Technology Labs
USA

Abstract

Researchers in many fields use dozens of different rapidly growing on-line knowledge sources, each with its own structure and access methods. Successful research often depends on a researcher's ability to discover connections among many different sources of information. The popularity of Google suggests that high-quality indexing would provide a uniform method of access, although it still leaves researchers with vast, undifferentiated lists of results. Hence, the research challenge for semantic web designers: can a knowledge-based approach provide a better way for researchers to explore knowledge and discover useful insights for their research?

In this talk, I will use the example of bio-medical knowledge discovery to explore the key issues in semantic indexing of large amounts of heterogeneous information. I will propose a method and architecture for the creation of practical tools for semantic indexing and exploration.

The example I'll be using is the Knowledge Discovery Tool, or KDT, which contains a knowledge model of a large number of bio-medical concepts and their relationships: from genes, proteins, biological targets and diseases to articles, researchers and research organizations. Based on this model, the KDT index identifies over 2.5 million bio-medical entities with two billion relationships among those entities spanning 15 different knowledge sources. Clearly, the creation and maintenance of such an index cannot be done manually. KDT utilizes an extensive set of rules that cleanse, analyze and integrate data to create a uniform index.

Using its index, KDT presents the user with a uniform graphical browsing space integrating all underlying knowledge sources. This space is "warped" and filtered based on domain-specific rules customized for the needs of various groups of users, such as pharmaceutical researchers, clinicians, etc. Another customized set of rules discovers and graphically highlights potential indirect relationships among various entities that might be worth exploring (e.g., relationships between genes or between diseases). Finally, the tool enables several modes of collaboration among its users from annotations to activities tracking.

Currently, KDT is undergoing testing in two pilot settings: an early stage of the drug discovery process in a pharmaceutical company and a bio-medical academic research group.

About The Speaker

Anatole Gershman joined Accenture Technology Labs in 1989 and in 1997 became its overall Director of Research. Under his leadership, research at the laboratories is focusing on early identification of potential business opportunities and the design of innovative applications for the home, commerce and work place of the future. These include electronic commerce, high-performance virtual enterprise, knowledge management, and human performance support. To achieve these goals, the laboratories are conducting research in the areas of ubiquitous computing, human-computer interaction, interactive multimedia, information access and visualization, intelligent agents, and simulation and modeling.

Prior to joining Accenture, Anatole spent over 15 years conducting research and building commercial systems based on Artificial Intelligence and Natural Language processing technology. He held R&D positions at Coopers & Lybrand, Cognitive Systems, Inc., Schlumberger, and Bell Laboratories. In 1997, Anatole was named among the top 100 technologists in the Chicago area by Crain's Chicago Business. In 2000, Industry Week named Anatole one of the "R&D stars to watch."

Anatole studied Mathematics and Computer Science at Moscow State Pedagogical University and received his Ph.D. in Computer Science from Yale University in 1979.

Generic Model Management: A Database Infrastructure for Schema Manipulation

Philip A. Bernstein
Microsoft Research
USA

Abstract

Meta data management problems are pervasive in the development and maintenance of semantic web applications. Although solutions to these problems are similar to each other, today they are solved in an application-specific way and usually require much object-at-a-time programming. To make solutions more generic and easier to program, we propose a higher level interface, called Model Management. The main abstractions are models and mappings between models. It treats these abstractions as bulk objects and offers such operators as Match, Merge, Diff, Compose, Extract, and ModelGen. We will present an overview of Model Management and recent results about some of the operators.

About The Speaker

Phil Bernstein is a researcher at Microsoft Corporation. Over the past 25 years, he has been a product architect at Microsoft and at Digital Equipment Corp., a professor at Harvard University and Wang Institute of Graduate Studies, and a VP Software at Sequoia Systems. During that time, he has published over 100 articles on the theory and implementation of database systems, and coauthored three books, the latest of which is "Principles of Transaction Processing for the System Professional" (Morgan Kaufmann, 1997). He holds a B.S. from Cornell University and a Ph.D. from University of Toronto. A summary of his current research on meta data management can be found at <http://www.research.microsoft.com/~philbe>.

From Semantic Search to Analytics and Discovery on Heterogeneous Content: Changing Focus from Documents and Entities to Relationships

Amit Sheth
LSDIS Lab,
The University of Georgia and Semagix, Inc.
USA

Abstract

Research in search techniques was a critical component of the first generation of the Web, and has gone from academe to mainstream. Research and products supporting Semantic Search also look promising.

A second generation "Semantic Web" is being realized in one form of a scalable ontology-driven information system, where semantic metadata allow software to associate meaning with heterogeneous content. This is enabling a fundamental shift in focus from documents and entities within documents to discovering and reasoning about relationships. And it will transform the hunt for documents that humans can examine or analyze into a more automated content analysis, resulting in actionable information and insights into heterogeneous content. In this talk, we juxtapose the following shifts, to paint the exciting new possibilities:

- From documents and entities to relationships
- From techniques that focus on either unstructured data (text) or structured content to both types and semi-structured data
- From directly analyzing data to ontology based processes of creating high quality metadata and analyzing metadata
- From search and browsing for delivering relevant documents and locating entities within contents to discovering complex relationships and delivering actionable information with insights; from semantic search to analytics and discovery-based semantic applications

This talk will interleave academic research with state-of-the-art commercial uses, including tools and real-world applications and experiences. The critical challenge in dealing with the Web scale of ontologies (with huge description base/assertion set), metadata (very large RDF graphs), and their analysis in discovering relationship will be discussed.

About The Speakers

Amit Sheth is a Professor at the University of Georgia and CTO of Semagix, Inc. He started the LSDIS lab at Georgia in 1994. Earlier he served in R&D groups at Bellcore, Unisys, and Honeywell. He founded his second company, Taalee, in 1999 based on technology developed at the LSDIS lab, and managed it as CEO until June 2001. Following Taalee's acquisition/merger, he currently serves as CTO and a co-founder of Semagix, Inc. His research has led to three significant commercial products, several deployed applications and over 150 publications. More: <http://lsdis.cs.uga.edu/~amit>

Spatially Navigating the Semantic Web for User Adapted Presentations of Cultural Heritage Information in Mobile Environments

Marco Neumann

Digital Media Centre, Dublin Institute of Technology
Dublin 2, Ireland
marco.neumann@dit.ie

Abstract. The integration of local and global information is an essential requirement for future location-based services. The development of two technologies for mobile devices, namely positioning devices like GPS and wireless communication networks, is encouraging the development of new kinds of spatial- and context-aware applications. The CHI project investigates the applicability of these technologies for context-aware mobile computing applications that take advantage of new metadata-standards to enable semantic, user and device adapted services in the field of Tourism and Cultural Heritage management and presentation.

1 Introduction

The ability to query hyper-linked cultural heritage data sets, based on the user's context is a crucial functionality of future location-based services. The local information here is information about a place with a unique spatial and temporal relationship, which can be used to distinguish between places or information that only exist with regard to an explicit reference to a place and time. Global information is information that exists as conceptual knowledge but does not bear spatial reference e.g. structure of organisations, abstract knowledge about something applicable to recognise similarities or analogies in other contexts. As emphasised by Dey [1], context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves. The primary context in the CHI (Cultural Heritage Interfaces) [2] system is the position of the user in a virtual environment and a specific mobile device, which are integrated together with the user's preferences. The rationale of the CHI project is to retrieve automatically relevant data from a cultural heritage database based on the user's context, namely the current GPS coordinates, the display device limitations, the user preference and profile stored in a Vector data type. Furthermore, the system takes advantage of the available metadata information, encoded into the resource to extract the semantic value of existing documents for a selected area.

2 CHI System

The CHI project technology demonstrator (Figure 1) is implemented in a J2EE three-tier architecture, consisting of client layer, application server layer and database layer. The complete system communication between client and database layer is conducted through the application server layer. The Client VRML/JAVA sends the current location information in the form of Irish National Grid or Lat/Long coordinates via HTTP networking protocol to the Oracle application server along with the device characteristics and user profile and preferences. On the application server the query building and query result set formatting is executed against a spatially enabled Oracle database layer.

When the result of the query indicates the existence of content information, the system notifies the client about available documents with their respective Uniform Resource Identifiers (URI). The client then requests these documents automatically from the application server, which generates a XML JDOM document in memory and subsequently applies a specific XSLT style conversion to the resulting in a device-formatted document. The formatted document is then sent via HTTP protocol to the client device.

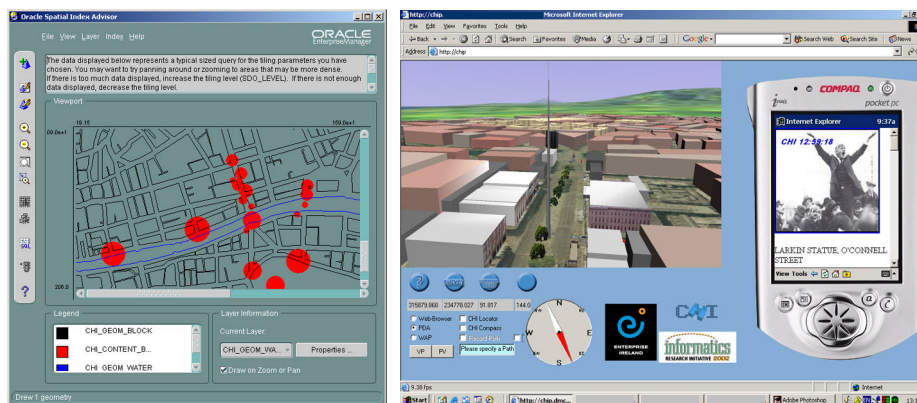


Figure 1 Oracle Spatial Index Advisor and CHI Technology Demonstrator

3 Semantic adaptation

After successful implementation of the spatial database components and visualization strategies and contextual information tailoring for mobile devices, the CHI project proposes the introduction of semantic layers to improve search query results. The concept of semantics has to be defined in the context of the CHI implementation. The use of the term “semantics” in regard to information systems is ambiguous and has led occasionally to false assumptions. Semantics in general describe the relations between

things and their varying significance for the receiver. This rather wide interpretation is not addressed in current research. However, one prominent and focused attempt at a pragmatic approach is the Semantic Web representation of data on the World Wide Web based on the Resource Description Framework (RDF). [3]

RDF integrates applications using XML for syntax and URI for naming. The Semantic Web therefore extends the current web where information is given well-defined meaning to better enable computers and people to work in cooperation. [4]

The accumulation of vast data resources on the World Wide Web has reached the limitations of conventional search approaches and new search strategies are needed. Current search procedures only account for simple string matching and boolean combinations of keywords. How much relevant information from unstructured data sources can be gained is up to the specification and capacity of the interpreter. To search for particular information in the current web architectures, the user is restricted to keyword matching or category browsing. The documents bear no explicit semantic information about themselves. To query documents on the web, search engines have to index available documents and this happens to be in most cases by parsing the complete document for keywords and Boolean combinations. Advanced search engines introduce new techniques like Latent Semantic Indexing where patterns in the text are recognized to assist in categorizing the document.

The semantics of documents and their respective knowledge domain relevance for the searching system remains untouched in most cases. Adopted approaches from artificial intelligence and knowledge management research promise to assist in exploiting the semantic value of online documents. For the most part the application of ontologies dominate present research where an ontology is used for the construction of complex models of relationships between data features and specialized domain area constraints to enhance query results.

The Semantic Web efforts by the World Wide Web Consortium [5] represent the attempt to extend the current web to give information well-defined meaning, therefore allowing machine processing and human evaluation.

3.1 CHI Semantic Query Scenario

While the user navigates the CHI system the client layer dispatches a query to the EJB middleware. The documents in a selected area are passed on to the semantic interpreter to determine the conceptual environment. The user's agent (i.e. the client) evaluates the semantic property and compares the conceptual environment of the document(s). The result is compared to the agent's conceptual definition to satisfy the initial search context. However, in order for ontologies to be shared, they must be congruent with other shared ontologies, otherwise they have to be compared and integrated, which is an active ontology research topic. [6]

The Semantic Web goes beyond these limitations and introduces a predefined semantic markup for web resources. The semantics are encoded in RDF (Resource Descriptions Framework) statements triples, consisting of Resource, Property and Value sometimes termed 'subject', 'predicate' and 'object' to describe a particular relationship. Semantics encoded into RDF triples can not only be used by human readers but also processed by machines. RDF therefore is mainly a mechanism to represent resources and their description in a direct-labeled graph (Figure 2).

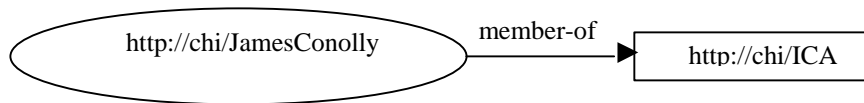


Figure 2 RDF direct-labeled graph

3.2 Ontology description and RDF Schema

To improve the information retrieval process and provide the user of the CHI system with more relevant information about available data resources the RDF metadata has to be related to the CHI domain ontology, which is implemented into a RDF Schema.

The query process (see figure 2) for semantic evaluation of RDF descriptions implemented on the Application Server session EJB and utilizes the Jena Java API for RDF [7] to generate the model graph depicted in Figure 3. For the purpose of the initial implementation of semantic exploitation, the CHI ontology only defines relationships between content documents stored in the Oracle database. Each content document can be accessed with a unique URL, which automatically adapts the database documents into a XML device independent tree structure and finally applies XSLT style sheet conversion to suit mobile device requirements for display.

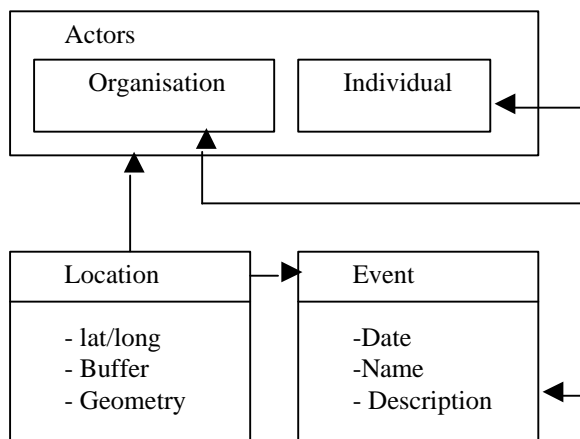


Figure 3 Relationship model of CHI entities

The introduction of RDF metadata allows the CHI System to locate, through querying RDF statements with RDQL query language, conceptual similar documents and selects only the spatially nearest related document for immediate display transformation. Additionally the user can take tangents and traverse the graph manually with the help of embedded hyperlinks in the cultural heritage document. The curator of cultural heritage content as well has the option to annotate data with time properties for allowing the introduction of narrative structuring of possible presentations resulting in pre-defined walk paths. The spatial database guides the user from one cultural heritage location to another with naive geographic directions: e.g. “go NE 300m” iteratively refined until the user has reached the next point of interest.

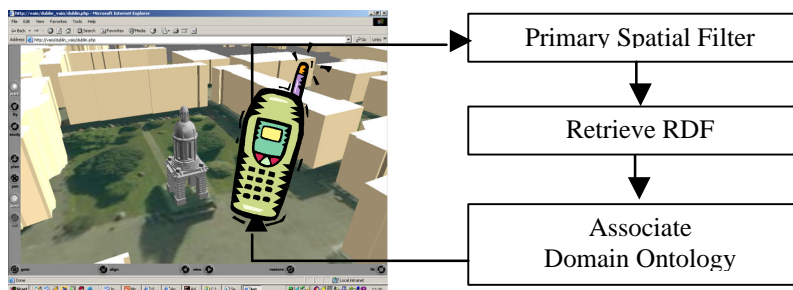


Figure 4 CHI semantic web information retrieval

4 Conclusion

In this paper we have presented the applicability of Semantic Web approaches to enhance query results within the CHI spatial database environments. The CHI project develops tools to respond to queries without the user of the system having to know about the conceptual structure. As noted in [8], given the lack of current approaches to exploit any form of semantics to assist users to accomplish their tasks, the introduction of metadata information capable of expressing the basic semantic relationships of resources and furthermore the integration into ontology-driven information systems is a desirable step to embrace decentralised web resources for information search. [9] Future location-based services have to take advantage of intelligent information retrieval strategies to exploit the potential of augmented information systems in mobile environments. [10] The exploitation of metadata and their integration into domain conceptualisations is one necessary condition.

5 Acknowledgement

Support for this research from Enterprise Ireland through the Informatics Programme 2001 on Digital Media is gratefully acknowledged.

References

1. Dey, Anind K.: Providing Architectural Support for Building Context-Aware Applications. PhD thesis, Georgia Institute of Technology, 2000.
2. Carswell, J.; Eustace, A.; Gardiner, K.; Kilfeather, E.; Neumann, M.: An Environment for Mobile Context-Based Hypermedia Retrieval. in Proceedings of 13th International Conference on Database and Expert Systems Applications (DEXA2002), IEEE CS Press, Aix en Provence, France. 2002. 532-536
3. Resource Description Framework (RDF) Model and Syntax Specification. 1999. URL <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
4. Berners-Lee, T., Hendler, J. and Lassila, O.: The Semantic Web. Scientific American 184(5): 2001. 34-43
5. Semantic Web. W3C. URL: <http://www.w3.org/2001/sw/>
6. Wache,H.;Vögele, T.;Visser,U.; Stuckenschmidt, H.; Schuster,G.; Neumann, H.; Hübner S.: Ontology-Based Information Integration: A Survey. The BUSTER Project, Intelligent Systems Group. Center for Computing Technologies. University of Bremen. 2001.
7. McBride, B.: Jena: A Semantic Web Toolkit. Hewlett-Packard Laboratories, Bristol, UK. IEEE INTERNET COMPUTING, November/December 2002. 55-59
8. Egenhofer, M. J.: Toward the Semantic Geospatial Web. National Center for Geographic Information and Analysis. Department of Spatial Information Science and Engineering. Department of Computer Science. Main. 2002.
9. Martin, Philippe: Knowledge Representation, Sharing, and Retrieval on the Web. In: Web Intelligence. Eds. Zhong, Ning; Liu, Jiming; Yao, Yiyu. 2003 pp. 243-276
10. Zipf, A. and Aras, H.: Proactive Exploitation of the Spatial Context in LBS - through Interoperable Integration of GIS-Services with a Multi Agent System (MAS). AGILE 2002. Int. Conf. on Geographic Information Science of the Association of Geographic Information Laboratories in Europe (AGILE). 04.2002. Palma. Spain.
11. Pradhan, S.: Semantic Location. Hewlett-Packard Laboratories, Palo Alto, CA, USA. Springer-Verlag London. 2000.
12. Farrugia, J.; Egenhofer, M. J.: Presentations and Bearers of Semantics on the Web , in Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2002). 2002. 408-412.
13. Fensel, Dieter; van Harmelen, Frank; Horrocks, Ian: OIL and DAML+OIL: Ontology Languages for the Semantic Web. In: Davies, John; Fensel, Dieter; van Harmelen , Frank: editors, Towards the Semantic Web – Ontology-based Knowledge Management. Wiley. London, UK. 2002.

Text-Based Gene Profiling with Domain-Specific Views

Patrick Glenisson, Bert Coessens, Steven Van Vooren, Yves Moreau, and Bart
De Moor

Departement Elektrotechniek, Katholieke Universiteit Leuven, Kasteelpark Arenberg
10, 3001 Leuven (Heverlee)
{pgleniss, bcoessen}@esat.kuleuven.ac.be

Abstract. The current tendency in the life sciences to spawn ever growing amounts of high-throughput assays has led to the situation where the interpretation of data and the formulation of hypotheses lag the pace with which information is produced. Although the first generation of statistical algorithms scrutinizing single, large-scale data sets found their way into the biological community, the great challenge to connect their results to the existing knowledge still remains. Despite the fairly large number of biological databases that is currently available, we find a lot of relevant information presented in free-text format (such as textual annotations, scientific abstracts, and full publications). Moreover, many of the public interfaces do not allow queries with a broader scope than a single biological entity (gene or protein). We implemented a methodology that covers various public biological resources in a flexible text-mining system designed towards the analysis of groups of genes. We discuss and exemplify how structured term- and concept-centric views complement each other in presenting gene summaries.

1 Introduction

The availability of the complete sequence of the human genome, along with those of several other model organisms, sparked a novel research paradigm in the life sciences. In ‘post-genome’ biology the focus is shifting from a single gene to the behavior of groups of genes interacting in a complex, orchestrated manner within the cellular environment. Recent advances in high-throughput methods enable a more systematic testing of the function of multiple genes, their interrelatedness, and the controlled circumstances in which these observations hold. Microarrays, for example, measure the simultaneous activity of thousands of genes in a particular condition at a given time. They enable researchers to identify potential genes involved in a great variety of biological processes or disease-related phenomena. As a result, scientific discoveries and hypotheses are stacking up, all primarily reported in the form of free text. A recent query with PUBMED¹ (the key bibliographic database in the life sciences) for the keyword

¹ <http://www.ncbi.nlm.nih.gov/PubMed/>

microarray showed that almost a third (i.e., about 1000) of the publications related to this technology is dated after January 2003. However, since the data and information, and ultimately the extracted knowledge itself, lack usability when offered in a raw state, various specialized database systems are designed to provide a complementary resource in designing, performing, or analyzing large-scale experiments. To date, we essentially distinguish two types of databases: the first type holds essential information, such as genomic sequence data, expression data, etc. without any extras (e.g., Genbank², ArrayExpress³); the second type offers curated annotations, cross-links to other repositories and multiple views on the same problem (e.g., LocusLink⁴, SGD⁵). Although meticulous upkeep of such databases is still struggling for due credit within the community, it is indispensable for the advancement of the field [1].

The process of successfully gaining insight into complex genetic mechanisms will increasingly depend on a complementary use of a variety of resources, including the aforementioned biological databases and specialized literature on the one hand, and the expert's knowledge on the other. We therefore consider the knowledge discovery process as cyclic, (i.e., requiring several iterations between heterogeneous information sources to extract a reliable hypothesis). For example, to date, linking up analyzed microarray data to the existing databases and published literature still requires numerous queries and extensive user intervention. This process of drilling down into the entries of hundreds of genes is notably inefficient and requires higher-level views that can more easily be captured by a (non-)expert's mind. Figure 1 depicts how this cyclic nature applies to the analysis of gene expression data.

Moreover, until now, it has been largely overlooked that there is little difference between retrieving an abstract from MEDLINE and downloading an entry from a biological database [2]. Fading boundaries between text from a scientific article and a curated annotation of a gene entry in a database is readily illustrated by the GeneRIF feature in LocusLink, where snippets of a relevant article pertaining to the gene's function are manually extracted and directly pasted as an attribute in the database. Conversely, we witness the emergence of richly documented web supplements accompanying a scientific publication that allow a virtual navigation through the results presented (see for example <http://www.esat.kuleuven.ac.be/neurdiff/> [3]). Additionally, through the use of hypertext, electronic publications will be able to offer more structured views. Hence, we should not expect the growing amount of free text to be halted by the advent of specialized repositories.

The broadening of the biologist's scope, along with the swelling amount of information, results in a growing need to move from single gene or keyword-based queries to more refined schemes that allow a deeper interaction between the user- and context-specific views of text-oriented databases.

² <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Nucleotide>

³ <http://www.ebi.ac.uk/arrayexpress/>

⁴ <http://www.ncbi.nlm.nih.gov/LocusLink/>

⁵ <http://www.yeastgenome.org/>

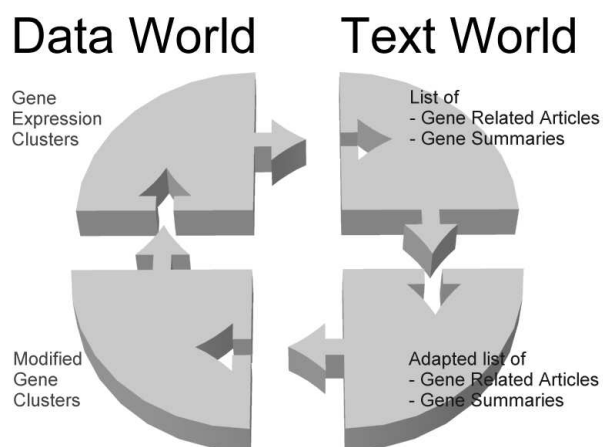


Fig. 1. Cyclic nature of the knowledge discovery process. It shows a high-level view of how it is embodied in microarray cluster analysis: starting from a cluster of genes resulting from a gene expression analysis (the ‘Data World’), the corresponding literature profiles are queried and analyzed (the ‘Text World’), resulting in either the addition of extra genes of interest or the omission of irrelevant genes. This updated cluster can subsequently be reanalyzed in expression space, which concludes a first cycle.

To facilitate such integrated views, controlled vocabularies that describe all properties of the underlying concepts are of great value when constructing interoperable and computer-parsable systems. A number of structured vocabularies have already arisen (most notably the Gene Ontology⁶) and, slowly but surely, certain standards are being adopted to store and represent biological data.

We can conclude that there is a certain urge towards a semantic biology web and although far from mature, some semantic web ideas have found their way into the bioinformatics community as means to knowledge representation and extraction.

Our general goal is to develop a methodology that can exploit and summarize vast amounts of textual information available in scientific publications and curated biological databases to support the analysis of *groups* of genes (e.g., resulting from gene expression analysis). As discussed above, the complexity of the domain at hand requires such a system to provide flexible views on the problem, as well as to extensively cross-link to other systems. As a result, we created a pilot text mining system, named TextGate, on top of a prevalent biological resource (LocusLink [4]) that aims, in the end, at implementing the interactive (or cyclic) nature of the knowledge discovery process.

A conceptual overview of the system is shown in Figure 2. We essentially indexed two sources of textual information. Firstly, we downloaded the entire

⁶ <http://www.geneontology.org>

LocusLink database⁷ and identified those fields that contain useful free-text information. Secondly, we collected all MEDLINE abstracts that were linked to by LocusLink. We indexed both information sources with two different domain vocabularies (one based upon Gene Ontology and one based upon the unique gene names found in the HUGO nomenclature database⁸). The resulting indices are used as basis for literature profiling and further query building on the set of genes of interest.

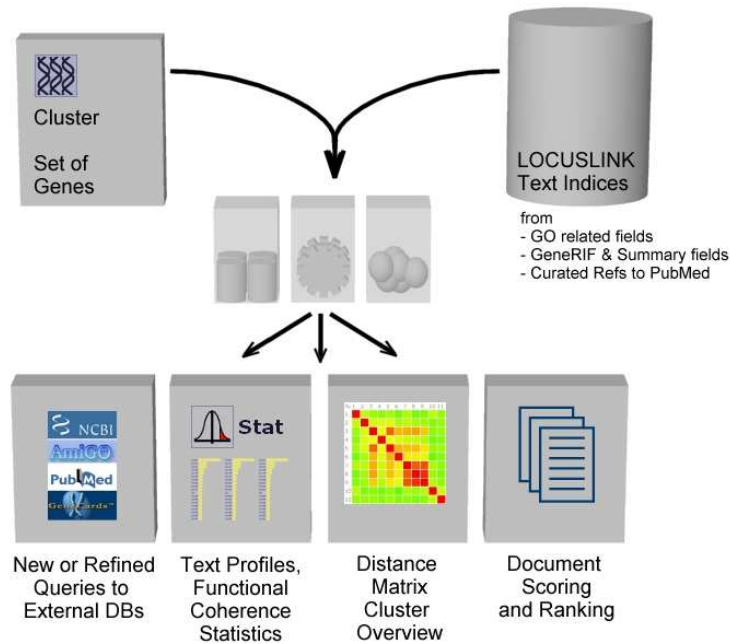


Fig. 2. Conceptual overview of the methodology behind the TextGate application. Indexing of textual gene information from the LocusLink database and abstracts from MEDLINE resulted in indices for respectively genes and documents. Starting from a gene or group of genes, the most relevant documents can be retrieved by comparing indices. Afterwards, statistical analysis and further queries can be performed.

Our work is related to several other reported and available systems. PubGene⁹ [5] is a database containing cooccurrence and cocitation networks of human genes derived from the full MEDLINE database. For a given set of genes it reports the literature network they reside in together with their high scor-

⁷ as of April 8 2003

⁸ <http://www.gene.ucl.ac.uk/hugo/>

⁹ <http://www.pubgene.org>

ing MESH headings¹⁰. MedMiner [6] retrieves relevant abstracts by formulating expanded queries to PUBMED. They use entries from the GeneCard database [7] to fish up additional relevant keywords to compose their query. The resulting filtered abstracts are comprehensively summarized and feedback loops are provided. GEISHA is a tool to profile gene clusters, again using the PUBMED engine, with an emphasis put on comprehensive summarization within a statistical framework [8]. This list of systems is not exhaustive and certainly does not encompass the spectrum of text-mining methods in genomics. Nevertheless, we believe that they well represent the first-generation systems oriented towards the considerations presented above.

The rest of this paper is organized as follows. In Section 2, we describe LocusLink and MEDLINE as our information sources and how the indexed information is used to query the information space we work in. In Section 3, we discuss the construction of our two domain vocabularies and their rationale. Section 4 describes the web-based application built upon the described methodology. In Section 5 the possibilities for query expansion and cross-linking to external data sources are explored. Finally, in Section 6, we provide two illustrative biological examples of a term-based summarization and a co-linkage analysis.

2 Information Selection

2.1 LocusLink as Gene Information Source

LocusLink [4] was used as the source of textual information about genes. LocusLink is a database that organizes information from collaborating public databases and from other groups within the National Center for Biotechnology Information¹¹ to provide a locus-centric¹² view of genomic information from human, mouse, rat, zebrafish, *Drosophila melanogaster*, and HIV-1.

Each LocusLink entry (one for each locus and 225,614 in total) has a unique LocusID and consists of a number of fields with information about a gene. Examples of fields include the originating organism, summary information about the gene, official and preferred gene symbols and names, OMIM¹³ [9] and PUBMED identifiers, and Gene Ontology annotations.

Although indexing these LocusLink entries can be done on all fields at once, we identified the subset that was most informative in a text-mining context. From this subset of fields we identified (possibly overlapping) groups of fields that constitute either a more specific or a more general *view* on the database. The basic aim of this design choice is that, although we wish to create a free-text index of each entry, we still want to preserve some of LocusLink's logical field structure.

¹⁰ MESH headings are a set of keywords attached by a manual indexer to each MEDLINE abstract.

¹¹ <http://www.ncbi.nlm.nih.gov/>

¹² A locus is a specific position on the chromosome.

¹³ OMIM is a catalog of human genes and genetic disorders.

2.2 MEDLINE as Document Information Source

As introduced before, MEDLINE is the largest bibliographic database containing over 12,000,000 citations in the biomedical literature from 1960 to present. Its great value arises from the fact that most citations have an abstract in English included.

We downsampled the MEDLINE collection to the subset of 73,172 documents found in the LocusLink entries. We assume this set to be reasonably trusted and gene-specific, and therefore it constitutes a good resource for conducting our experiments.

2.3 Textual Information in the Vector Space Model

In the vector space model [10], a text body is represented by a vector (or text profile) of which each component corresponds to a single (multi-word) term from the entire set of terms taken into account (i.e., the vocabulary, see Section 3). For every component a value denotes the presence or importance of a given term, represented by a weight. Indexing is the calculation of these weights:

$$\mathbf{d}_i = (w_{i,1}, w_{i,2}, \dots, w_{i,N}). \quad (1)$$

Each $w_{i,j}$ in the vector of document i is a weight for term j from the vocabulary of size N . This representation is often referred to as *bag-of-words*. In this paper we confine the discussion to the IDF weighting scheme, as it turned out to be a reasonable choice for modeling pieces of text comprising about 500 terms. The underlying assumption is that term importance is inversely proportional to frequency of occurrence. Let D be the number of documents in the collection and D_t be the number of documents containing term t , IDF is defined as:

$$\text{idf} = \log \left(1 + \frac{D}{D_t} \right). \quad (2)$$

Since, in principle, we can index the textual information from both LocusLink and MEDLINE abstracts with the same vocabulary, we can represent both *genes* and *documents* as vectors of term weights [11]. We distinguish two cases:

Combining multiple documents into a single gene profile

Since each gene can have one or more curated MEDLINE references associated to it in LocusLink, we combine these abstracts by taking the *mean* profile. This is illustrated in Figure 3.

Combining multiple gene profiles into a group profile

To summarize a cluster of genes and explore the most interesting terms they share, we compute the mean and variance of the terms over the group. Although simple, these statistics already reveal information on interesting terms characterizing the gene group.

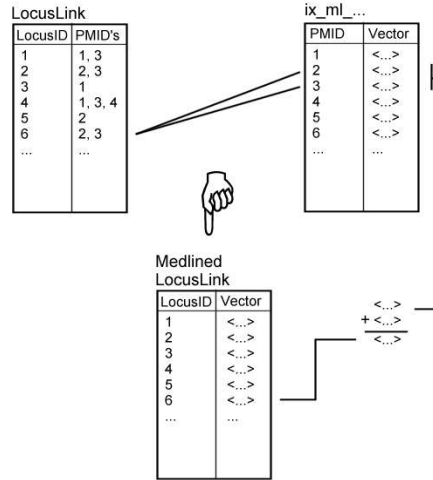


Fig. 3. Generating profiles for LocusID's via MEDLINE abstract text profiles. As described in Section 2, some indices are generated using the linked abstracts as sole source of information.

The vector representation of a gene or gene group can be used as a query to retrieve documents and vice versa. The similarity of one document to another, or of a document d_i to a query q , can be calculated using the cosine distance:

$$\text{sim}_{\cos}(d_i, q) = \frac{\sum_j w_{i,j} w_{q,j}}{\sqrt{\sum_j w_{i,j}^2} \sqrt{\sum_j w_{q,j}^2}}. \quad (3)$$

3 A Domain Vocabulary as Canvas to the Literature

Depending on the vocabulary chosen, the derived vector space model will be useful only within a given scope. Both the scale and diversity of the information contained in the MEDLINE database form a barrier to a fast, functional interpretation of groups of genes. A well-selected corpus, together with a domain- or problem-oriented vocabulary, already alleviates this problem in a first approximation. As explained above, the MEDLINE abstracts referred to in LocusLink constitute an acceptable, noise-free, and domain-specific collection. However, the information covered in this subset is still immensely vast. Although a corpus-derived vocabulary might be the first logical choice in a vector-based text mining approach, we constructed a tailored vocabulary in the light of the following issues:

Phrases

Are additional (statistical or Natural Language Processing) algorithms needed to extract multi-word terms or are external lists available?

Synonyms

Do we need synonym detection algorithms or can we resort to external lists?

Concept nomenclature

Genes, proteins, diseases, chemical substances, and so on are all possible concepts of interest to the user. Hence, concept-centric views or representations might be required instead of term-centric ones. Again the question comes up whether such lists are available or need to be generated.

Database integration

Can the choice of the vocabulary enhance interoperability with other databases or systems?

Structured representation

In which way can we ultimately model dependencies between the vector components?

These issues gave rise to the construction of two vocabulary types. The first type is term-centric. It was derived from Gene Ontology (GO) [12] and comprises 17,965 terms. GO is a dynamic controlled hierarchy of (multi-word) terms with a wide coverage in life science literature, and in genetics in particular. We considered it as an ideal source to extract a highly relevant and relatively noise-free domain vocabulary. Moreover, since GO is increasingly used to annotate databases, we envision an improved interoperability with other systems. We note that, at this time, we chose to neglect the structure defining the relations between the objects, as well as the limited amount of synonym information. Genes, however, are not only referred to by their symbols (e.g., TP53), but often also by their full name, typically constituting a phrase (e.g., tumor protein p53, Li-Fraumeni syndrome) that *can* bear an indication of its function. We extracted this information and merged it with the terms from GO.

A second vocabulary type is rather concept-centric (here, gene-centric) and was constructed with the screening of cooccurrence and colinkage in mind. In our setup *cooccurrence* denotes simultaneous presence of gene names within a *single* abstract, as in [5]. *Colinkage* is a weaker form of cooccurrence and screens for simultaneous presence in the *pool* of abstracts that are linked to a given group of genes. To this end, we derived from the HUGO database [9] (although LocusLink could equally have served as a resource) a vocabulary of all uniquely defined human gene symbols and their synonyms. Since these official gene symbols are frequently requested and used by scientists, journals and databases, we assume they will occur in scientific literature with high specificity. In total this vocabulary consists of 26,511 gene symbols.

4 The TextGate Application

As many combinations of restricted views and weighting schemes (Section 2), as well as representations (Section 3) are possible, we created a database of various literature indices. Within the scope of this paper this serves the goal of offering

a comprehensive interface to various views on the LocusLink database and the textual information captured inside. In a broader sense, this literature index database is part of an experimental platform to test and evaluate (combinations of) settings on a variety of biological annotation databases.

Different combinations of indexing schemes (by taking different fields of the LocusLink entries into consideration) and vocabularies show interesting possibilities towards analysis of genes and gene groups (as shown in Section 6 where three biological analysis cases are discussed).

Figure 4 shows the server architecture of the TextGate application. The different functionalities can be accessed via a browser or more directly by invoking the appropriate SOAP web service.

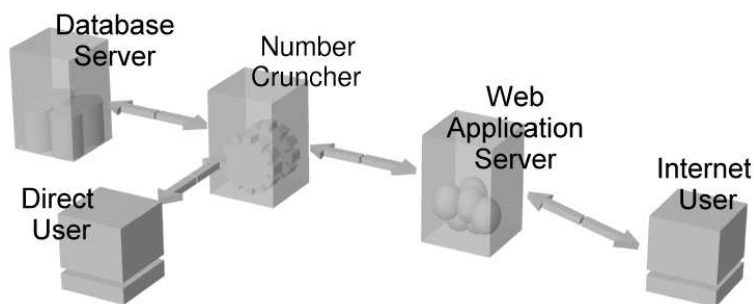


Fig. 4. Architectural overview of the TextGate knowledge discovery tool.

The user can perform a lookup of a single gene or a set of genes. In the case of profiling multiple genes, mean and variance statistics over the terms are displayed. Also, the application offers the possibility to output a distance matrix for a cluster of genes, which visualizes the distances (as calculated with Formula 3) between the text vectors of all genes in a cluster.

As said before, the functionalities of the application are also available via calls to a SOAP¹⁴ web service. The web service can be invoked by sending the appropriate SOAP request to the TextGate web service router. The SOAP message is interpreted by an Apache Tomcat server and specific requests are sent to a number cruncher that executes the necessary calculations (as can be seen in Figure 4).

This web service architecture allows for an easy integration of the functionalities of our tool with third-party applications. SOAP clients that invoke the service can be written in the programming language of choice. Currently, in our group, we already established an integrated web environment and web service

¹⁴ SOAP (Simple Object Access Protocol) is an XML-based W3C Proposed Recommendation for exchanging structured information in a decentralized, distributed environment.

architecture for microarray analysis, called INCLUSive [13], in which TextGate fits naturally.

5 Query Expansion and Hyperlinking

Essentially, TextGate adopts a ‘small world’ view by scrutinizing only a restricted set of textual information extracted by specific canvases on the literature (determined by the choice of the various representations discussed in Sections 2 and 3). In practice, relevant keywords, phrases, or gene names are only useful to a researcher if they can be linked (back) to existing biological resources.

In a first attempt to strengthen this desired connection, we implemented a query composer for a variety of other databases, among which PUBMED, GeneCards, and the Gene Ontology database are the most prominent, but also OMIM, UniGene, and 15 other sources belong to the list of possible destinations. Figure 5 visualizes this functionality.

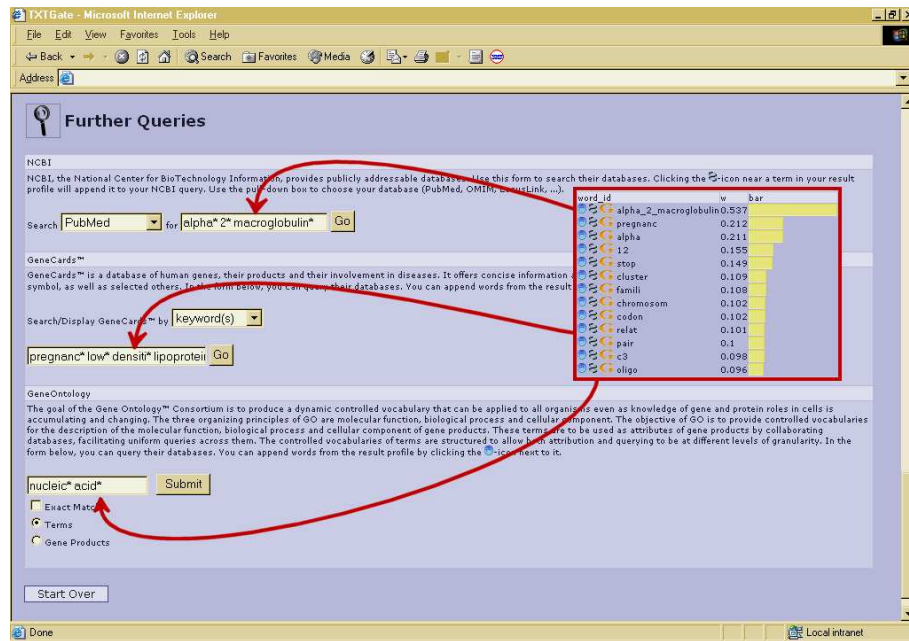


Fig. 5. The cyclic approach to knowledge mining by composing refined queries to a set of public databases.

6 Example Biological Cases

In this section, we wish to provide two illustrative examples of a term-based summarization and a colinkage analysis.

6.1 Gene Ontology and Transcriptional Up- and Downregulation

In this experiment, we generated two gene clusters based upon Gene Ontology (GO) annotations of human genes. To construct the first cluster, we retrieved all human genes that are annotated with the concept *transcription activation*. The second cluster are all human genes annotated with the concept *transcription repression*. Both concepts apply to the process of transcriptional regulation in the cell (see Figure 6). Whether a protein complex promotes or inhibits transcription of a gene, depends upon its constitution and environmental conditions. This makes the distinction between both concepts not a trivial task, since a protein can be active in a complex as inhibitor and as activator. The genes in both groups are enlisted in Table 1.

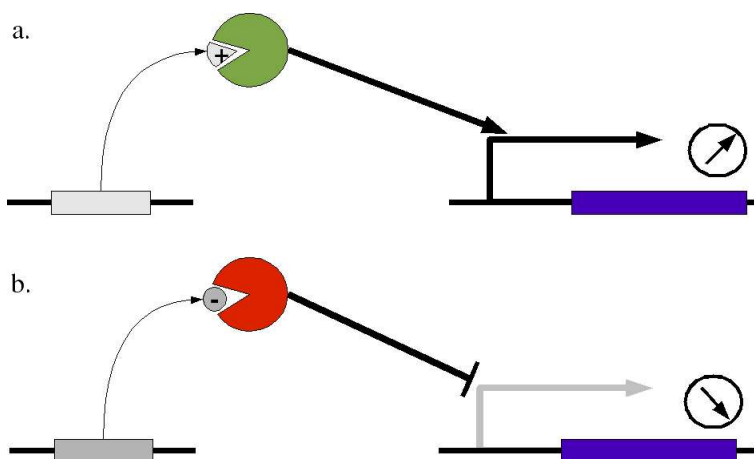


Fig. 6. The activation (a) and repression (b) of the transcription of a gene by DNA-binding protein complexes. The squares represent genes on the DNA. The circles represent protein complexes. In case (a), binding of an activator protein (produced by its corresponding gene) to the complex initiates, and subsequently activates transcription of a given gene while in case (b), binding of a repressor protein (produced by its corresponding gene) inhibits expression of that gene.

In the first place this indicates that our text-mining approach is reasonably trustable. As our confidence in these kind of methods will grow, one could invert the reasoning and consider this case to give an indication of whether or not the GO curators have made a good choice of splitting the concept of *transcriptional*

Table 1. Gene symbols and LocusLink identifiers for the two clusters of human genes that are annotated with respectively the Gene Ontology terms *transcription activation* and *transcription repression*.

Activation cluster		Repression cluster	
<i>Gene Symbol</i>	<i>LocusID</i>	<i>Gene Symbol</i>	<i>LocusID</i>
BRCA1	672	BTF	9774
BRCA2	675	DMAP1	55929
CGBP	30827	DNMT3L	29947
COPEB	1316	EED	8726
EDF1	8721	EPC1	80314
ELF1	1997	HDAC4	9759
ELF2	1998	HDAC6	10013
EPC1	80314	IFI16	3428
ETV4	2118	LRRFIP1	9208
FOXC1	2296	MBD1	4152
FOXD3	27022	MBD2	8932
HNRPD	3184	NAB1	4664
HOXA9	3205	NRF	55922
HOXC9	3225	NSEP1	4904
HOXD9	3235	PIASY	51588
KLF2	51713	RBAK	57786
MADH1	4086	REST	5978
MADH5	4090	RING1	6015
MITF	4286	THG-1	81628
MYB	4602	UBP1	7342
NSBP1	79366	ZFHX1B	9839
ONECUT1	3175	ZNF24	7572
RREB1	6239	ZNF253	56242
SEC14L2	23541	ZNF33A	7581
SUPT3H	8464	ZNFN1A4	64375
TITF1	7080		
TP53BP1	7158		
TRIP4	9325		
UBE2V1	7335		
ZNF38	7589		
ZNF148	7707		
ZNF398	57541		

regulation in transcription activation and transcription repression: if for those two different clusters TextGate shows that in essence the same terms occur this would mean that there is not really a significant difference between the genes GO associated to *transcription activation* and *transcription repression*. If, however, specific terms linked to activation and repression respectively occur for the activation cluster and the repression cluster, then making two taxons under *transcriptional regulation* was a good choice.

In Table 2, the term ranking and variance are shown for the activation cluster (top of the table) and the repression cluster (bottom). We see an obvious difference in term occurrence. For the activation cluster, **transcript_activ** ranks third place, and for the repression cluster, **repressor** and **repress** rank first and second, respectively. Note that **dna_bind** scores high for both clusters because DNA-binding is a general aspect of transcriptional regulation.

6.2 Colinkage of Colon Cancer Genes

In Section 3 we discussed how changing the way domain vocabularies and index tables are constructed provides us with a different view on the information. Using only the gene names from the HUGO database [9] as domain vocabulary, we can take a specific stance towards investigating colinkage of genes.

For this test case, we constructed a set of genes by consulting a textbook on molecular biology [14] and choosing genes that are related to colon cancer manually. This set was then provided to TextGate using the colinkage index. The set of genes is shown in Table 3. The results are shown in Table 4.

To validate this result, we verified that these gene names indeed turn up in the literature in relation to colon cancer.

The highest scoring gene is the *CD44 antigen*. This gene is indeed related to colon cancer, as shown in a paper by Barshishat *et al.* [15].

The second ranking gene name is *UBE3A* (ubiquitin protein ligase E3A). At first sight, it is not directly related to colon cancer, but after closer investigation of the available literature, we found that this gene is involved in degradation of TP53, which plays a crucial role in the regulation of cell division (mitosis) [16]. This explains the detection of frequent co-citation.

7 Conclusion and Future Work

As contemporary biology is evolving towards an information science, integrative views on biological problems will be of increasing importance. Integration is a broad term and is understood differently in the database community than for instance in the field of machine learning. Our perspective on integration was adopted with both the (presumed) cyclic nature of the knowledge discovery process and of a text-mining application in mind. We created various indices on two text-oriented databases (the annotation database LocusLink and the literature repository MEDLINE) that enabled text summarization of multiple genes at once. Supported by grateful realizations in the development of annotation

Table 2. For the *transcription activation* and *transcription repression* clusters we show the ranking of the 20 terms with the highest mean (left side) and the ranking of the 20 with the highest variance (right side). We note the presence of some noise due to the nature of the term extraction process.

Activation cluster			
<i>Term</i>	<i>Mean</i>	<i>Term</i>	<i>Variance</i>
transcript_factor	0.205	ovarian	0.011
dna_bind	0.188	thyroid	0.007
transcript_activ	0.139	site_select	0.005
nuclear	0.129	h3	0.005
transcript	0.125	zinc	0.005
promot	0.117	p53	0.004
bind	0.113	ey	0.004
tumor	0.113	hepatocyt	0.004
domain	0.112	melanocyt	0.004
famili	0.11	cluster	0.004
chromosom	0.106	prime	0.004
site	0.098	bridg	0.004
pair	0.096	transcript_factor	0.003
involv	0.095	transform_growth_factor_beta	0.003
region	0.093	retino_acid_metabol	0.003
yeast	0.092	tumor_suppressor	0.003
two	0.09	ubiquitin_conjug_enzym	0.003
zinc	0.088	leukemia	0.003
contain	0.088	7	0.003
map	0.087	pigment	0.003
Repression cluster			
<i>Term</i>	<i>Mean</i>	<i>Term</i>	<i>Variance</i>
repressor	0.238	methyl_cpg_bind	0.019
repress	0.205	deacetylas	0.013
dna_bind	0.172	cytosin_5	0.009
zinc	0.164	repressor	0.009
transcript_repressor	0.158	histon	0.008
deacetylas	0.157	polycomb_group	0.008
transcript_factor	0.151	dna_methyl	0.006
domain	0.147	ring	0.006
histon	0.127	zinc	0.006
transcript	0.123	transcript_repressor	0.005
yeast	0.116	methyltransferas	0.005
famili	0.109	silenc	0.005
gene_express	0.109	hi	0.005
methyl_cpg_bind	0.105	interferon_gamma	0.005
region	0.104	stat2	0.004
nucleu	0.104	cell_structur	0.004
interact	0.103	leucin_metabol	0.004
protein_metabol	0.1	polycomb	0.004
bind	0.1	lrr	0.004
line	0.095	methyl	0.004

Table 3. A set of seven genes involved in colon cancer.

HUGO Name	LocusID
k-RAS2	3845
NEU1	4758
MYC	4609
APC	324
DCC	1630
P53	7157
MSH2	4436

Table 4. For the colon cancer cluster we show the ranking of the 20 colinkage concepts with the highest mean (left side) and the ranking of the 20 colinkage concepts with the highest variance (right side). We note the presence of some noise due to the nature of the concept extraction process.

Gene	Mean	Gene	Variance
cd44	0.446	myc	0.013
ube3a	0.429	pten	0.012
i	0.344	apc	0.01
wwox	0.28	tp53	0.01
sparc	0.27	dcc	0.009
pax6	0.234	msh2	0.005
wa	0.232	pax6	0.004
rieg2	0.223	ra	0.003
at	0.162	wwox	0.003
nr4a2	0.156	map	0.003
ha	0.136	pms2	0.003
gstz1	0.125	rieg2	0.003
msh2	0.081	mlh1	0.003
1	0.081	12	0.003
3	0.078	ha	0.002
all	0.077	wa	0.002
5	0.075	hla	0.002
kptn	0.066	all	0.002
tp53	0.065	nr4a2	0.002
nup214	0.064	gstz1	0.001

standards, nomenclature conventions, and ontologies, TextGate is able to formulate sensible queries to a variety of other resources (including back the GO). However, the system is far from complete, and represents only a first step in the construction of a knowledge discovery platform. Our mid-term challenges include:

Extension to an IR engine

At this point TextGate uses the index tables in a gene-centric way to summarize and link information. As biological experiments are always carried out in a particular context, allowing term-centric queries (see e.g., the recently established TREC¹⁵ track) would further enhance the usability of the system. This would fully close the cycle between terms, genes, documents, and database annotations.

Extension of the conceptual representations

Up to now we neglected the structure of GO. Embedding its structure as well as adding additional ontologies for functional genomics¹⁶, or biomedicine¹⁷ would provide more structured views on information. A second improvement involves the incorporation of improved semantics (e.g., negations) in our system.

Finally, since the core functionality of the TextGate system is also provided as a SOAP service, it can seamlessly be integrated with other systems, primarily the expression analysis pipeline currently present in our lab¹⁸.

Acknowledgments

P.G. and B.C. are research assistants of the K.U.Leuven. S.V.V is an intern in fulfillment of the Master in Bioinformatics Program at the K.U.Leuven. Y.M. is a post-doctoral researcher of FWO-Vlaanderen and assistant professor at the K.U.Leuven. B.D.M. is a full professor at the K.U.Leuven. Research supported by Research Council K.U.Leuven: [GOA-Mefisto 666, IDO (IOTA Oncology, Genetic networks), several PhD/postdoc and fellow grants]; Flemish Government: [FWO: PhD/postdoc grants, projects G.0115.01 (microarrays/oncology), G.0240.99 (multilinear algebra), G.0407.02 (support vector machines), G.0413.03 (inference in bioi), G.0388.03 (microarrays for clinical use), G.0229.03 (ontologies in bioi), research communities (ICCoS, ANMMM)]; AWI: [Bil. Int. Collaboration Hungary/Poland]; IWT: [PhD Grants, STWW-Genprom (gene promoter prediction), GBOU-McKnow (Knowledge management algorithms), GBOU-SQUAD (quorum sensing), GBOU-ANA (biosensors)]; Belgian Federal Government: [DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006))]; EU: [CAGE]; ERNSI; Contract Research/agreements: [Data4s, Electrabel, Elia, LMS, IPCOS, VIB]. We acknowledge Peter Antal for starting up this research direction.

¹⁵ <http://trec.nist.gov/>

¹⁶ for example: <http://www.sofg.org/index.html>

¹⁷ for example: <http://www.nlm.nih.gov/research/umls/umlsmain.html>

¹⁸ <http://www.esat.kuleuven.ac.be/inclusive/>

References

1. Navarro, D., Niranjana, V., Peri, S., Jonnalagadda, C., Pandey, A.: From biological databases to platforms for biomedical discovery. *Trends Biotechnol.* **21** (2003) 263–268
2. Gerstein, M., Junker, J.: Blurring the boundaries between scientific papers and biological databases. *Nature Online*, <http://www.nature.com/nature/debates/e-access/Articles/gerstein.html> (web debate, on-line 7 May 2001)
3. Dabrowski, M., Aerts, S., Hummelen, P.V., Craessaerts, K., De Moor, B., Annaert, W., Moreau, Y., De Strooper, B.: Gene profiling of hippocampal neuronal culture. *J. Neurochem.* **85** (2003) 1279–1288
4. Pruitt, K., Maglott, D.: RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Res.* **29** (2001) 137–140
5. Jenssen, T., Laegreid, A., Komorowski, J., Hovig, E.: A literature network of human genes for high-throughput analysis of gene expression. *Nature Genet.* **28** (2001) 21–28
6. Tanabe, L., Scherf, U., Smith, L., Lee, J., Hunter, L., Weinstein, J.: MedMiner: An internet text-mining tool for biomedical information, with application to gene expression profiling. *BioTechniques* **27** (1999) 1210–1217
7. Rebhan, M., Chalifa-Caspi, V., Prilusky, J., Lancet, D.: GeneCards: A novel functional genomics compendium with automated data mining and query reformulation support. *Bioinformatics* **14** (1998) 656–664
8. Blaschke, C., Oliveros, J., Valencia, A.: Mining functional information associated with expression arrays. *Funct. Integr. Genomics* **1** (2001) 256–268
9. McKusick, V.: *Mendelian Inheritance in Man. A Catalog of Human Genes and Genetic Disorders.* Twelfth edn. Johns Hopkins University Press (1998)
10. Baeza-Yates, R., Ribeiro-Neto, B.: *Modern Information Retrieval.* ACM Press / Addison-Wesley (1999)
11. Glenisson, P., Antal, P., Mathys, J., Moreau, Y., Moor, B.D.: Evaluation of the vector space representation in text-based gene clustering. In: *Proceedings of the Pacific Symposium on Biocomputing. Volume 8.* (2003) 391–402
12. The Gene Ontology Consortium: Gene Ontology: tool for the unification of biology. *Nature Genet.* **25** (2000) 25–29
13. Coessens, B., Thijs, G., Aerts, S., Marchal, K., Smet, F.D., Engelen, K., Glenisson, P., Moreau, Y., Mathys, J., Moor, B.D.: INCLUSive - a web portal and service registry for microarray and regulatory sequence analysis. *Nucleic Acids Res.* **31** (2003) 3468–3470
14. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: *Molecular Biology of the Cell.* Fourth edn. Garland Science Publishing (2002)
15. Barshishat, M., Levi, I., Benharroch, D., Schwartz, B.: Butyrate down-regulates CD44 transcription and liver colonisation in a highly metastatic human colon carcinoma cell line. *Br. J. Cancer* **87** (2002) 1314–1320
16. Levine, A.: p53, the cellular gatekeeper for growth and division. *Cell* **88** (1997) 323–331

Context-Aware Semantic Association Ranking¹

[Boanerges Aleman-Meza](#), [Chris Halaschek](#), [I. Budak Arpinar](#), and [Amit Sheth](#)

[Large Scale Distributed Information Systems \(LSDIS\) Lab](#),

Computer Science Department, University of Georgia, Athens, GA 30602-7404
{boanerg, ch, budak, amit}@cs.uga.edu

Abstract. Discovering complex and meaningful relationships, which we call Semantic Associations, is an important challenge. Just as ranking of documents is a critical component of today's search engines, ranking of relationships will be essential in tomorrow's semantic search engines that would support discovery and mining of the Semantic Web. Building upon our recent work on specifying types of Semantic Associations in RDF graphs, which are possible to create through semantic metadata extraction and annotation, we discuss a framework where ranking techniques can be used to identify more interesting and more relevant Semantic Associations. Our techniques utilize alternative ways of specifying the context using ontology. This enables capturing users' interests more precisely and better quality results in relevance ranking.

1 Introduction

The focus of contemporary data and information retrieval systems has been to provide efficient support for the querying and retrieval of data. Search engines have made good progress in the ability to locate one of the relevant pieces of information from among huge information on the Web. There has also been noteworthy progress in metadata extraction, which involves recognition of entities such as names of persons, locations, and in some cases, domain specific attributes of entities. Semantic metadata are metadata elements that describe in context, domain specific information offering additional insight about a document or other content items. For example, relevant semantic metadata relating to a content item about a terrorist organization could be countries the organization is active in, known terrorist activities, key organizational members, number of members on watch lists, etc. The progress in information retrieval or search does not extend to support effective decision-making and knowledge discovery.

Due to the increasing move from data to knowledge, and the increasing popularity of the vision of the Semantic Web [3], there is significant interest and ongoing work, in automatically extracting and representing the metadata as semantic annotations to documents and services on the Web [18,8,7]. Several communities such as the Gene Ontology Consortium, Federal Aviation Administration (Aviation Ontol-

¹ This work is funded by NSF-ITR-IDM Award # 0219649 titled "Semantic Association Identification and Knowledge Discovery for National Security Applications."

ogy), Molecular Biology Ontology Working Group, Stanford University's Knowledge Systems Lab (Enterprise Ontology), are also coming together, to effectively conceptualize the domain knowledge, and enable standards for exchanging, managing and integrating data more efficiently. Research in the Semantic Web has also spawned several commercially viable products through companies such as [Semagix](#) [17,14] and Ontoprise [15] to name a few.

Given these developments, the stage is now set for the next generation of technologies, which will facilitate getting actionable knowledge and information from massive data sources thereby assisting in information analysis. Many users try to analyze information by either browsing the information space, or using a search engine. Search engine based systems only locate documents based on keywords or key phrases. These approaches are not very representative of what the user actually wants. Therefore, most of the retrieved documents are either irrelevant, or contain the information buried deep within other data. The onus is then on the user, who must decide, which of the retrieved documents are relevant, and then use their mental model, of the information they are looking for, in order to obtain the relevant information.

The main goal of this work is to ease the process of analyzing across different sources of data and enable users to uncover previously unknown and potentially interesting relations (or associations) [2,19]. In the quest for finding associations, it is also possible to find too many of them between the entities. Therefore, it is also important to locate interesting and meaningful relations and to rank them before presenting to the user.

1.1 Semantic Associations

The associations lend meaning to information, making it understandable and actionable, and provide new and possibly unexpected insights. When we consider data on the Web, different entities can be related in multiple ways that cannot be pre-defined. For example, a "professor" can be related to a "university", "students", "courses", and "publications"; but s/he can also be related to other entities by different relations like *hobbies*, *religion*, *politics*, etc. In the semantic Web vision, the Resource Description Framework (RDF) data model [11] provides a framework to capture the meaning of an entity (or resource) by specifying how it relates to other entities (or classes of resources). Each of these relationships between entities is what we call a "semantic association" and users can formulate queries to find the semantic association(s). For example, semantic association queries in flight security domain may include the following:

1. Is the passenger known to be associated with an organization on the watch list?
2. Does the passenger work for an organization that is known to sponsor an organization on a watch-list?
3. Is there a connection between the passenger and one or more passengers on the same flight or different flights?

Most of useful semantic associations involve some intermediate entities and associations. Relationships that span several entities may be very important in domains such

as national security, because this may enable analysts to see the connections between disparate people, places and events.

Semantic associations are based on intuitive notions such as connectivity and semantic similarity. In [2], we have presented a formalization of semantic associations over metadata represented in RDF. Concepts are linked together by properties denoted by arcs and labeled with the property name. Different types of semantic associations in an RDF graph are formally defined in the following:

Definition 1 (Semantic Connectivity): Two entities e_1 and e_n are semantically connected if there exists a sequence $e_1, P_1, e_2, P_2, e_3, \dots, e_{n-1}, P_{n-1}, e_n$ in an RDF graph where $e_i, 1 \leq i \leq n$, are entities and $P_j, 1 \leq j < n$, are properties.

Definition 2 (Semantic Similarity): Two entities e_1 and f_1 are semantically similar if there exist two semantic paths $e_1, P_1, e_2, P_2, e_3, \dots, e_{n-1}, P_{n-1}, e_n$ and $f_1, Q_1, f_2, Q_2, f_3, \dots, f_{n-1}, Q_{n-1}, f_n$ semantically connecting e_1 with e_n and f_1 with f_n , respectively, and that for every pair of properties P_i and $Q_i, 1 \leq i < n$, either of the following conditions holds: $P_i = Q_i$ or $P_i \subseteq Q_i$ or $Q_i \subseteq P_i$ (\subseteq means `rdf:subPropertyOf`). We say that the two paths originating at e_1 and f_1 , respectively, are semantically similar.

Definition 3 (Semantic Association): Two entities e_x and e_y are *Semantically Associated* if e_x and e_y are *semantically connected* or *semantically similar*.

We use the following operators for expressing queries about *semantic associations*.

Definition 4 (ρ -Query) A ρ -Query, expressed as $\rho(x, y)$, where x and y are entities, results in the set of all semantic paths that connect x and y .

Definition 5 (σ -Query) A σ -Query, expressed as $\sigma(x, y)$, where x and y are entities, results in the set of all pairs of semantically similar paths originating at x and y .

We are currently working on a ranking technique for similarity associations, which is not discussed in this paper. Furthermore, it is conceptually different than ranking semantic connections because it involves ranking the set of all pairs of semantically similar paths originating at entities x and y . Thus semantic associations and semantic association queries are used to refer to only semantic connectivity and ρ -Queries respectively in the rest of the paper.

1.2 Ranking Semantic Relations

A typical semantic query can result in many semantic paths semantically linking the entities of interest. Because of the expected high number of paths, it is likely that many of them would be regarded as irrelevant with respect to the user's domain of interest. Thus, the semantic associations need to be filtered according to their perceived relevance. Also, a customizable criterion needs to be imposed upon the paths representing semantic associations to focus only on relevant associations. Additionally, the user should be presented with a ranked list of resulting paths to enable a more efficient analysis. The issues of filtering and ranking raise some interesting and challenging scientific problems.

To determine the relevance of semantic associations it is necessary to capture the context within which they are going to be interpreted and used (or the domains of the user interest). For example, consider a sub-graph of an RDF graph representing two soccer players who belong to the same team and who also started a new restaurant together. If the user is just interested in the *sports* domain the semantic associations involving restaurant related information can be regarded as irrelevant (or ranked lower). This can be accomplished by enabling a user to browse the ontology and mark a region (sub-graph) of nodes and/or properties of interest. If the discovery process finds some associations passing through these regions then they are considered relevant, while other associations are ranked lower or discarded. More formally, *ontological regions* can represent context. In this paper we present a flexible method for specifying context through an ontology-based context specification language.

Ranking of semantic associations effectively requires more than using the “ontological context” for relevance determination. The ranking process needs to take into consideration a number of criteria which can distinguish among associations which are perceived as more and less meaningful, more and less distant, more and less trusted etc. In this paper, the ranking score assigned to a particular semantic association is defined as a function of these parameters. Furthermore different weights can be given to different parameters according to users’ preferences (e.g., trust could be given more weight than others). This is a new and different problem than ranking documents using traditional search engines where documents are usually ranked according to the number of (sometimes subject-specific) references to them.

Thus our contributions in this paper are two-folds:

- Capturing users’ interests semantically through an ontology-based context specification language,
- Using a ranking function incorporating user-defined semantics (e.g., context) and universal semantics (e.g., associations conveying more information).

The rest of the paper is organized as follows: Section 2 presents related work. Section 3 introduces context specification language and discusses ranking technique. Section 4 concludes the paper.

2 Related Work

Knowledge representation approaches tried to capture relationships based on logics, or sets theory, etc. Our approach is to consider relations in the semantic Web, those that are expressed semantically using the RDF model. Then from a set of semantic associations we try to distinguish the relevant ones quantitatively. Research in the area of ranking semantic relations includes [12], where the notion of “semantic ranking” is presented to rank queries returned within semantic Web portals. Their technique reinterprets query results as “query knowledge-bases”, whose similarity to the original knowledge-base provides the basis for ranking. The actual similarity between a query result and the original knowledge-base is derived from the number of similar super classes of the result and the original knowledge-base. In our approach, the relevancy of results usually depends on a context defined by users.

Our earlier work [2] introduces using “context”, path length, and property relevance as a basis for ranking. Basically, [2] defines a notion of context which includes a set of ontologies and a set of relationship name pairs with a value. The value indicates the precedence level, a degree of importance for a particular context. This approach considers context based on value assignments for different ontologies. In this work instead, we provide context specification at a level (of classes and properties) that allows precise definitions of areas of interest for the user.

While the issues of ranking semantic relations are fundamentally different from those addressed in contemporary information retrieval ranking approaches, it is worth discussing some of these techniques. [5] presents the *page rank* algorithm used by Google. Page rank weights are assigned on the basis of page references, thus more popular pages have a higher rank. [21] presents Teoma’s technique of *subject specific popularity*, in which a page’s rank is based on the number of same-subject pages that reference it, not just its general popularity. Earlier, Northern Light had introduced the concept of folders and the documents resulting from keyword search results were segregated by these folders representing relevant categories. While relevant, these ranking algorithms lack the consideration of formal semantics (as captured through ontology representation) and explicitly specified context when assigning ranks, both of which are needed when ranking semantic associations. Although the current semantic association ranking scheme differs from ranking Web pages through not involving social contributors such as a *voting* mechanism, it is an interesting research direction to involve similar techniques for assessing importance and value of semantic associations.

Attempts to model context include [9], which proposed a context representation mechanism to solve conflicts of semantic and schematic similarities between database objects. [6] introduced an ontology that captures users’ context and situation by considering goals, tasks, actions and system’s context in order to observe and model human activities. The approach is mainly focused to use context to reduce user’s intervention in the system.

3 Ranking Semantic Associations

In this work, we provide semantic associations which are ranked for a given semantic association query. Our approach for ranking semantic associations is primarily based on capturing the interests of a user. Therefore, a context specification is the first step towards measuring how relevant a semantic association is.

3.1 Context Specification

A context specification captures the users’ interest in order to provide her with the relevant knowledge within numerous indirect relationships between the entities. We consider data in an RDF model with an associated RDF Schema [4] that describes the relationships between entities. Since the types of the entities are described in the RDF Schema, we can use the associated class and relationship types to restrict our attention

to the entities and relations of interest. Thus, by defining regions (or sub-graphs) of the RDF Schema (RDFS) we are capturing the areas of interest of the user. Particularly important for us is the ability to define that the path of interest (semantic association) should include properties and/or classes of interest for the user. A *region* of interest is a subset of classes (entities) and properties of a schema.

The detail to which a region of interest can be specified may vary for different applications. We have considered the following cases: (i) Class level: paths that include instances of that class are relevant, and (ii) Property level: paths including the specified properties are relevant.

Within the Class level, we may also restrict or allow subclasses to be considered relevant as well as the classes higher in the class hierarchy. For example, an “*Organization*” class may be considered relevant together with subclasses “*PoliticalOrganization*”, “*FinancialOrganization*” and “*TerroristOrganization*”, but a class “*Account*” that is parent of the class “*CorporateAccount*” may not be of importance.

At a Property level, we can specify restrictions similar to those of the Class level. An interesting and powerful context restriction that can be specified in properties is indication of which classes the property can be applied to (“domain” in RDFS) as well as which classes a property points to (“range” in RDFS). An example is a property “*involvedIn*” with a domain “*Organization*” and range “*Event*” (that is, *Organization* \rightarrow *involvedIn* \rightarrow *Event*). Our context specification allows restriction of the type of classes for domain and/or range. For example, it is possible to indicate that the property “*involvedIn*” is relevant when the entity that it is applied to is of class “*TerroristOrganization*” (a subclass of “*Organization*”).

We specify in a flexible yet detailed manner which Classes and Properties are relevant using XML. The following is an example of specifying Classes with restrictions:

```
<region id="R1" weight=".65">
  <classLevel name="TerroristAct" includeSubclasses="all"/>
  <classLevel name="TerroristOrg" includeSubclasses="no"/>
  <propertyLevel name="involvedIn" domainRestrictions="TerroristOrg"
rangeRestrictions="TerroristAct, Kidnapping, SuicideAttack" />
</region>
```

A region has a weight defining its relative importance. The particular XML example shown above captures the area of interest that is used as *region A* in **Fig. 5** in Section 3.2.2. Note that a user can define several ontological regions with different weights to specify the association types s/he is interested in.

3.2 Weight Assignments

Semantic associations represented as paths connecting two entities can span across multiple domains (or regions) and involve any number of entities and properties. Our ranking approach defines a path rank as a function of various intermediate weights.

As a path is traversed it will have many different intermediate weights which ultimately contribute to its overall rank. We classify these weights into two categories, *Universal* and *User-Defined*.

3.2.1 Universal Weights

Certain weights will influence a path rank regardless of the query or context of interest. We call them *Universal Weights*. The following subsections identify and define *Universal Weights* that contribute to the overall path rank.

Subsumption Weight. When considering entities in ontology, those that are lower in the hierarchy can be considered to be more specialized instances of those further up in the hierarchy [16]. Thus, lower entities have more specific meaning. **Fig. 1** depicts a class, “*Organization*”, as well as various subclasses of it. In the figure, “*Organization*” is the highest class in the hierarchy, and thus is the most general. It is clear that a “*Political Organization*” is a more defined “*Organization*”.

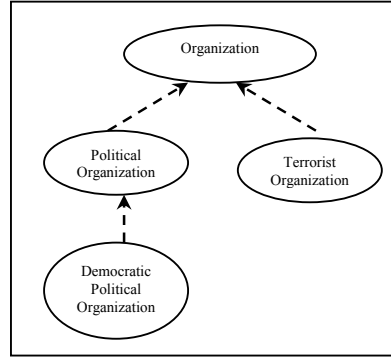


Fig. 1. Class Hierarchy Example

Similarly, a “*Democratic Political Organization*” conveys more meaning than both an “*Organization*” and a “*Political Organization*”. Hence, it is very apparent that as the hierarchy is traversed from the top down, subclasses become more specialized than their super-classes. The concept of class specialization in a path is captured by a Universal Weight that we call a *Subsumption Weight*. The intuition is assigning more weights to more “specific” semantic associations because they convey more meaning than “general” associations.

We will now provide some brief definitions used to define the overall *Subsumption Weight* of a path. First, we define a component, c , within a path P to be any entity or property contained in P . Thus, $c = \{entity\} \cup \{property\}$.

Next we define a *component weight* of the i^{th} component c_i , in a path P such that

$$Component\ Weight_i = \frac{H_{c_i}}{|H|} \quad (1)$$

where H_{c_i} is the position of the i^{th} component in its hierarchy H (the class at the top has value 1) and $|H|$ is the total height of the classes/properties hierarchy. Hence, $Component\ Weight_i \rightarrow (0,1]$. For example, given **Fig. 1** above, the component weight of the classes *Democratic Political Organization*, c_3 , and *Political Organization*, c_2 , would be

$$c_3 = \frac{H_{c_3}}{|H|} = \frac{3}{3} = 1 \text{ and } c_2 = \frac{H_{c_2}}{|H|} = \frac{2}{3} = 0.6. \quad (2)$$

We can now define the overall *Subsumption Weight* of a path P such that

$$S_P = \frac{1}{|c|} \times \prod_{i=2}^{|c|+1} c_i. \quad (3)$$

where $|c|$ is the number of components in P (excluding the start and end entities because they will never change in a result set) and c_i is the *component weight* of the i^{th} component in the path. Thus the *Subsumption Weight* of a path P , S_P , is the product of all the component weights within P , normalized by the number of components in the path (to avoid bias in path length). To illustrate this, we use the ontology that has been developed for the national security domain in our lab (see **Fig. 2**).

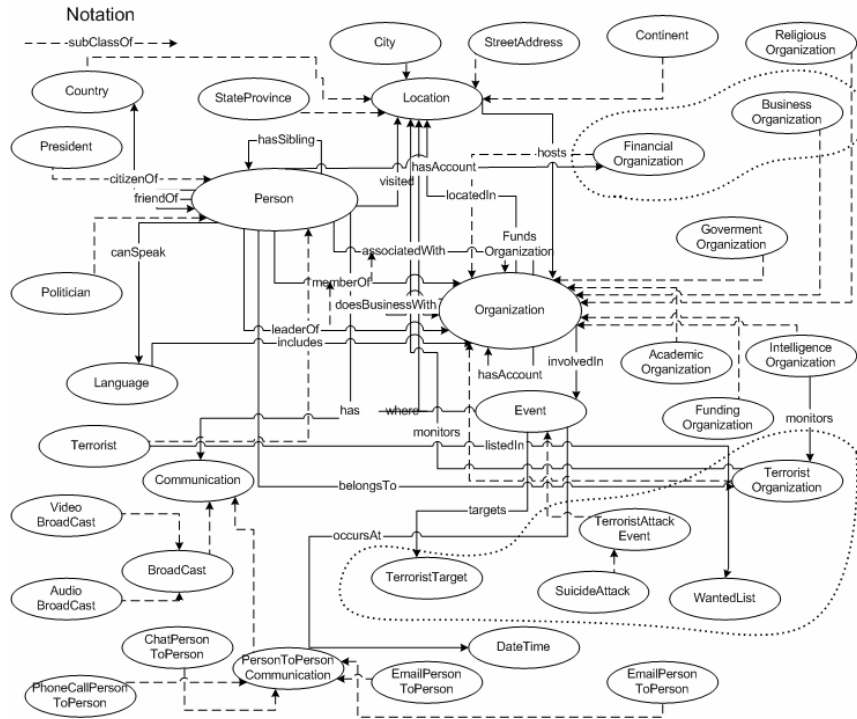


Fig. 2. Sample Ontology

Consider the following paths between entities e_1 and e_5 depicted in **Fig. 3**. First, one can see that all three paths are somewhat similar. The middle path seems to be a bit more specific than the top path, in that the person is member of a “*Terrorist Organization*,” not just any “*Organization*,” that is “*involvedIn*” a “*Suicide Attack*”. When

inspecting the bottom path we see that this person is actually a “*leaderOf*” some “*Terrorist Organization*” that was “*involvedIn*” the same “*Suicide Attack*”. Thus we assume that the third path conveys more meaning than the first two. When ranking these three paths with respect to their total meaning conveyed, one would expect to see that last path ranked higher than the others (in absence of additional user defined context/weights).

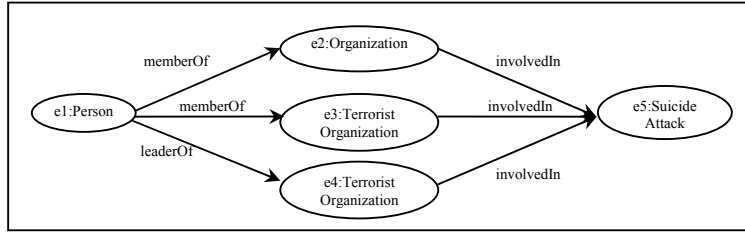


Fig. 3. Subsumption Weight Example

Now we will determine the *Subsumption Weight*, S_1 , of the first path in **Fig. 3**, $e_1 \rightarrow e_2 \rightarrow e_5$. The corresponding *Subsumption Weight* for this path would be given by

$$S_1 = \frac{1}{3} \times \left(\frac{1}{2} \times \frac{1}{2} \times \frac{1}{1} \right) = .083 . \quad (4)$$

Similarly, the middle path $e_1 \rightarrow e_3 \rightarrow e_5$ has a *Subsumption Weight* of .167 and a higher value of .334 for the path $e_1 \rightarrow e_4 \rightarrow e_5$.

Hence as desired previously, with respect to only the meaning conveyed in the path, the *Subsumption Weight* will assign higher weights to paths with a more defined meaning. Thus, quality and completeness of the ontology become important to avoid biased ranking ([16] addresses issues on explicitness and formalization of ontologies). Note that we are considering specificity of relations besides entities. This is why the third semantic association is ranked higher than the second one. Furthermore, statistical properties of ontology (e.g., connectivity of certain nodes, etc.) can contribute to *Universal Weight* yet discussion of those metrics is out of scope of this paper.

3.2.2 User-Defined Weights

In contrast to *Universal Path Weights*, some path weights will be query (or context) specific. These will be referred to as *User-Defined Weights*. The following subsections identify and define *User-Defined Weights* that contribute to the overall path rank.

Path Length Weight. In some queries, a user may be interested in the most direct paths (i.e., the shortest path). This may infer a stronger relationship between two entities. Yet in other cases a user may wish to find possibly hidden, indirect, or discrete paths (i.e., longer paths). The latter may be more significant in domains where there may be deliberate attempts to hide relationships; for example, potential terrorist cells remain distant and avoid direct contact with one another in order to defer possible detection [10] or money laundering [1] involves deliberate innocuous looking transactions. Hence, the user should determine which *Path Length* influence, if any, should be used (largely domain dependent).

We will now define the *Path Length Weight*, L , of a path P , where $L_P \rightarrow [0, 1]$. If a user wants to favor shorter paths, (5a) is used, where $|c|$ is the number of components in the path P (excluding the first and last nodes). In contrast, if a user wants to favor longer paths (5b) is used.

$$L_P = \frac{1}{|c|} \text{ (a); } L_P = 1 - \frac{1}{|c|} \text{ (b).} \quad (5)$$

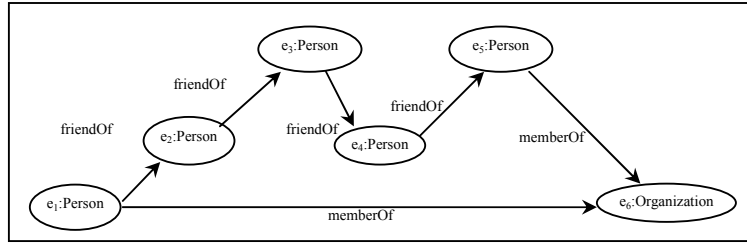


Fig. 4. Path Length Examples

To demonstrate the *Path Length Weight*, consider **Fig. 4**. This figure depicts two possible paths between a person and an organization. Given this example, suppose a user is interested in more direct path between entities. In this case, the longer of the two paths (call it P_1) should be ranked lower than the shorter one (P_2), so (5a) should be used.

Using (5a), the *Path Length Weight* of the two paths would be

$$L_{P_1} = \frac{1}{9}, \text{ where as } L_{P_2} = \frac{1}{1}. \quad (6)$$

Thus the shorter of the two paths has a higher rank value as initially expected. If a user were alternatively interested in longer paths, (5b) would be used instead. In this case

$$L_{P_1} = 1 - \frac{1}{9} = .889, \text{ where as } L_{P_2} = 1 - \frac{1}{1} = 0. \quad (7)$$

Thus, P_1 has a higher *Path Length Weight* than P_2 , again as desired.

Context Weight. As discussed in Section 3.1, it is possible to capture a user’s interest through a *Context Specification*. Thus, using the *context* specified, it is possible to rank a path according to its relevance with a user’s domain of interest.

With the Context Specification proposed in Section 3.1, a user can assign a weight to particular regions of ontology. When considering how to use these weights many issues arise. For example, paths can pass through numerous regions of interest. Large and/or small portions of paths can pass through these regions as well. Another consideration is whether all of the nodes in a path actually lie within a specified region. While we could omit paths that contain some nodes outside of all regions, we have decided to rank them lower because they are still considered relevant since they pass through some region. Suppose a user specifies the following region *A* containing the class “*TerroristAct*” and its subclasses and region *B* containing the class “*FinancialOrganization*” and its subclasses. The resulting regions, *A* and *B*, are within the terrorist and financial domains respectively. **Fig. 5** illustrates various paths which pass through these regions.

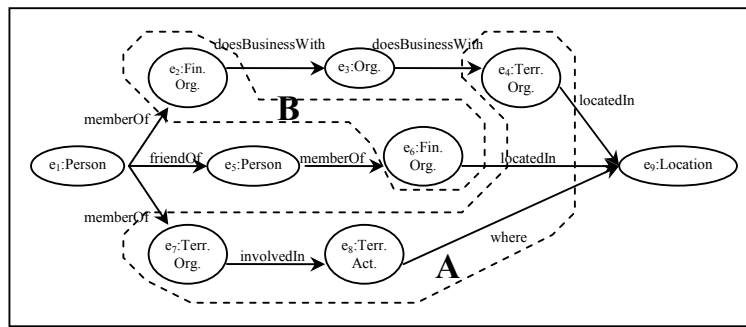


Fig. 5. Context Related Paths

The topmost path (call it P_1) passes through regions *B* and *A*, the middle path (P_2) passes through *region B*, and the third path (P_3) at the bottom passes through *region A*. Next, let the (user-defined) weight associated with a *region x* be represented as r_x . Also assume that $r_A = .75$ and $r_B = .50$.

The weight assignment illustrates the user is more interested in terrorism domain but also wants to consider financial associations, albeit with lesser priority. If we take into consideration the components of a path, excluding its start and end entities, the expected ranking of these three paths would be P_3, P_1, P_2 . Path P_3 would have the highest rank because all of its components (entities and properties) are included in some context, which happens to be the context with the highest weight. P_1 would be ranked next because it has a component in *B*, but (unlike P_2) also has a component in *A*. Given this background we will define the *Context Weight* of a path. First, let the i^{th} *region* be represented by R_i . Thus, we define the *Context Weight* of a given path P , C_P , such that

$$C_p = \frac{1}{|c|} \left(\left(\sum_{i=1}^{\#regionsPisIn} (r_i \times (\sum_{c \in R_i} c)) \right) \times \left(1 - \frac{\#c \notin R}{|c|} \right) \right) . \quad (8)$$

where r_i is the user defined weight of the *region* R_i , c is a component in the path P (excluding the start and end entities), and $|c|$ is number of components in the path (again excluding the start and end entities). That is, for each context that P passes through, sum the total number of components in P that are in the *region* R_i and multiply it by the weight attributed to that *region*, r_i . In order to reward paths in which all components are included in some region, the total number of components not in any region is divided by the total number of components, which is then subtracted from 1. This is then multiplied by the previous summation. Lastly, this total is normalized by the total number of components in the path. Note that a property component is considered to be in some *region* if it is entirely included in that *region* or one of the entities it is involved with (at either end) is in that *region*. If the two entities in which some property is involved are contained in two separate *regions*, the higher of the two *region* weights will be the *region* weight for that property. Also note that due to the subclass relationship of entities, properties which do not directly appear in a *region* may actually be included in some situations. To illustrate this, we will assign a *Context Weight* to the three paths presented **Fig. 5**.

P_1 passes through both regions A and B , which have a weight of .75 and .50 respectively. In both of these regions, three components are involved. Thus the initial summation is $(0.75 \times 3) + (0.5 \times 3) = 3.75$. There is one component (*Organization*) in P_1 which is not included in a region, so we have

$$3.75 \times \left(1 - \frac{1}{7} \right) = 3.21 . \quad (9)$$

This is normalized by the number of components in P_1 , hence we have

$$C_{P_1} = \frac{1}{7} \times 3.21 = .458 . \quad (10)$$

Next consider P_2 . This path only passes through region B , which has a weight .50. In this region, three components are involved. Thus the initial summation is $(0.50 \times 3) = 1.5$. There are two components ("*friendOf*" and "*Person*") in P_2 which are not included in a region, so we have

$$1.5 \times \left(1 - \frac{2}{5} \right) = .9 . \quad (11)$$

This is normalized by the number of components in P_2 , so

$$C_{P_2} = \frac{1}{5} \times .9 = .18 . \quad (12)$$

Lastly, consider P_3 . This path only passes through region A , which has a weight .75. In this region, five components are involved. Thus the initial summation is (0.75×5)

= 3.75. There are no components in P_3 which are not included in some *region*, so we have

$$3.75 \times \left(1 - \frac{0}{2}\right) = 3.75 . \quad (13)$$

This is normalized by the number of components in P_3 , so

$$C_{P_3} = \frac{1}{5} \times 3.75 = .75 . \quad (14)$$

Hence, as expected initially the ranking is P_3 (0.75), P_I (0.458), and P_I (0.18).

Trust Weight. Various relationships (properties) in a path originate from different sources. Some of these sources may be trusted while others may not (e.g., Reuters could be regarded as a more trusted source on international news than some of the other news organizations). Thus, trust values need to be assigned to relationships depending on the source. The process of automatically assigning trust to a specific relationship is out of the scope of this paper; instead we assume that users or other processes previously specified the trust value of relationships. Let the trust weight of the i^{th} property p_i of a path be t_{p_i} , where $t_{p_i} \rightarrow [0,1]$. We now define the Trust Weight of an overall path P as

$$T_P = \prod_{i=1}^{\#p \in c_P} t_{p_i} . \quad (15)$$

where c_P are all the *property* components within the path P . Thus, T_P is the product of all property weights in the P .

3.3 Ranking Criterion

Section 3.2, defines various path weight influences. We will now define the overall path rank, using these weights. As mentioned earlier, *Universal Weights* will always affect the overall path weight, while the *User-Defined Weights* will only be used when specified by the user. Let the Overall *Path Weight* of a path P denoting a semantic association be a linear function such that

$$W_P = k_1 \times S_P + k_2 \times L_P + k_3 \times C_P + k_4 \times T_P . \quad (16)$$

where k_i add up to 1.0 and are intended to allow fine-tuning of the different ranking criteria (e.g., *trust* can be given more weight than *path length*).

3.4 Preliminary Results

As a test-bed for querying semantic associations we have implemented a prototype named PISTA (see **Fig. 6**). In PISTA (Passenger Identification, Screening, and Threat Analysis) we have designed an ontology for national security domain (see **Fig. 2**). This ontology has names of organizations, countries, people, terrorists, terrorist acts etc. that are all inter-related to each other with named relationships to reflect real-world knowledge about the domain (e.g., “terrorist” “belongs to” “terrorist organization”).

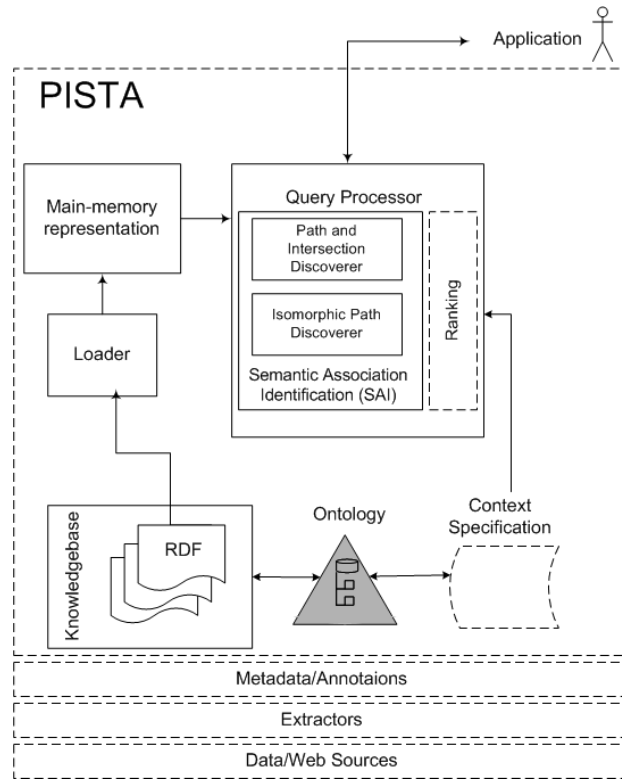


Fig. 6. PISTA Architecture

The sources from which metadata were extracted were selected to populate the ontology with entities related to terrorism. The metadata is represented in RDF, on which semantic association queries were performed. For information extraction we have used Semagix’s suite which includes a set of tools for extraction of entities from (semi)-structured information sources [17]. This toolkit allows extraction of entities from Web pages and establishes relationships between them. This extraction is based on our national security ontology thereby placing an extracted entity in its appropriate

place in the hierarchy of classes. Currently, there are over 6,000 entities and more than 11,000 explicit relations among them.

For querying semantic associations, we have implemented search algorithms, which use the schema information in conjunction with the RDF data that find semantic associations (Definition 3) that represent the relationships between any two entities. We represent both the RDF Schema and the RDF data as main memory directed graphs based on the Jena model [13]. Then, search for semantic similarity recursively finds similar paths between two entities by relying on the schema to find similar entities/relationships (i.e., which belong to same parent class) (see Definition 2). We also use a graph traversal algorithm (based on breadth-first search), which does not consider the direction of the edges when searching for semantic connectivity associations (see Definition 1).

For example, consider following semantic association query $\rho("Nasir Ali", "AlQeada")$. In PISTA this query results in 2234 associations. A small subset of these associations is shown in the table below (not in a particular order).

Nasir Ali \rightarrow friendWith \rightarrow T. Smith \rightarrow memberOf \rightarrow AlQeada
Nasir Ali \rightarrow friendWith \rightarrow Cabbar Ali \rightarrow visited \rightarrow Afganistan \rightarrow hosts \rightarrow AlQeada
Nasir Ali \rightarrow friendWith \rightarrow T. Smith \rightarrow hasAccount \rightarrow J. Funds \rightarrow fundsOrganization \rightarrow AlQeada
Nasir Ali \rightarrow friendWith \rightarrow OsamaBinLaden \rightarrow leaderOf \rightarrow AlQeada
Nasir Ali \rightarrow hasAccount \rightarrow J. Funds \rightarrow fundsOrganization \rightarrow AlQeada
Nasir Ali \rightarrow associatedWith \rightarrow A. G. College \rightarrow hasAccount \rightarrow J. Funds \rightarrow fundsOrganization \rightarrow AlQeada
Nasir Ali \rightarrow memberOf \rightarrow TRO \leftarrow memberOf \leftarrow OsamaBinLaden \rightarrow leaderOf \rightarrow AlQeada
Nasir Ali \rightarrow associatedWith \rightarrow TRO \rightarrow doesBusinessWith \rightarrow AlQeada

For illustration, we have a context defined by a region that captures ‘terrorism’ interest with weight of 0.6 (lower region in **Fig. 2**) and another region capturing ‘financial’ interest with weight of 0.4 (upper region in **Fig. 2**). The following table shows how the relationships are ranked when we apply our ranking formula. The ranking criteria (constants k_i in equation (16)) for this example assign values of 0.6 to *context weight*, 0.2 to *subsumption weight*, 0.1 to *path length weight* (longer paths favored), and 0.1 to *trust weight* (we assumed same trust for all entities/properties in this example).

Ranked Results	Rank
Nasir Ali \rightarrow memberOf \rightarrow TRO \leftarrow memberOf \leftarrow OsamaBinLaden \rightarrow leaderOf \rightarrow AlQeada	0.5560
Nasir Ali \rightarrow associatedWith \rightarrow TRO \rightarrow doesBusinessWith \rightarrow AlQeada	0.5488
Nasir Ali \rightarrow has Account \rightarrow J. Funds \rightarrow fundsOrganization \rightarrow AlQeada	0.5123
Nasir Ali \rightarrow friendWith \rightarrow T. Smith \rightarrow has Account \rightarrow J. Funds \rightarrow fundsOrganization \rightarrow AlQeada	0.3208
Nasir Ali \rightarrow associatedWith \rightarrow A. G. College \rightarrow has Account \rightarrow J. Funds \rightarrow fundsOrganization \rightarrow AlQeada	0.2941
Nasir Ali \rightarrow friendWith \rightarrow OsamaBinLaden \rightarrow leaderOf \rightarrow AlQeada	0.2733
Nasir Ali \rightarrow friendWith \rightarrow T. Smith \rightarrow memberOf \rightarrow AlQeada	0.2511
Nasir Ali \rightarrow friendWith \rightarrow Cabbar Ali \rightarrow visited \rightarrow Afganistan \rightarrow hosts \rightarrow AlQeada	0.2344

The top ranked semantic association comes up first because its entities all belong to the “terrorism” region (with higher relevance than “financial”) and it is one of the

longer associations. The second ranked semantic association includes entities only within the “terrorism” region as well, but it is a shorter path (longer paths are preferred in this example). The third association consists only of entities within the “financial” region, which we would expect to be ranked lower than the first two because we have weighted the “terrorism” region higher. The remaining paths contain some nodes not within any region, thus they are ranked below the previous three associations as expected. The fourth and fifth semantic associations are ranked as such because they are both longer than the rest and contain more entities within the two regions of interest. Note that the fourth association is ranked above the fifth because the “friendWith” relationship is more specific than the “associatedWith” relationship. When inspecting the last three associations, it is seen that they contain the least number of entities within a context. Thus, we would expect them to be ranked lower than the rest (due to the context being weighted so heavily). When we look at the sixth and seventh ranked associations, we see that the sixth is more specific in that entity “OsamaBinLaden” is the “leaderOf” “AlQaeda”, where the entity “T. Smith” is only a “memberOf” the same *Terrorist Organization*. The path ranked lowest contains the least number of entities in some region of interest, as expected.

4 Conclusions and Future Work

Semantic associations primarily capture information relating two entities. We are interested in the path that relates two entities by a sequence of interconnected links. Discovering of such relations (explained in [2]) gives results containing multiple paths connecting two entities. These paths have different meaning depending on the type of relation or the type of entities in each of components (either resource or property) of the path. The number of semantic associations between entities will grow much faster than the rate of the growth of a graph representing a knowledgebase and corresponding ontology. Also, understanding the relevance of each of the semantic association as a result of a query is arguably harder than determining a document’s relevance and ranking in a result provided by a typical search engine. Hence determining a good ranking strategy is crucial.

In this paper, we defined a ranking formula that considers *Subsumption Weight* (how much meaning a semantic association conveys depending on the places of its components in the ontology), *Path Length Weight* (that allows preference of either immediate or distant relationships), *Context Weight* (how relevant is the path to the user interest – defined using our context specification framework), and *Trust Weight* (determining how reliable a relationship is according to its provenance).

Currently we are working on ranking similarity associations (Definition 2). In fact this involves discovering all semantic connections between two entities (Definition 1), and then measuring if and how these associations can be broken into semantically symmetric associations (e.g., two terrorist attacks may be similar because they might be symmetrically connected to same methods). A formal query language for semantic associations is currently under development.

In order to assess the effectiveness of the ranking scheme outlined in this paper, standard ranking metrics such as precision and recall can be employed. However, we

think metrics for context-aware ranking should be different than the traditional metrics only using precision and recall. Because we rank the results considering a context specified by the user, and the evaluation criterion would be very subjective according to user's interests. Therefore, we believe a user-oriented assessment criterion is needed.

The future work also includes improving the semantic association discovery algorithms using the ranking scheme we have described in this paper for better scalability in very large data sets. For example, some partial paths can be pruned on the fly if their (partial) rank value drops under a predefined threshold.

Acknowledgements: We thank [Semagix, Inc.](#) for providing its Freedom product, which is based on the SCORE technology and related research out at the LSDIS Lab [20]. [PISTA](#) application has benefit from Semagix Inc.'s application in this area [14]. Brainstorming and discussions with members of our research project team, specifically Krysh Kochut, John Miller, Kemafor Anyanwu, and Cartic Ramakrishnan, have enriched this work.

References

- [1] Anti Money Laundering, Application White Paper, Semagix, Inc. http://www.semagix.com/pdf/anti_money_laundering.pdf
- [2] K. Anyanwu and A. Sheth, "[r-Queries: Enabling Querying for Semantic Associations on the Semantic Web](#)", The Twelfth International World Wide Web Conference, Budapest, Hungary (2003)
- [3] T. Berners-Lee, J. Hendler, and O.Lassila, "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities", Scientific American, May 2001
- [4] D. Brickley and R.V. Guha. Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation. March 2000
- [5] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Proc. 7th International World Wide Web Conference (1998)
- [6] J. L. Crowley, J. Coutaz, G. Rey and P. Reignier, "Perceptual Components for Context Aware Computing", UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002
- [7] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien, SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation, The Twelfth International World Wide Web Conference Budapest, Hungary (2003)
- [8] B. Hammond, A. Sheth, and K. Kochut, "[Semantic Enhancement Engine: A Modular Document Enhancement Platform for Semantic Applications over Heterogeneous Content](#)", in Real World Semantic Web Applications, V. Kashyap and L. Shklar, Eds., IOS Press, pp. 29-49, December 2002
- [9] V. Kashyap, A. Sheth. [Semantic and schematic similarities between database objects: a context-based approach](#). VLDB Journal (1996) 5: 276–304.

- [10] V. Krebs, "Mapping Networks of Terrorist Cells". *Connections*, 24(3): 43-52. (2002).
- [11] O. Lassila and R. R. Swick: "Resource Description Framework (RDF) Model and Syntax Specification", W3C Recommendation, World Wide Web Consortium, Cambridge (MA), February 1999
- [12] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. SE-mantic PortAL – The SEAL approach. to appear: In *Creating the Semantic Web*. D. Fensel, J. Hendler, H. Lieberman, W. Wahlster (eds.) MIT Press, MA, Cambridge (2001)
- [13] B. McBride "Jena: Implementing the RDF Model and Syntax Specification", in: Steffen Staab et al (eds.): "Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001", May 2001.
- [14] National Security and Intelligence, A Semagix White Paper, 2003. http://www.semagix.com/pdf/national_security.pdf
- [15] Ontoprise® GmbH, <http://www.ontoprise.com>
- [16] M. Rodriguez, and M. Egenhofer, "Determining Semantic Similarity among Entity Classes from Different Ontologies", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 2, March/April 2003.
- [17] Semagix. <http://www.semagix.com>.
- [18] U. Shah, T. Finin, A. Joshi, R. S. Cost, and J. Mayfield, "Information Retrieval on the Semantic Web", 10th International Conference on Information and Knowledge Management, November 2002.
- [19] A. Sheth, I. B. Arpinar, and V. Kashyap, "[Relationships at the Heart of Semantic Web: Modeling, Discovering, and Exploiting Complex Semantic Relationships.](#)" Enhancing the Power of the Internet Studies in Fuzziness and Soft Computing, M. Nikravesh, B. Azvin, R. Yager and L. Zadeh, Springer-Verlag, 2003 (in print).
- [20] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, Y. Warke, [Semantic Content Management for Enterprises and the Web](#), *IEEE Internet Computing*, July/August 2002, pp. 80-87.
- [21] Teoma: <http://sp.teoma.com/docs/teoma/about/searchwithauthority.html>

I know what you mean: semantic issues in Internet-scale publish/subscribe systems*

Ioana Burcea, Milenko Petrovic, and Hans-Arno Jacobsen

Department of Electrical and Computer Engineering
Department of Computer Science
University of Toronto, Canada
{ioana,petrovi,jacobsen}@eecg.toronto.edu

Abstract. In recent years, the amount of information on the Internet has increased exponentially developing great interest in selective information dissemination systems. The publish/subscribe paradigm is particularly suited for designing systems for routing information and requests according to their content throughout wide-area network of brokers. Current publish/subscribe systems use limited syntax content-based routing. Since publishers and subscribers are anonymous and decoupled in time, space and location, often over wide-area network boundaries, they do not necessarily speak the same language or use the same data and language format. Consequently, adding semantics to current publish/subscribe systems is important. In this paper we identify and examine the issues in developing semantic-aware content-based routing for publish/subscribe broker networks.

1 Introduction

The increase in the amount of data on the Internet has led to the development of a new generation of applications based on selective information dissemination where data is distributed only to interested clients. Such applications require a new middleware architecture that can efficiently match user interests with available information. Middleware that can satisfy this requirement include event-based architectures such as publish/subscribe systems.

In publish/subscribe systems (hereafter referred to as pub/sub systems), clients are autonomous components that exchange information by publishing events and by subscribing to events¹ they are interested in. In these systems, publishers produce information, while subscribers consume it. A component usually generates a message when it wants the external world to know that a certain event has occurred. All components that have previously expressed their interest in receiving such events will be notified about it. The central component of this architecture is the event dispatcher (also known as event broker). This component records all subscriptions in the system. When a certain event is published,

* First International Workshop on Semantic Web and Databases, Berlin 2003

¹ We use the terms *event* and *publication* interchangeably.

the event dispatcher matches it against all subscriptions in the system. When the incoming event verifies a subscription, the event dispatcher sends a notification to the corresponding subscriber.

The earliest pub/sub systems were topic-based. In these systems, each message (event) belongs to a certain topic. Thus, subscribers express their interest in a particular subject and they receive all the events published within that particular subject. The most significant restriction of these systems is the limited selectivity of subscriptions. The latest systems are called content-based systems. In these systems, the subscriptions can contain complex queries on event content.

Pub/sub systems try to solve the problem of selective information dissemination. Recently, there has been a lot of research on solving the problem of efficiently matching events against subscriptions. The proposed solutions are either centralized, where a single broker stores all subscriptions and event matching is done locally [1, 7, 8], or distributed, where many brokers need to collaborate to match events with subscriptions because not all subscriptions are available to every broker [3, 5]. The latter approach is also referred to as content-based routing because brokers form a network where events are routed to interested subscribers based on their content.

The existing solutions are limited because the matching (routing) is based on the syntax and not on the semantics of the information exchanged. For example, someone interested in buying a car with a “value” of up to 10,000 will not receive notifications about “vehicles,” “automobiles” or even “cars” with “price” of 8,999 because the system has neither understanding of the “price”-“value” relationship, nor of the “car”-“automobile”-“vehicle” relationship.

In this paper we examine the issues in extending distributed pub/sub systems to offer semantic capabilities. This is an important aspect to be studied as components in a pub/sub systems are decoupled, apriori anonymous, often widely distributed and do not necessary speak the same language.

2 Related work

We are not aware of any previous work addressing the semantic routing problem in pub/sub systems. Most research on semantic has been done in the area of heterogeneous database integration [4, 11, 16]. The issues addressed in this area refer to enabling integration of heterogeneous information systems so that users can access multiple data sources in an uniform manner. One way of solving this problem is by using ontologies. Semantic information systems use an ontology to represent domain-specific knowledge and allow users to use the ontology terms to construct queries. The query execution engine accesses the ontology either directly or via an inference engine in order to optimize the query and generate an execution plan. Use of an ontology to generate an execution plan is central in determining the right source database and method for retrieving the required information. This allows uniform access to multiple heterogeneous information sources. The problem of adding semantic capability to pub/sub systems can be seen as an “inverse” problem to the heterogeneous database integration problem.

In semantic pub/sub systems, subscriptions are analogous to queries and events correspond to data, so now the problem is how to match data to queries.

Some systems [4, 2] use inference engines to discover semantic relationships between data from ontology representations. Inference engines usually have specialized languages for expressing queries different from the language used to retrieve data, therefore user queries have to be either expressed in, or translated into the language of the inference engine. The ontology is either global (i.e., domain independent) or domain-specific (i.e., only a single domain) ontology. Domain-specific ontologies are smaller and more commonly found than global ontologies because they are easier to specify. Additionally, there are systems that use mapping functions exclusively and do not operate with inference engines [11, 16]. In these systems, mapping functions serve the role of an inference engine.

Web service discovery is a process of matching user needs to provided services; user needs are analogous to events and provided services to subscriptions in a pub/sub system. Web service discovery systems [13, 17] are functionally similar to a pub/sub system. During a discovery process, a web service advertises its capabilities in terms of its inputs and outputs. An ontology provides an association between related inputs or outputs of different web services. A user looks for a particular web service by searching for appropriate inputs and outputs according to the user's needs. Relevant services are determined by either exact match of inputs and outputs, or a compatible match according to ontology relationships.

The main push for using ontologies and semantic information as means of creating a more sophisticated application collaboration mechanisms has been from the Semantic Web community². Recently their focus was on developing DAML+OIL—a language for expressing, storing and exchange of ontologies and query languages for DAML+OIL [9]. Our vision of a distributed semantic publish/subscribe system is similar to that of the semantic web. The issues of distributing ontological information and bridging of different ontologies are common to both.

A system for distributed collaboration [6] creates a virtual network of proxies (functionally similar to brokers) using IP multicast connecting both data producers and consumers (users). Using a common ontology, sources provide descriptions (metadata similar to subscriptions and events) of multimedia data they are providing and users provide their capabilities. The metadata is distributed among proxies to create a *semantic multicast graph* along which data is distributed to interested users.

To improve scalability, peer-to-peer systems are looking in the direction of semantic routing. HyperCuP [15] uses a common ontology to dynamically cluster peers based on the data they contain. A cluster is identified using a more general concept than any of its members in the ontology. Ontology concepts map to cluster addresses so a node can determine appropriate routes for a query by looking up more general concepts of the query terms in the concept hierarchy.

² www.semanticweb.org

Edutella [12] uses query hubs (functionally similar to brokers) to collect user metadata and present the peer-to-peer network as a virtual database, which users query. All queries are routed through a query hub, which forwards queries to only those nodes that can answer them.

3 Local Matching and Content-based Routing

Due to space limitation, we will not provide an extensive background about pub/sub systems and content-based routing. Instead, we briefly present the most important concepts that help the reader understand the ideas conceived in this paper.

The key point in pub/sub systems is that the information sent into the system by the publisher does not contain the addresses of the receivers. The information is forwarded to interested clients based on the content of the message and clients subscriptions [5]. In a centralized approach, there is only one broker that stores all subscriptions. Upon receiving an event, the broker uses a matching algorithm to match the event against the subscriptions in order to decide which subscribers want to receive notifications about the event [1, 8].

Usually, publications are expressed as lists of attribute-value pairs. The formal representation of a publication is given by the following expression: $\{(a_1, val_1), (a_2, val_2), \dots, (a_n, val_n)\}$. Subscriptions are expressed as conjunctions of simple predicates. In a formal description, a simple predicate is represented as *(attribute_name relational_operator value)*. A predicate *(a rel_op val)* is matched by an attribute-value pair *(a, val)* if and only if the attribute names are identical $(a = a)$ and the *(a rel_op val)* boolean relation is true. A subscription *s* is matched by a publication *p* if and only if all its predicates are matched by some pair in *p*. In this case we say that the subscription is matched at syntactic level.

The distributed approach involves a network of brokers that collaborate in order to route the information in the system based on its content [3, 5]. In this case, practically, each broker is aware of its neighbours interests. Upon receiving an event, the broker matches it against its neighbours subscriptions and sends the event only to the interested neighbours. Usually, the routing scheme presents two distinct aspects: subscription forwarding and event forwarding. Subscription forwarding is used to propagate clients interests in the system, while event forwarding algorithms decide how to disseminate the events to the interested clients. Two main optimizations were introduced in the literature in order to increase the performance of these forwarding algorithms: subscription covering and advertisements [3, 5].

Subscription covering

Given two subscriptions s_1 and s_2 , s_1 covers s_2 if and only if all the events that match s_2 also match s_1 . In other words, if we denote with E_1 and E_2 the set of events that match subscription s_1 and s_2 , respectively, then $E_2 \subseteq E_1$.

If we look at the predicate level, the covering relation can be expressed as follows: Given two subscriptions $s_1 = \{p_1^1, p_2^1, \dots, p_n^1\}$ and $s_2 = \{p_1^2, p_2^2, \dots, p_m^2\}$, s_1 covers s_2 if and only if $\forall p_k^1 \in s_1, \exists p_j^2 \in s_2$ (p_k^1 and p_j^2 refer to the same

Subscription s_1	Subscription s_2	Covering Relation
(product = "computer", brand = "IBM", price \leq 1600)	(product = "computer", brand = "IBM", price \leq 1500)	s_1 covers s_2
(product = "computer", brand = "IBM", price \leq 1600)	(product = "computer", price \leq 1600)	s_2 covers s_1
(product = "computer", brand = "IBM", price \leq 1600)	(product = "computer", brand = "Dell", price \leq 1500)	s_1 does not cover s_2 , s_2 does not cover s_1

Table 1. Examples of subscriptions and covering relations

attribute) such that if p_j^2 is matched by some attribute-value pair (a, val) , then p_k^1 is also matched by the same (a, val) attribute-value pair. In other words, s_2 has potentially more predicates and the common ones are more restrictive than those in s_1 (i.e., the domain of values that satisfy them is potentially smaller). Table 1 presents some examples of subscriptions and the corresponding covering relations.

When a broker B receives a subscription s , it will send it to its neighbours if and only if it has not previously sent them another subscription s' , that covers s . Broker B is ensured to receive all events that match s , since it receives all events that match s' and the events that match s are included in the set of the events that match s' .

Advertisements

Advertisements are used by publishers to announce the set of publications they are going to publish [3]. Advertisements look exactly like subscriptions³, but have a different role in the system: they are used to build the routing path from the publishers to the interested subscribers.

An advertisement a determines an event e if and only if all attribute-value pairs match some predicates in the advertisement. Formally, an advertisement $a = \{p_1^1, p_2^1, \dots, p_n^1\}$ determines an event e , if and only if $\forall (a, v) \in e, \exists p_k \in a$ such that (a, v) matches p_k .

An advertisement a intersects a subscription s if and only if the intersection of the set of the events determined by the advertisement a and the set of the events that match s is a non-empty set. Formally, at predicate level, an advertisement $a = \{a_1, a_2, \dots, a_n\}$ intersects a subscription $s = \{s_1, s_2, \dots, s_n\}$ if and only if $\forall s_k \in s, \exists a_j \in a$ and some attribute-value pair $(attr, val)$ ⁴ such that $(attr, val)$ matches both s_k and a_j . Table 2 presents some examples of subscriptions and advertisements and the corresponding intersection relations.

When using advertisements, upon receiving a subscription, each broker forwards it only to the neighbours that previously sent advertisements that intersect

³ However, there is an important distinction between the predicates in an advertisement and those in a subscription: the predicate in a subscription are considered to be in a conjunctive form, while those in an advertisement are considered to be in disjunctive form.

⁴ s_k and a_j refer to the same attribute $attr$

Subscription s	Advertisement a	Intersection Relation
(product = “computer”, brand = “IBM”, price \leq 1600)	(product = “computer”, brand = “IBM”, price \leq 1500)	a intersects s
(product = “computer”, price \leq 1600)	(product = “computer”, brand = “IBM”, price \leq 1600)	a intersects s
(product = “computer”, brand = “IBM”, price \leq 1600)	(product = “computer”, brand = “Dell”, price \leq 1500)	a does not intersect s

Table 2. Examples of subscriptions, advertisements and intersection relations

with the subscription. Thus, the subscriptions are forwarded only to the brokers that have potentially interesting publishers.

4 Towards Semantic-based Routing

In order to add a semantic dimension to distributed pub/sub systems, we have to understand how to adapt or map the core concepts and functionalities of existing solutions for content-based routing to the new context that involves semantic knowledge.

In this section we first introduce some extensions to the existing matching algorithms in order to make them semantic-aware and then we discuss the implications of using such a solution for semantic-based routing.

4.1 Semantic Matching

In this section we summarize our approach to make the existing centralized matching algorithms semantic-aware [14]. Our goal is to minimize the changes to the existing matching algorithms so that we can take advantage of their already efficient techniques and to make the processing of semantic information fast. We describe three approaches, each adding more extensive semantic capability to the matching algorithms.

The first approach allows a matching algorithm to match events and subscriptions that use semantically equivalent attributes or values—*synonyms*. The second approach uses additional knowledge about the relationships (beyond synonyms) between attributes and values to allow additional matches. More precisely, it uses a *concept hierarchy* that provides two kinds of relations: specialization and generalization. The third approach uses *mapping functions* which allow definitions of arbitrary relationships between the schema and the attribute values of the event.

The synonym step involves translating all strings with different names but with the same meaning to a “root” term. For example, “car” and “automobile” are synonyms for “vehicle” which then becomes the root term for the three words. This translation is performed for both subscriptions and events and at

both attribute and value level. This allows syntactically different events and subscriptions to match. This translation is simple and straightforward. The semantic capability it adds to the system, although important, may not be sufficient in some situations, as this approach does not consider the semantic relation between attributes and values. Moreover, this approach is limited to synonym relations only.

Taxonomies represent a way of organizing ontological knowledge using specialization and generalization relationships between different concepts. Intuitively, all the terms contained in such a taxonomy can be represented in a hierarchical structure, where more general terms are higher up in the hierarchy and are linked to more specialized terms situated lower in the hierarchy. This structure is called a “concept hierarchy. Usually, a concept hierarchy contains all terms within a specific domain, which includes both attributes and values.

Considering the observation that the subscriber should receive only information that it has precisely requested, we come up with the following two rules for matching based on a concept hierarchy: (1) the events that contain more specialized concepts have to match the subscriptions that contain more generalized terms of the same kind and (2) the events that contain more generalized terms than those used in the subscriptions do not match the subscriptions.

In order to better understand these rules, we look at the following examples. Suppose that we have in the system a subscription:

$$S : (book = StoneAge)AND(subject = reptiles).$$

When the event:

$$E : \{(encyclopedia, StoneAge), (subject, crocodiles)\}$$

is entering the system, it should match the subscription S , as the subscriber asked for more general information that the event provides (in other words, an *encyclopedia* is a special kind of *book* and *crocodiles* represent a special kind of *reptiles*). On the other hand, considering the subscription:

$$S : (encyclopedia = StoneAge)AND(subject = reptiles)$$

and the incoming event

$$E : \{(book, StoneAge), (subject, crocodiles)\},$$

the event E should not match the subscription S , as the book contained in the event may be a dictionary or a fiction book (as well as an encyclopedia). Note that, although the subscription S contains in its second predicate a value more specialized than that in the event, the first predicate of the subscription is not matched by the event, and therefore, the event does not match the subscription. The last rule prevents an eventual spamming of the subscribers with useless information.

Mapping functions can specify relationships that, otherwise, cannot be specified using a concept hierarchy or a synonym relationship. For example, they can be used to create a mapping between different ontologies. A mapping function is a many-to-many function that correlates one or more attribute-value pairs to one or more semantically related attribute-value pairs. It is possible to have many mapping functions for each attribute. We assume that mapping functions are specified by domain experts. In the future, we are going to investigate using

a fully-fledged inference engine as a more compact representation of mapping functions and the performance trade off this entails.

We illustrate the concept of mapping functions with an example. Let us say that there is a university professor X, who is interested in advising new PhD graduate students. In particular, he is only interested in students who have had 5 or more years of previous professional experience. Subsequently, he subscribes to the following:

$S : (university = Y)AND(degree = PhD)AND(professional\ experience > 4)$

Specifically, the professor X is looking for students applying to university Y in the PhD stream with 5 or more years of experience. For each new student applying to the university, a new event, which contains among others the information about previous work experience, is published into our system. Thus, an event for a student who had some work experience would look like

$E : \{(school, Y), (degree, PhD), (graduation\ date, 1990)\}$.

In addition, the system has access to the following mapping function:

$f_1 : (graduation\ date) \rightarrow professional\ experience.$

You can think of function f_1 implemented as a simple difference between to-days date and the date of students graduation and returning that difference as the value of *professional experience*. For the sake of the example, f_1 assumes that the student has been working since graduation. Finally, the result of f_1 is appended to event E and the matching algorithm matches E to professor Xs subscription S .

In addition, we can think about events and subscriptions as points or regions in a multidimensional space [10] where the distance between points determines a match between an event and a subscription. This way it is possible that an event matches a subscription even if some attribute/value pair of the event is more general than the corresponding predicate in the subscription as long as the distance between the event and the subscription, as determined by *all* their constituent attribute-value pairs and predicates, respectively, is within the defined matching range.

To summarize, the synonym stage translates the events and the subscriptions to a normalized form using the root terms, while the hierarchy and the mapping stages add new attribute-value pairs to the events. The new events are matched using existing matching algorithms against the subscriptions in the system. In conclusion, we say that e semantically matches s ⁵ if and only if the hierarchy and the mapping stages can produce an event $e = e \cup E$ ⁶ that matches s at syntactic level.

4.2 Semantic-based Routing

At first glance, it is apparent that existing algorithms for subscription and event forwarding can be used with a semantic-aware matching algorithm in order to

⁵ e and s are in their normalized form

⁶ E represents the set of attribute-value pairs that are added by the hierarchy and the mapping stages. Note that E can be an empty set.

achieve semantic-based routing. However, this approach is not straight forward. In this section we discuss some open issues that arise from using a semantic-aware matching algorithm in content-based routing.

Subscription covering

Although it is defined at syntax level, the covering relation, as presented in Section 3, can be used directly with the semantic matching approach, discussed above, without any loss of notifications. In other words, if s_1 covers s_2 and a certain broker B will forward only subscription s_1 to its neighbours, it will still receive both events that semantically match s_1 and s_2 . This happens because the relation between the set of events E_1 and E_2 that semantically match s_1 and s_2 , respectively, is preserved, i.e., $E_2 \subseteq E_1$. Truly, if e semantically matches s_2 , then the hierarchy and the mapping stages can produce an event e' that matches s_2 at syntactic level. If e' matches s_2 at syntactic level, then, according to the definition of covering relation, e' matches s_1 at syntactic level. Since e' is produced by adding semantic knowledge to e , this means that e semantically matches s_1 , i.e. $E_2 \subseteq E_1$. Thus, broker B is ensured to receive all events that semantically match s_2 , since it receives all events that semantically match s_1 and the events that semantically match s_2 are included in the set of the events that semantically match s_1 .

Although the syntactic covering relation can be used without loss of notifications, some redundant subscriptions may be forwarded into the network. This happens because the set of events E_1 and E_2 that semantically match s_1 and s_2 can be in the following relation $E_2 \subseteq E_1$ without necessarily s_1 covering s_2 at syntax level. In other words, although s_1 does not cover s_2 at syntactic level, it may cover it semantically speaking. For example, consider the following subscriptions: $s_1 = ((product = "printed\ material")AND(topic = "semantic\ web"))$ and $s_2 = ((product = "book")AND(topic = "semantic\ web"))$. In this case, all events that semantically match s_2 will also match s_1 as a *book* is a form of *printed material*; thus $E_2 \subseteq E_1$, but s_1 does not cover s_2 (at syntax level). Therefore, the covering relation needs to be extended to encapsulate semantic knowledge. One simple way of transforming the covering relation to be semantic-aware is to use the hierarchy approach. In this case, subscription s_1 will cover s_2 as the *printed material* term is a more general term than *book*.

Advertisements

While the covering relation can be directly used with the semantic matching algorithms, this is not the case for advertisements. As explained earlier in this paper, advertisements are used to establish the routing path from the publishers to the interested subscribers. How the events are routed in the system depends on the intersection relation between advertisements and subscription. Consider the following example: advertisement $a = ((product = "printedmaterial"), (price \geq 10))$ and subscription $s = ((product = "book"), (price \leq 20))$. Advertisement a does not intersect s at syntactic level because there is no predicate p in a and not any attribute-value pair $(attr, val)$ such that $(attr, val)$ matches both p and the following predicate $(product = "book")$ of subscription s . (v. Section 3. Thus, the subscription will not be forwarded towards the publisher that emitted

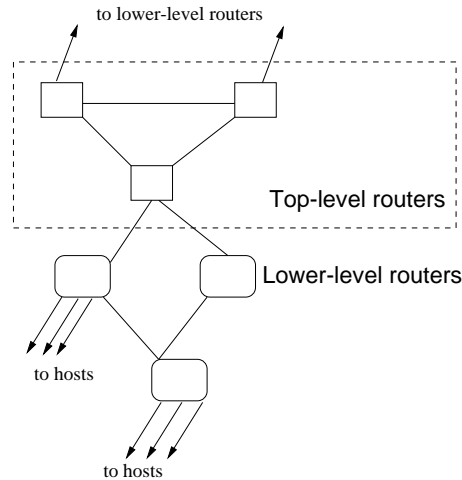


Fig. 1. Conceptual illustration of a two-level distributed semantic pub/sub network. Top-level routers have only high level descriptions of ontologies from the lower level routers.

the advertisement. All publications that will be produced by this publisher will not be forwarded to the subscriber, although some of them may matched its subscriptions.

Distributed semantic knowledge

The discussion above about subscription covering and advertisements considered that each broker contains the same semantic knowledge (i.e., same synonyms, hierarchies and mapping functions). However, the replication of the same semantic knowledge to all brokers in the system may not be feasible and it may be detrimental to scalability.

We envision a system where semantic knowledge is distributed between brokers⁷ in the same way that the Internet distributes link status information using routing protocols. A semantic knowledge database is equivalent to routing tables in terms of functionality.

The Internet is a hierarchical computer network. At the top of the hierarchy are relatively few routers containing very general information in routing tables. The tables do not contain information about every host on the Internet, but only about a few network destinations. Thus, high level pre-defined ontological information could be distributed in the same way among the top routers (Figure 1). It is difficult to envision what this higher level information will be at this time, but we only need to take a look at Internet directories such as Google and Yahoo to get an idea of top level semantic knowledge. Both of these directories provide a user with only a few key entries as starting point for exploring the vast Internet information store. We see top level brokers exchanging only covering and advertisement information.

⁷ We use the term *broker* and *router* interchangeably.

Lower in the Internet hierarchy routers maintain routing tables with destinations to specific hosts. Even though top level brokers use a common ontology, lower level brokers do not have to. For example, consider two different pairs of communicating applications: financial and medical. Financial applications are exchanging stock quotes, while medical are exchanging news about new drugs. These two application use different ontologies. The ontology information for each application can be distributed between multiple routers. These low level brokers will advertise more general descriptions of the ontologies they have to higher level brokers. Using this information, any new application will be able to locate the broker with specific ontologies. Any application wishing to integrate medical and financial information can create a mapping ontology between the financial and medical ontologies and provide a general description of the mapping ontology to higher level broker like in the previous case. We see that high level concepts can be used to route information between brokers who do not have access to specific ontologies. We can look at these general terms as very terse summaries of ontologies.

Our vision of a large scale semantic-based routing raises many questions:

- **top-level routing:** How to bridge multiple distributed ontologies to enable content routing? How can we avoid or reduce duplication of ontological information among brokers? What is an appropriate high level generalization that can bring together different ontologies? How do semantic routing protocols look like?
- **lower-level routing:** How to efficiently store ontological information at routers? Large knowledge databases will probably require secondary storage beyond what is available at routers. How does this affect routing? If routers have to use covering at this level how can they dynamically control the generality of covering to affect network performance?

5 Conclusions

In this paper we underline the limits of matching and content-based routing at syntactic level in pub/sub systems. We propose a solution for achieving semantic capabilities for local matching and look into the implications of using such a solution for content-based routing. We also present our vision on next-generation semantic-based routing. Our intent was to give rise to questions and ideas in order to improve existing content-based routing approaches and make them semantic-aware.

References

1. Marcos Kawazoe Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar Deepak Chandra. Matching events in a content-based subscription system. In *Symposium on Principles of Distributed Computing*, pages 53–61, 1999.

2. Y. Arens and C. A. Knoblock. Planning and reformulating queries for semantically-modeled multidatabase systems. In *Proceedings of the 1st International Conference on Information and Knowledge Management*, pages 99–101, 1992.
3. Antonio Carzaniga, David S. Rosenblum, and Alexander L Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.
4. Christine Collet, Michael N. Hubris, and Wei-Min Sheri. Resource integration using a large knowledge base in CARNOT. *IEEE Computer*, pages 55–62, December 1991.
5. G. Cugola, E. Di Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9), 2001.
6. Son Dao, Eddie Shek, Asha Vellaikal, Richard R. Muntz, Lixia Zhang, Miodrag Potkonjak, and Ouri Wolfson. Semantic multicast: intelligently sharing collaborative sessions. *ACM Computing Surveys*, 31(2es):Article No. 3, 1999.
7. Yanlei Diao, Peter Fischer, Michael Franklin, and Raymond To. Yfilter: Efficient and scalable filtering of XML documents. In *Proceedings of ICDE2002*, 2002.
8. Françoise Fabret, Hans-Arno Jacobsen, Françoise Llirbat, Joao Pereira, Ken Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proceedings of SIGMOD 2001*, 2001.
9. Ian Horrocks and Sergio Tessaris. Querying the semantic web: A formal approach. In *Proceedings of the Semantic Web - ISWC 2002: First International Semantic Web Conference*, Sardinia, Italy, 2002.
10. Hubert Leung and H.-Arno Jacobsen. Subject spaces: A state-persistent model for publish/subscribe systems. In *Computer Science Research Group Technical Report CRSG-459*, University of Toronto, September 2002.
11. E. Mena, A. Illarramendi, V. Kashyap, and A. P. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *International Journal on Distributed and Parallel Databases*, 8(2):223–271, April 2000.
12. W. Nejdl, B. Wolf, Ch. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr, and T. Risch. Edutella: A p2p networking infrastructure based on rdf. In *11th International World Wide Web Conference (WWW2002)*, 2002.
13. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *The 1st International Semantic Web Conference*, 2002.
14. Milenko Petrovic, Ioana Burcea, and H.-Arno Jacobsen. S-ToPSS - A Semantic Publish/Subscribe System. In *Very Large Databases (VLDB03)*, Berlin, Germany, September 2003.
15. Mario Schlosser, Michael Sintek, Stefan Decker, and Wolfgang Nejdl. A scalable and ontology-based p2p infrastructure for semantic web service. In *The Second IEEE International Conference on Peer-to-Peer Computing*, 2002.
16. Edward Sciore, Michael Siegel, and Arnon Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, 1994.
17. David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *International Semantic Web Working Symposium*, 2001.

A Context-Oriented RDF Database

Mohammad-Reza Tazari

Computer Graphics Center, Dept. Mobile Information Visualization
Fraunhoferstraße 5, 64283 Darmstadt, Germany
Saied.Tazari@zgdv.de

Abstract. The importance of contextual knowledge in knowledge management and organizational memory is shown in topical literature. Even in an initial visionary scenario for the Semantic Web, one can immediately encounter the contextual knowledge needed to realize the necessary services. Hence, it is not inappropriate to claim that context management is an integral service of the Semantic Web. After discussing the distributed nature of contextual knowledge, we define some requirements for a context-oriented database service and then introduce CORD as a service satisfying those requirements based on the Semantic Web technologies. Selected features of CORD that provide some contribution to the discussions within the Semantic Web research community, like embedded resources, query language, and definition of rules, are discussed in some detail.

1 Introduction

Most of the Semantic Web applications will be context-aware and personalized services. A superficial look at the visionary scenario for explaining some of the features of the Semantic Web in [2] shows the correctness of this claim. Already in the first two sentences of this scenario¹, the following contextual knowledge must be available to realize the service:

- User location (here, Pete's location)
- Setup of the location (to identify devices near Pete and services offered there)
- States of the resources (which of the sound-making devices are “on”, the phone “ringing”, the phone in “talk” state)
- Characteristics and capabilities of resources (which services can operate precisely those sound-making devices near Pete that are on and loud)

Although the term *context* has a common meaning in the Semantic Web community – see, for example, the definition by Tim Berners-Lee under <http://www.w3.org/2000/10/swap/doc/Glossary> – we are purposing here a special user-centric view of context. That is, by context we mean the user context in terms of personal, environmental, and

¹ “The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other local devices that had a volume control.” [2]

2 Mohammad-Reza Tazari

temporal conditions surrounding him or her. This is the situational view to the context as it is investigated in Mobile Computing and Ubiquitous/Pervasive Computing, too. The whole imaginary scenario in [2] is full of assumptions about the existence of such contextual knowledge.

As already stated in [5], knowledge about the user context is highly distributed. Except for the “current time” that in fact belongs to the user context in diverse forms², but has nothing to do with the distribution aspect of the contextual knowledge, most of the other parts of this knowledge can be classified as follows (see also figure 1):

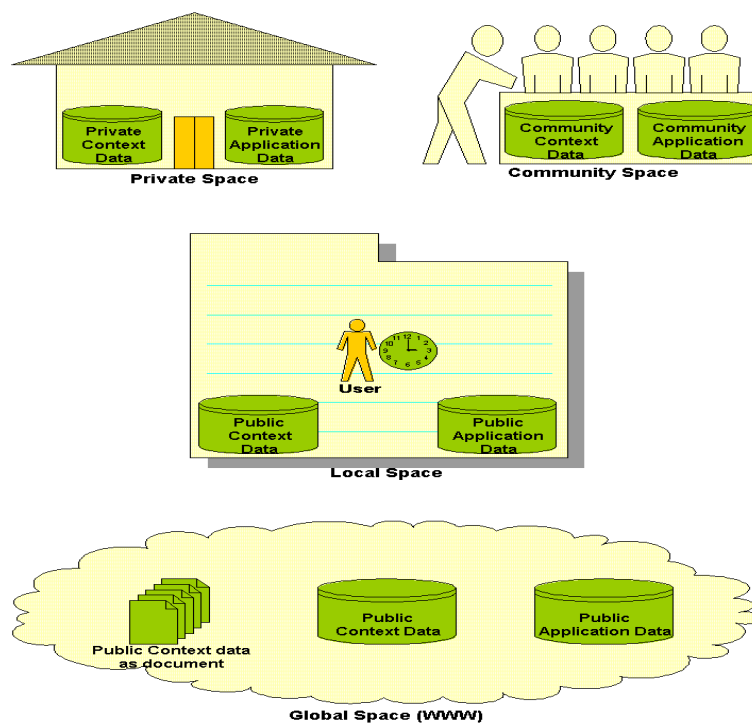


Fig. 1. Distribution of contextual knowledge. At any given time, the user finds himself at a specific location and may be able to use resources there. Nevertheless, he may also be able to use resources from his private space, community spaces to which he belongs, and public resources available globally

- Knowledge directly bound to the user and captured in diverse logical units, such as the user profile³, profiles of user's mobile and accompanying devices, application

² E.g. the absolute time, hour of the day, am / pm, day of the week, etc.

³ We believe that most parts of the user context may be imparted in form of profiles and define a *profile* as the storage unit for a coherent collection of key-value pairs describing a distinct resource, location, or user. If the described resource exists in an electronic form, then its profile provides the corresponding metadata; otherwise a profile may simultaneously serve as the electronic representation of the resource itself.

data from the domain of personal information management (i.e. PIM data, such as to-dos, appointments, contacts), application-specific user preferences⁴, etc.

- Knowledge bound to the user's location and captured as the location profile and profiles of resources available there, where the location profile serves as an integrating unit for all other info units.
- Knowledge bound to communities (to which the user belongs) and captured as group-based defaults, profiles of shared resources, and shared application data.
- Public knowledge independent of the user, communities, and locations that will be made available through the Web, such as profiles of public resources (e.g. services that can be utilized by all, independent of the locations of the two ends⁵) and profiles of classes of resources, which provide default values for a set of concrete resources.

Obviously, the contextual knowledge includes many shared units, such as the user profile, the location profile, and the profiles of several resources, of which different context-aware applications may make use. Hence, a standardized service is needed for managing profiles and offering shared mechanisms, relieving context-aware applications from certain common overheads like monitoring the user context and recognizing interesting situations. We call such a service the *context management service*. In [5], we discussed the requirements for a context management service and in [15], the aspect of modeling user context. Here, we focus on the data management aspects of this service.

1.1 Requirements for a Context-Oriented Database Service

In the discussion above, we have emphasized three specific points having to do with data management aspects of the context management service: data distribution, organizing data in profiles, and support for default values (group-based or class-based defaults). The first aspect leads us to the requirement that a context-oriented database service (CODBS) must overcome the problem with the distribution of contextual knowledge. Secondly, if profiles as a collection of key-value pairs are the storage units of a CODBS, then it must support arbitrarily structured keys and values⁶. Thirdly, support for default values would mean that there must be a mechanism for profiles of more concrete resources to inherit data from profiles of related, albeit more abstract, resources. All of the above actually reveal different aspects of data organization, namely data organization within a profile, between profiles, and between databases.

To specify further requirements, we must zoom in on the data organization within a profile. The organization of data within a profile will primarily be reflected in its keys. That is, a fundamental requirement is the possibility of expressing complex

⁴ Context-aware and personalized applications may have some personalization scheme that is specific to them and hence must be managed separately from the user profile that is a shared unit based on a shared ontology that models user profiles in general.

⁵ The locality of the resource may eventually play an important role in order for it to be selected / referenced / used from among all competing resources.

⁶ This requirement is refined further in the next paragraph.

structures via keys. Another aspect, however, has to do with the values. Values may be literal, which raises the question about support for data types, or references to other resources. A key may be associated with a single value or with more than one value. The latter case leads to the support for sequences, bags, and sets of alternative values. Values may be valid only for a specific time period⁷ or independent of time. Last but not least, they may be conditional/situational, meaning that a key may be associated with different alternative values for different situations.

Assuming that for each profile type there is a schema defining its structure and asserting some statements about its semantics, a CODBS must also use schemas in order to be able to ensure data integrity by accepting data that is in accordance with the structural and type-related assertions made in the schema. In addition, context-aware applications will be able to ask for the underlying schemas if they are not able to interpret some contextual knowledge.

Finally, a CODBS must provide a triggering mechanism for catching database events, because changes in the state of the contextual knowledge may influence the situation in which the user finds himself. The transition from one situation to another is an important event for context-aware applications.

Hence, we can summarize the requirements for a CODBS, as follows:

1. Managing profiles in accordance with their underlying schemas and guaranteeing data integrity based on the assertions made in the schemas
2. Providing a centralized view of the highly distributed contextual knowledge
3. Providing a triggering mechanism depending on complex situational DB events
4. Support for conditional values
5. Support for defining hierarchies of profiles that share the same schema to facilitate the automatic inheritance of default values
6. Support for expressing complex structures via keys within profiles
7. Support for literal values with different data types
8. Support for sequences, bags, and sets of alternative values
9. Support for temporary values [13]
10. Support for using references to other resources as values

2 CORD: The Context-Oriented RDF Database

We have developed an RDF database called CORD that is the foundation for our context management service. The context-manager itself is the wrapper agent that provides an interface for agent communication [5]. CORD implements most of the features enumerated as requirements for a CODBS in 1.1. After justifying the basic approach, we discuss in the following subsections those aspects of our solution that provide some contribution to the discussions within the Semantic Web research community.

⁷ Especially sensory data may be of a temporary nature.

2.1 Choosing the Semantic Web Technology

Obviously, the exchange of contextual knowledge must be based on a knowledge representation paradigm. On the other hand, profiles are nothing other than descriptions about distinct resources. These two statements alone, along with the fact that RDF provides solid concepts for not only describing resources, but also for modeling them, justify the selection of RDF, RDF schema, and OWL. Besides, the XML syntax of RDF fit perfectly into our multi-agent system, where XML was the content language of choice in agent communication messages.

2.2 Why a New Database Service?

Many of the projects dealing with RDF data stores use a relational DBMS (see, for example, the two surveys in [1] and [10] summarizing some of them). A general-purpose mapping of the RDF data model onto the relational model, where no assumptions about the type of resources being described are made, leads to the definition of few tables with few columns (see, for example, proposed DB schemas at [11]). Basically, if we consider the RDF data model as a set of triples, a three-column table will come up with a huge number of rows storing the statements, each with a subject, a predicate, and an object. Even if we consider the RDF data model as a directed, labeled graph, the relational database design will come up with similar results. With such a modeling, answering queries about complex resources may lead to many self-joins on one big table – depending on the entry point given by the query – where the consequences for the performance are not known.

Choosing an object-oriented database management system would not change the above situation, either. The issue is: relational or object-oriented DBMSs may meaningfully be used where a specific domain with concrete entity types is being modeled. That is, if you know the types of resources being described in your RDF data store, then you can provide a conventional database design with a meaningful database schema. The database schema would then reflect at least parts of what you state in the RDF/OWL schema for modeling the same resource types. A wrapper could then provide the knowledge stored in the database in terms of RDF statements to the world outside.

Due to the fact that the context management solution must be open for managing profiles of resources having arbitrary types, choosing a relational or object-oriented DBMS would confront us with the same dilemma as described in the previous paragraphs. On the other hand, a glance at our requirements, especially the requirements #3, #4, #5, and #9, shows that an existing database service may hardly satisfy all of them. Although most of the DBMSs do provide a triggering mechanism, even the utilization of stored procedures in the domain of relational databases or the class methods in the domain of object-oriented databases is no solution for the efficient recognition of interesting situations, that may be defined using complex conditions⁸. The main reason is that the situations to be recognized are not definable all at once, but their definitions will be added and removed dynamically. For the

⁸ Cp. also the discussion in section 2.5.

traditional database services, this would mean dynamism at the schema level. The concept of conditional values, discussed in section 2.5, is new and no direct support could be found in the domain of relational databases for storing them in arbitrary columns of arbitrary rows of tables. The methods in Object-oriented databases do not solve the problem, either, because they are defined within classes and are the same for all instances. The automatic inheritance of default values requires hierarchical relationship between rows of tables or instances of classes, which is not given, either. Finally, support for temporary values presupposes a timeline management for each column of each row or each field of each instance, which is not supported by traditional DBMSs.

For the purpose of profile management, we tried to provide the OWL schemas for user profiles, location profiles, terminal profiles, service profiles, and agent profiles (as a special case for service-offering software components) [15, 16]. Not only the term “profile management”, but also the complex structure of the above-enumerated profile types, the requirement for inheriting default values from more abstract profile instances in more concrete profile instances, and the concrete use cases in our projects caused us to choose profiles as our storage units. Having to meet the requirements listed in section 1.1, choosing profiles as the storage unit, having to manage profiles of arbitrary types, and considering the fact that each instance of the context manager deals with *few* instances of *complex* resources caused us to decide in favor of developing CORD.

2.3 Profiles and Their XML-based RDF Representation

A profile is a reusable resource that can be identified via a URI. This URI, which may be given as the value for *xml:base* in the XML representation of the profile, has the following structure:

```
cord://<host>:<port>/profiles/<profile-name>
```

Internally, profiles are implemented as (hashed) trees quite similar to *ldap* or *Windows™ registry*. The main difference with those solutions is the lack of a global root binding all of the (sub-)trees in one big tree integrating them.

As stated before, profiles are containers of key-value pairs. We call each such pair a *context element*, where the key serves both as the URI of the context element and as the source of its semantics. In the tree representation of profiles, however, there is no clear-cut distinction between keys and values. In addition to the leaves of the tree that represent the literal values or URI references, any node in the tree can be seen as a value associated with its path. Then, the path together with the base URI of the profile serves as the key. Except for the root of the tree that represents the whole profile resource (denoted as `cord://<host>:<port>/profiles/<profile-name>#`), all other branch nodes represent some embedded resource identified with a URI of the form `cord://<host>:<port>/profiles/<profile-name>#<path>`. Paths are made of *NDNames* (XML-names⁹ minus ‘.’) concatenated by dots (‘.’), e.g. `a.b.c` would be a valid path. Each NDName

⁹ See <http://www.w3.org/TR/REC-xml#NT-Name>.

corresponds to a property of the concrete resource addressed thus far. The possibility of using paths as part of keys meets the requirement for expressing complex structures via keys within profiles.

An example will further illustrate the usage of paths in keys. Let's assume that a schema with the URI `http://www.zgdv.de/CORD/schemas/UserProfile` defines, among others, the following concepts:

- the classes *UserProfile*, *PersonalInfo*, and *PersonName*.
- the property *personalInfo* with domain *UserProfile* and range *PersonalInfo*.
- the property *name* having *PersonalInfo* as its domain and *PersonName* as its range.
- the properties *first*, *middle*, *last*, and *nick* having *PersonName* as their domain and *xsd:string* as their range.

Then, the following RDF description represents my profile partially:

Sample 1. Partial RDF representation of a user profile

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns="http://www.zgdv.de/CORD/schemas/UserProfile#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xml:base="cord://st.zgdv.de:999/profiles/me">
  <UserProfile rdf:about="#">
    <personalInfo>
      <PersonalInfo rdf:about="#personalInfo">
        <name>
          <PersonName rdf:about="#personalInfo.name">
            <first>Mohammad-Reza</first>
            <last>Tazari</last>
            <nick>Saied</nick>
          </PersonName>
        </name>
      </PersonalInfo>
    </personalInfo>
  </UserProfile>
</rdf:RDF>
```

This results in the tree representation shown in figure 2 and the set of context elements shown in table 1.

Table1. Set of context elements (key-value pairs) resulting from Sample 1

Key	Value
<code>cord://st.zgdv.de:999/profiles/me#</code>	The whole profile resource
<code>cord://st.zgdv.de:999/profiles/me#rdf:type</code>	<code>http://www.zgdv.de/CORD/schemas/UserProfile#UserProfile</code>
<code>cord://st.zgdv.de:999/profiles/me#personalInfo</code>	The embedded resource rooted at 'personalInfo'
<code>cord://st.zgdv.de:999/profiles/me#personalInfo.rdf:type</code>	<code>http://www.zgdv.de/CORD/schemas/UserProfile#PersonalInfo</code>
<code>cord://st.zgdv.de:999/profiles/me#personalInfo.name</code>	The embedded resource rooted at 'name'
<code>cord://st.zgdv.de:999/profiles/me#personalInfo.name.rdf:type</code>	<code>http://www.zgdv.de/CORD/schemas/UserProfile#PersonName</code>
<code>cord://st.zgdv.de:999/profiles/me#personalInfo.name.first</code>	Mohammad-Reza
<code>cord://st.zgdv.de:999/profiles/me#personalInfo.name.last</code>	Tazari
<code>cord://st.zgdv.de:999/profiles/me#personalInfo.name.nick</code>	Saied

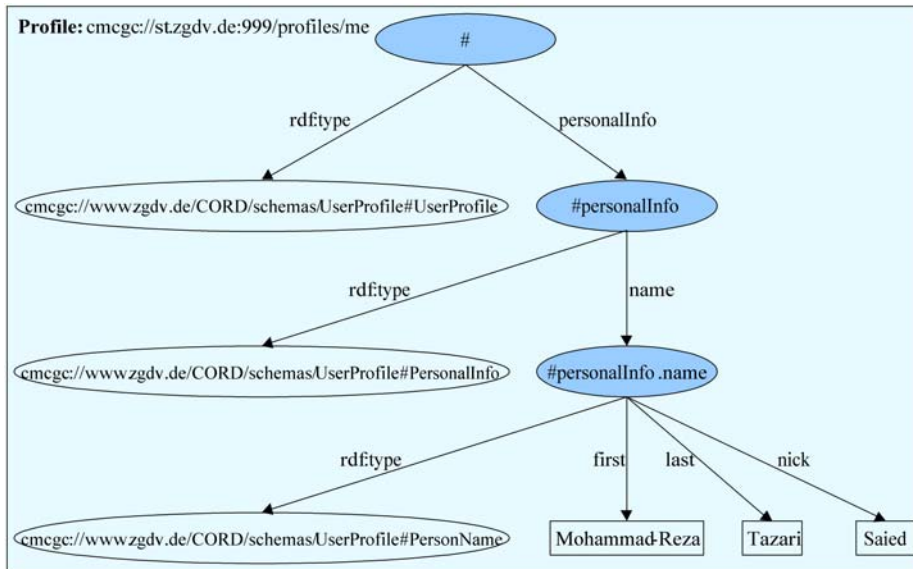


Fig. 2. Internal tree representation resulting from Sample 1 (leaf nodes have no background color)

Embedded Resources. The above model leads to some issues that are not handled in RDF or OWL standards. The most important issue concerns resources embedded in the profile. Each non-leaf node within the tree representation of a profile actually represents such an embedded resource as an identifiable resource.

The embedded resources result from either the *part-of* association – one of the fundamental concepts in object-oriented modeling – or the theorem of *weak entity classes* in database management. In UML, for example, associations having a diamond on one side indicate that the class on that side represents composite objects having instances of the class on the other side as their parts. They go even further and say that if the diamond is darkened, then the instances of the class on the other side may never exist independently from instances of the class on the side of the diamond – quite similar to the concept of weak entity classes in database management.

However, there is currently no way to specify part-of associations as such in RDF schema or OWL. CC/PP [8], as an RDF-based approach for articulation and exchange of contextual knowledge in profiles, has proposed the concept of *components* that can be seen as a solution for this problem. It was not consistent enough, though. It seems that the understanding of profiles in CC/PP is something like the “.ini”-files in Windows™; i.e. all attributes must appear in some component and no attributes within components may have another component as value. None of these restrictions matched our requirements. This approach ignores the evolution of such config-files into tree structures like Windows™ registry¹⁰. This is the main reason why we decided to establish the following conventions to enforce our idea of profiles:

¹⁰ [6], another research activity on context management, has similar criticisms on CC/PP.

- A schema modeling a profile type always defines a special class named after the schema itself that serves as the “main class”. All other classes defined in the schema are either super/sub-classes of the main class, appear in a (nested) part-of association, or represent some weak entity class. The root of a profile instance always represents an instance of this “main class” or its super/sub-classes.
- All resources contained in a profile instance other than the root of the profile have a path as their ID that results from concatenating (via dots) properties binding them to the root of the profile. Hence, property names may not contain dots.
- Many-valued properties refer to instances of `rdf:Alt`, `rdf:Bag`, or `rdf:Seq` as embedded resources with an ID built up in the same way as stated in the previous bullet. This has two implications: 1) the leaf nodes of a profile are always literal values or URI references to other resources and 2) if the elements of the container are some other embedded resources, then they have an ID resulting from appending a `‘.rdf_n’` to the ID of the container, where n is a decimal integer greater than zero with no leading zeros. This means that the requirement for supporting “sequences, bags, and sets of alternative values” is combined with the requirement for “expressing complex structures via keys within profiles”.
- In order to maintain profile boundary, all references to external resources are stored as URI references.
- All arcs that transform the tree representation into a graph are automatically redirected to point to a leaf having the equivalent local URI reference as its value.

2.4 The Query Language

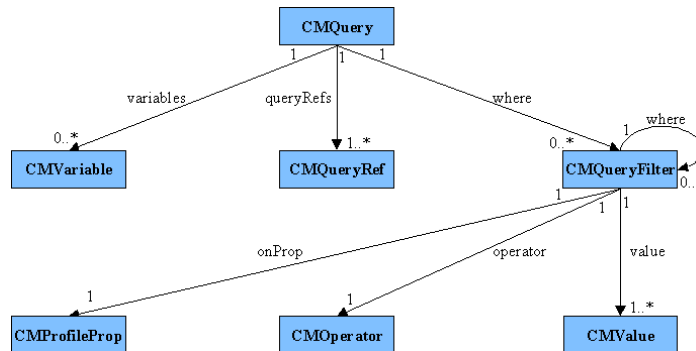


Fig. 3. Concept for CORD queries

Queries submitted to a CORD instance must be RDF descriptions based on the concept summarized in figure 3. Unlike the existing solutions that try to propose a general-purpose RDF query language concentrating on the syntax (see, for example, [7], [9], [12] and [14]), we have concentrated on the application of RDF in describing profiles and our related concepts like paths. Although we are practically using only the XML syntax of RDF to submit queries, deriving a compact SQL-like notation from the same concept is straightforward. In the following subsections, each of the main parts, from which CORD queries are formed, will be discussed in some detail.

Query References. A query must contain at least one query reference. Each query reference will be expanded to a set of matched keys (compare table 1) and an RDF description containing the context elements identified by those keys will be returned as query result. A query reference is a CORD URI-reference with the following wildcarding possibilities (most combinations of them are also legal):

- If the `<host>:<port>` slot is missing, then the CORD instance receiving the query will consolidate all other known CORD instances in the ascertainment of the query results.
- If the profile name is missing, then all profiles managed by the CORD instance will match.
- If the query reference ends with the fragment separator (`#`), then the whole content of the profile will match.
- If the `<path>` slot begins with a dot (`.`), then any context element having the remainder of the `<path>` slot as the suffix of its own path will match.
- If the `<path>` slot ends with a dot (`.`), then the sub-tree rooted at the resource matching the leading part of the `<path>` slot will match.
- If the `<path>` slot contains two subsequent dots (`..`), then any context element within the sub-tree rooted at the resource matching the leading part of the `<path>` slot and having the remainder of the `<path>` slot as the suffix of its path will match.

The special case of `cord://<host>:<port>/profiles/`, where again the `<host>:<port>` slot may be left empty, will cause a query result to contain only a bag of URI references to the matching profiles without the descriptions of their contents.

Variables. A query may have a *sequence* of initialized variables that store literal values or URI references. A subsequent variable may use a previous variable storing a URI reference to store a subordinate value (cp. last paragraph in this section about the usage of variables). This facilitates, among other things, the switching to referenced profiles and inter-profile joins.

Variables may be of type *KeyVariable* or *ValueVariable*. The sub-type influences the interpretation of the value to be assigned to the variable. In the case of key variables, it is expected that the value is a URI reference that must be resolved so that its associated value is assigned to the variable. In the case of value variables, however, the value given will be assigned to the variable as-is, be it a URI reference or not.

There are some predefined variables that are set automatically, normally just before CORD begins to process a new request:

- The current time is stored in variables like *currentTime*, *amPM*, *dayOfMonth*, *dayOfWeek*, etc., quite similar to the constant fields defined in `java.util.Calendar`.
- The certificate of the agent that sent the request is stored in *accessor*. This is interesting for the definition of conditional values (see section 2.5), when the accessor plays a role in the decision about the applying value.
- As stated before, a query reference will be expanded to a set of matched keys. Each time, after selecting one of the matched keys for further processing, two variables are set automatically that keep their values until CORD leaves the context

of that matched key. These are *currentProfile*, which contains the URI reference of the profile from which the matched context element originates, and *matchedComponent*, which contains the URI reference of the lowest embedded resource matched during the expansion of the query reference into the matched key.

- Two other special variables are set automatically whenever a context element is added, updated, or deleted. These variables are only interesting for the processing of subscriptions (see section 2.5). They are *triggerKey*, which contains the URI reference of the changed element, and *triggerValue*, which contains the new value associated with the key of the context element, if applicable. The setting of *triggerKey* causes the time variables and the *currentProfile* variable to be set automatically in the context of processing the DB event.

In general, whether predefined or defined by the requestor, variables can be used to build up new query or other URI references, can be used in query filters or conditions of rules (see section 2.5), or wherever values are expected. Variable substitution occurs whenever a special construct is found in a way similar to macro expansions in the C programming language. For example, assuming that the standard variable *currentProfile* contains a reference to a user profile in a special context, one can use it in the following construct in place of a direct value:

```
<cord:VarRef>
  <cord:variable rdf:resource="&cord;currentProfile"/>
  <cord:suffix>personalInfo.name.last</cord:suffix>
  <cord:action rdf:resource="&cord;substituteAndEval"/>
</cord:VarRef>
```

If at runtime the variable has the value `cord://st.zgdv.de:999/profiles/me#`, then, due to the specified *action*, the above variable reference is first replaced and expanded to `cord://st.zgdv.de:999/profiles/me#personalInfo.name.last` which will then be evaluated to the literal value `Tazari`. Other possible values for *action* are *substitute* and *evalAndSubstitute*. Another property of the *VarRef* class not used in the above example is *cord:prefix*.

Query Filters. A container of query filters can be used to select only a subset of the matched keys resulting from the expansion of query references. If the container is of type *rdf:Seq*, then an implicit and-connector is assumed between the query filters given in the sequence; in the case of *rdf:Alt*, an implicit or-connector is assumed. Beside query filters, elements of such containers may also be a container of the other type to switch between connector types¹¹.

A query filter says which criterion must be satisfied in order to keep a previously matched context element in the set of those to be returned in the query result by specifying *what* must be compared *how* with *which value(s)*. To specify the *how*, one must select an operator from the enumeration defined by *CMOperator*. Currently, the

¹¹ The point with the container type and its relation with the connector type and the possibility of nesting them to switch from one connector type to another is not shown in figure 3 in order to keep the model straightforward.

possible values are *equal*, *greater*, *less*, *in*, *including*, *notEqual*, *notGreater*, *notLess*, *notIn*, and *excluding*.

The criterion must be selected from the enumeration defined by *CMProfileProp* and given as the value (in the form of a URI reference) for a property called *onProp*. The possible values are basically:

- *schema*: to select context elements coming from a specific profile type. For example, to filter context elements coming from user profiles, one may define a query filter on *schema* property, choose the *equal* operator, and give <http://www.zgdv.de/CORD/schemas/UserProfile> as a single URI reference for the *value* property.
- *parents/children*: to select context elements from a profile that has the given profiles as its parents/children.
- *begin/end/importance/priority*¹²: to select context elements whose values are valid during a certain time period (for temporary context values) or satisfy certain weighting criteria (not to be discussed further).
- *value*: to select context elements whose values satisfy the given condition.
- *currentProfile/matchedComponent*: combined with a suffix for building up a new query reference, they can be used to filter the matched context elements further. The resulting query reference forms a sub-query with the possibility to check the values returned by the sub-query in the same *CMQueryFilter* and to further filter them based on the conditions provided by the optional *where* property.

2.5 Rules

CORD supports two forms of rules that are structured similarly: one for forming conditional values and the other one for posting subscription requests. These are discussed in the following subsections.

Conditional Values. Values (especially those given for preferences) can be rule-based, in the sense that the value depends on some contextual state or situation. When a rule-based value is queried, first the cases within the rule will be examined using the current values of referenced context elements. If one of the alternative cases applies, then the associated value is returned, otherwise *rdf:nil*. Figure 4 summarizes the CORD concept for conditional values.

Basically a CORD rule is a “switch-case” construct. Each case has a condition part and a value part. The cases are considered in the *sequence* of their specification. As soon as a case is found whose condition part evaluates to true, the evaluation will cease and the value associated with that case is used as the result of the evaluation. A case without any conditions always evaluates to true. The condition part of each case is a container (of type *rdf:Seq* or *rdf:Alt* quite similar to the containers of query filters – see also footnote 9) of comparisons, where normally values of context elements are compared with literal values or with values of other context elements.

¹² Special properties introduced by CORD and applicable to all nodes within a profile.

Due to some complications in the implementation, conditional values are currently a special case of literal values; this leads to two side effects: 1) the restriction for cases to contain only one value (literal or URI-reference) and 2) the delay in parsing until the conditional value is accessed.

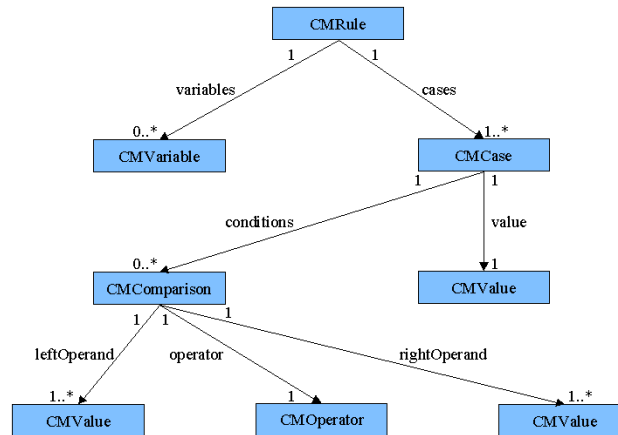


Fig. 4. Concept for conditional values in CORD

From another perspective, we can say that conditional values equip CORD with something like “passive inference”. With “passive inference” we mean that on the one hand CORD does not have its own rules to infer the new state of the contextual knowledge, but the logic of inferring comes from “producers”/“providers” of the contextual knowledge. On the other hand, the inferring process only leads to a selection between suggested alternatives depending on the current situation (cp. [5] for our concept of situations as contextual states).

The concept of variables and variable references is already discussed in section 2.4. However, an interesting aspect of using variable references in comparisons is that the corresponding variables don’t have to be defined in the variables part of the rule. If they are defined in the rule, they will usually refer to some shared contextual facts (facts known to the CORD instance at the time of rule interpretation). But, since such rules are interpreted at query time, one may define the variables in his or her queries. This way, the facts to be used in the evaluation of the rule may be the premises of the requestor.

Last but not least, a special feature resulting from the inheritance of default values is worth mentioning. Assuming that the profile p_1 is a parent of the profiles p_2 and p_3 , and a conditional value defined in p_1 is being inherited by both p_2 and p_3 , the use of local URI references in the rule might cause the same rule to return a different value in the context of p_2 compared to the value returned in the context of p_3 , even if the two evaluations are performed “simultaneously”. An example may illustrate this nice effect better: Assume that a travel planning agent stores a profile for each travel to be planned in the user’s personal context-manager letting it inherit from the default travel profile provided by the context-manager of the company where he works. If a

property *transportMeans* in the default profile has a conditional value in the following simplified form

```

if #distance.inKm greater 500
  return my:plane
else if #numberOfCompanions greater 1
  return my:rentACar
else
  return my:train

```

then a query about the transport means for a concrete travel where the user must travel alone to a city 225 km far from his residence would result in `my:train`.

Subscription Requests. Requestors may subscribe for notification or other actions to be performed by CORD. There are two kinds of subscriptions: simple or conditional.

A simple subscription is formed from a bag of query references and causes CORD to immediately inform the current context elements whose keys match the given query references. Additionally, CORD will watch for changes of context elements and will inform the subscriber about the new value whenever the key of the changed context element matches one of the given query references. Changes of the value will occur due to insert, update, and delete requests or due to the expiration of a time-stamped value (which causes the use of the next alternative value as the current value). This way, from the time of subscription, the subscriber will always know about the state of all context elements known to CORD (or made known at any time in the future) whose keys match the given query references.

The conditional version is based on Event-Condition-Action rules (ECA rules). The only events supported are again changes of values in the sense of the previous paragraph; hence, the “event” part of an ECA rule is nothing other than a bag of query references wildcarding those context elements whose change of states should trigger the evaluation of the condition-action part. This latter part of an ECA rule is quite similar to the rules outlined in the previous subsection. That is, there are variables, and conditions are structured exactly the same way as in the case of conditional values, i.e. containers of comparisons of context and constant values in a switch-case construct. Unlike those rules, instead of a value, a sequence of actions can be specified for each case. Actions are operations that can be done by CORD, which include:

- Sending the current values of some specified context elements to the subscriber or other receivers.
- Sending a given literal message to the subscriber or other receivers.
- Inserting new context elements or updating existing ones; this triggers other events and may lead to indirect notifications.

Usually, the subscribers specify the bag of query references in ECA rules in such a way that the keys of all context elements that play a role in the condition part would match those query references. Such rules are quasi “alive”: as soon as a case applies, it will be recognized. Therefore, they are the cornerstones for situation recognition for our context management service [5].

2.6 Insert, Update, and Delete

Insert and update requests must be submitted with the corresponding RDF descriptions, such as the one given in Sample 1. Delete requests, however, must contain a query description, which will lead to the deletion of matched context elements.

3 Summary and Future Work

We showed that contextual knowledge plays an important role in the Semantic Web and concluded that context management is *the* missing service in the Semantic Web. Our work contributes to filling this gap through the development of CORD, the context-oriented RDF database. CORD provides a solution based on the Semantic Web technology mainly for managing profiles. The concrete contributions of this paper are the introduction of: 1) a storage system for RDF-based profile data handling embedded resources, 2) a query language suitable for querying data organized in profiles, 3) a concept for storing rule-based values in profiles, and 4) a model for subscribing to context management services via the so-called event-condition-action rules.

We will continue this work by: 1) consolidating data from sources other than instances of CORD to satisfy the requirement #2 from section 1.1 completely, 2) equipping CORD with a special logic for reasoning about user location when sensory data is missing, based on information provided by PIM applications and the history of the location data, and 3) enhancing the existing privacy protection mechanism¹³ by employing P3P and APPEL concepts when dealing with public service providers.

Acknowledgement. This work is partially sponsored by the Information Society DG of the European Commission. It is part of the MUMMY project (IST-2001-37365, Mobile Knowledge Management – using multimedia-rich portals for context-aware information processing with pocket-sized computers in Facility Management and at Construction Site) funded by the Information Society Technologies (IST) Programme. See <http://mummy.intranet.gr>.

References

1. Barstow, A (2001). Survey of RDF/Triple Data Stores. World Wide Web Consortium. Retrieved April 10, 2003 from <http://www.w3.org/2001/05/rdf-ds/DataStore> (last update Feb. 26, 2003).
2. Berners-Lee, T. & Hendler, J. & Lassila, O. (2001). The Semantic Web. Scientific American, May 17, 2001. Retrieved February 26, 2003 from http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

¹³ Currently, CORD provides access control mechanisms for the communication with context-providing and -consuming components that belong either to the user or to the communities of which the user is a member.

3. deVos, A. (2002). An RDF Query Language Based on DAML. Langdale Consultant, revision 1.0, Feb. 2002. Retrieved April 10, 2003 from <http://www.langdale.com.au/RDF/DAML-Query.html>.
4. Fikes, R. & Hayes, P. & Horrocks, I. (2002). DAML Query Language (DQL) – Abstract Specification. The Joint United States / European Union ad hoc Agent Markup Language Committee, August 2002. Retrieved April 10, 2003 from <http://www.daml.org/2002/08/dql/dql>.
5. Grimm, M. & Tazari, M.R. & Balfanz, D. (2002). Towards a Framework for Mobile Knowledge Management. Proceedings of the 4th international conference on Practical Aspects of Knowledge Management (PAKM2002), Vienna, Austria, December 2002.
6. Indulska, J. & Robinson, R. & Rakotonirainy, A. & Henriksen, K. (2002). Experiences in Using CC/PP in Context-Aware Systems. Proceedings of the 4th International Conference on Mobile Data Management (MDM2003), Melbourne, Australia, January 2003. Lecture Notes in Computer Science. Springer Verlag, LNCS 2574. pp. 247-261.
7. Karvounarakis, G. & Alexaki, S. & Christophides, V. & Plexousakis, D. & Scholl, M. (2002). RQL: A Declarative Query Language for RDF. ACM 1-58116-449-5/02/0005, WWW2002, Honolulu, USA, May 2002.
8. Klyne, G. & Reynolds, F. & Woodrow, C. & Ohto, H. & Hjelm, J. & Butler, M.H. & Tran, L. (2003). Composite Capability / Preference Profiles (CC/PP): Structure and Vocabularies. <http://www.w3.org/TR/CCPP-struct-vocab/>, W3C Working Draft March 25, 2003.
9. Kokkelin, S. (2001). Transforming RDF with RDFPath. Working draft, March 2001. Retrieved April 10, 2003 from <http://zoe.mathematik.uni-osnabrueck.de/QAT/Transform/RDFTransform.pdf>.
10. Magkanaraki, A. & Karvounarakis, G. & Anh, T.T. & Christophides, V. & Plexousakis, D. (2002). Ontology Storage and Querying, Technical Report No. 308. Foundation for Research and Technology Hellas, Institute of Computer Science, Information Systems Laboratory. Crete, Greece, April 2002. Retrieved April 10, 2003 from ftp://ftp.ics.forth.gr/tech-reports/2002/2002_TR308.Ontology_Storage_and_Querying.pdf.
11. Melnik, S (2001). Storing RDF in a Relational Database. Retrieved April 10, 2003 from <http://www-db.stanford.edu/~melnik/rdf/db.html> (last update Dec. 3, 2001).
12. Miller, L. & Seaborne, A. & Reggiori, A. (2002). Three Implementations of SquishQL, a Simple RDF Query Language. Proceedings of the 1st International Semantic Web Conference (ISWC2002), Sardinia, Italy, June 2002. Retrieved April 10, 2003 from <http://www.hpl.hp.com/techreports/2002/HPL-2002-110.pdf>.
13. Schirmer, J. & Bach, H. (2000): Context-Management within an Agent-based Approach for Service Assistance in the Domain of Consumer Electronics. In: Proceedings of Intelligent Interactive Assistance, Mobile Multimedia Computing, Rostock, Germany, November 2000.
14. Sintek, M. & Decker, S. (2002). TRIPLE — A Query, Inference, and Transformation Language for the Semantic Web. Proceedings of the 1st International Semantic Web Conference (ISWC2002), Sardinia, Italy, June 2002. Retrieved April 10, 2003 from <http://triple.semanticweb.org/iswc2002/TripleReport.pdf>.
15. Tazari, M.R. & Grimm, M. & Finke, M. (2003). Modelling User Context. Proceedings of the 10th International Conference on Human-Computer Interaction (HCI2003), Crete (Greece), June 2003.
16. Tazari, M.R. & Plößler, K. (2003). User-Centric Service Brokerage in a Personal Multi-Agent Environment. To be presented in the International Conference on Integration of Knowledge Intensive Multi-Agent Systems (IEEE KIMAS'03), Cambridge MA (USA), October 2003.

An Adaptable Service Connector Model¹

Gang Li¹, Yanbo Han¹, Zhuofeng Zhao¹, Jianwu Wang¹, and Roland M. Wagner²

¹ Software Division, ICT, Chinese Academy of Science, PRC

{ligang, yhan}@ict.ac.cn {zhaozf, wjw}@software.ict.ac.cn

² Fraunhofer ISST, Dortmund, Germany

roland.wagner@isst.fhg.de

Abstract. The volatility of network environments requires service connections to adapt to changes of service resources and user requirements. In this paper, we treat service connections as individual components called service connectors and present an adaptable service connector model that adopts a role mechanism to adjust connections between services. A role is an abstraction of services with common functionalities. It offers a changeable connector structure, enables reconfiguration of service interaction and encapsulates changes in interacting participants, making service connections more adaptable.

1 Introduction

Service oriented computing is gaining popularity. In a typical contemporary service-oriented application, service connections are pragmatically implemented using protocols like SOAP. Through this kind of connection, services can be composed into applications. Service composition is regarded as a new approach for developing applications in network environments.

However, service composition still faces serious challenges due to the openness and dynamism of network environment, such as grids [1][2]. Let us take service grids as an example. Firstly, services freely join in or quit from a grid and most services in a grid continue evolving over time. Secondly, user requirements are subject to dynamic changes in a virtual enterprise environment. All these require service connections to be adaptable, so that service interactions can be easily reconfigured and involved services can be changed dynamically, while changes of service resources and user requirements take place. In this paper, we focus our research on how to make a service connection adapt to those changes.

Existing ways that are used to connect services includes control flow based connections [3], data flow based connections [4][5] and hybrid forms of these two types, such as service connection mechanisms in GSFL [6] and BPEL4WS [7]. After setting up an interaction channel between ports of different services through protocols, the validity of control flow based service connection is determined by service states. This type of connection is set up following interaction protocols inhering in services. For it is predefined and fixed, changes of service interactions or requirements tend to invalidate the connection. A data flow based connection links services through data dependencies. With shared data, this type of connection is free from influences caused by service port changes to some extent. However, because of its implicit definition, the structure of data flow based service connections is indistinct, which makes it difficult to adjust the connection. Although in the hybrid form above the two types complement each other, it does

¹ This paper is supported by the Young Scientist Fund of ICT Chinese Academy of Science under Grant No. 20026180-22 and the National Natural Science Foundation of China under Grant No. 60173018.

not contribute much to service connection adaptation. For example, BPLE4WS provides partner link types to describe service connections as partner links, but it does not offer methods for dynamically changing these links.

To make the connection adaptable, the connection structure ought to be changeable. As a semantic concept, role [8][9] provides an organizing mechanism through which the abstraction of services with common functions is derived and marked by role features. With this mechanism, a role offers flexible connection structure, enabling service connection adaptation by reconfiguration. Based on the above rationale, we implement a service connection as an explicit component named service connector and present a role-based service connector model. In this model, a role is used to fulfill the adaptation of a service connection with a stable interaction interface and a changeable connection structure.

The remainder of this paper is divided into 3 sections. Section 2 addresses the role-based service connector model in detail, including its connection structure and interaction protocol. With a case study, section 3 demonstrates the support of the model to making service connection adaptable. Finally, the contribution of our research is concisely summarized in section 4.

2 Role-based Service Connector Model

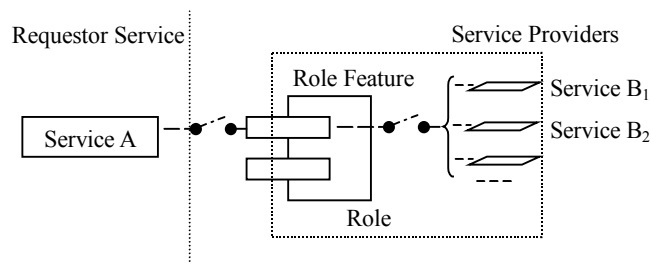


Fig. 1. Sketch Map of Role-based Service Connector Model

Figure 1 illustrates the main idea of the role-based service connector model by an example that Service A interacts with a role. Described by its features, the role is an abstraction of services such as Service B1 and Service B2. As a stable interface for service interaction, role features present functions provided by service providers that interact with the requestor services through the role. Marked by features, functions that a role provides are implemented by service providers that are invisible to requestors. When unanticipated changes cause a modification of a service interaction, the connector can adapt to changes of service interactions or requirements by reconfiguring its role features or service providers. When a service provider involved in an interaction is unavailable, another one with the same role feature can replace it, enhancing the adaptability and reliability of the service connection. A role is a virtual service, and it offers not only a changeable connector structure, but also a unified service interaction interface, providing essential support to connection adaptation in dynamic service composition [14].

Details about the role-based service connector model follow, namely aspects of the connection structure and the interaction protocol.

2.1 Connection Structure

In this section, to present the role-based service connector model precisely and concisely, we give its definition, related semantic and interaction protocol in a formal way. Based on this formalism, we prove that a role-based service connector is adaptable.

Definition 1 Role-based Service Connector

Given a three-tuple $\langle Name_r, Features_r, Service_r \rangle$, where $Name_r$ marks a role name; $Features_r$ is a set of role features, $Features_r = \{f_r | f_r = \langle rn, fn_r, va_r \rangle\}$, where rn denotes the role that feature f_r belongs to, fn_r is the name of f_r , va_r is the argument vector of f_r . Role features are an interface with which a role interacts with the environment; $Service_r$ is a set of service references, which are related to features.

The three-tuple defines a role-based service connector, if and only if it has the following properties:

- There is a function set, denoted as Map . Given $\forall f_r \in Features_r$, then $\exists m \in Map, S_{sr} \subseteq Service_r, m(f_r) = S_{sr}$, and services that belong to set S_{sr} have the same interaction interface marked by f_r .
- There is another function set, denoted as $Selectors$, consider $f_r, \exists sel \in Selectors, Ser \in S_{sr}, sel(S_{sr}) = Ser$.

The function m is named as feature mapping function, and the function sel is named as service selection function.

Thus a service connector presents a configurable service called instead of a requested service. The connector selects an appropriate provided service and maps its parameters to the request. Therefore it is a configurable encapsulation of a service or a group of semantically similar services, abstracted as a role. There are two interaction patterns involved in a role-based service connector model, namely the service-role and role-role patterns. In the first pattern, service providers are packaged by the role, which is depicted by figure 1. It fits the context where service providers are volatile. In the second pattern, both providers and requestors are packaged by roles, which fits the context where both sides are volatile.

The main parts of the model are presented above. Now, we define its connection semantic using category theory to deduce connector properties on adaptation. A category consists of object sets and sets of morphisms between objects, which focuses on describing and analyzing relations among any type of objects [10]. With category, relations among services and roles can be described formally, which offers an approach for analyzing and deducing properties of a role-based service connector in a method independent of implementation details. Hereby it presents the connection semantics in a role-based service connector model with categories that takes structured sets as objects.

Let $Service$ be the service involved in the interaction, $Service = \langle Name_s, Features_s \rangle$, where $Name_s$ is the service name, $Features_s = \{f_s | f_s = \langle sn, fn_s, va_s \rangle\}$, f_s is the service feature that describes a service port. Take $Services$ as objects, and construct the category $Serv$.

$Serv = \langle objC_{Serv}, MorC_{Serv}, dom_s, cod_s, \circ \rangle$, where The set $objC_{Serv}$ is the class of objects in $Serv$;

Given morphism $\varphi: objC_{Serv} \rightarrow objC_{Serv}, E, F \in objC_{Serv}, \varphi(E) = F, \varphi$ is specified by the following:

$$\delta_1: Name_E \rightarrow Name_F$$

$$\delta_2: Feature_E \rightarrow Feature_F$$

The set $MorC_{Serv}$ is the class of morphisms in $Serv$, namely $MorC_{Serv}$ is the set of connections between $Services$. The morphisms in $MorC_{Serv}$ is defined by φ .

The function dom_s is defined as $dom_s: MorC_{Serv} \rightarrow objC_{Serv}$, where $f \in MorC_{Serv}$, then $dom_s(f)$ denotes the domain of f .

The function cod_s is defined as $cod_s: MorC_{Serv} \rightarrow objC_{Serv}$, where $f \in MorC_{Serv}$, then $cod_s(f)$ denotes the codomain of f .

The symbol \circ denotes an operation, which is defined as $\circ: MorC_{Serv} \times MorC_{Serv} \rightarrow MorC_{Serv}$.

Let *Connector* be the role-based service connector involved in the interaction, $Connector = \langle Name_r, Features_r, Service_r \rangle$. In a similar way, take *Connectors* as objects, and construct the category *Conn*, $Conn = \langle objR_{Conn}, MorR_{Conn}, dom_c, cod_c, * \rangle$, where morphism ψ is used to define $MorR_{Conn}$ and describes interactions between roles.

Let $Ser \in objC_{Serv}$, $Con \in objR_{Conn}$, constructs function F_{SR} , $F_{SR}: objC_{Serv} \rightarrow objR_{Conn}$, $F_{SR}(Ser) = Con$, F_{SR} is specified by the following:

$$\eta_1: Name_{Ser} \rightarrow Name_{Con}$$

$$\eta_2: Feature_{Ser} \rightarrow Feature_{Con}$$

The connection fulfilled by a role-based service connector is marked as $Connection_R$. $Connection_R = \{ \langle Ser, F_{SR}(Ser) \rangle \} \cup \{ \langle Con, \psi(Con) \rangle \}$.

After giving connection semantics, we can now prove the following results according to it:

Theorem 1

In interaction where roles are involved, a role-based service connector has the following properties:

- Changes in the service set that correspond to a role feature do not cause changes of the connection;
- When the service set that corresponds to a role feature do not satisfy some requirements of requestors, the connection can adapt to these changes by reconfiguring the role feature and related services.

Proof:

(1) Let $r1$ be the role involved in an interaction. According to the definition of $r1$, we conclude:

$$\forall f_x \in Feature_{r1}, \exists m \in Map_{r1}, m(f_x) = s_{sx}, s_{sx} \subseteq Service_{r1}, \text{ and } \exists sel \in Selectors_{r1}, sel(s_{sx}) = Serx, Serx \in s_{sx}.$$

When changes take place in s_{sx} , a new function sel' can be constructed, $sel' \in Selectors_{r1}$, such that $sel'(s_{sx}) = Serx'$.

$\therefore Serx'$ and $Serx$ have the same features, and the requestor service interacts with the service that belongs to s_{sx} through the role feature.

\therefore There are no changes in the connection.

Proposition (1) follows.

(2) When the service set that corresponds to a role feature does not satisfy some requirements of the requestor services, the interaction interface or the service set has to be changed. If only the service set is to be adjusted, the connection can adapt to those changes according to proposition (1). If the interaction interface is to be changed, it results in a change of the role feature according to

Definition 1. Then proposition (2) can be proved as follows:

i) when changes take place in a service–role pattern,

let $s, r1$ be the service and role involved in the interaction, and $f_s \in Feature_s$. According to F_{SR} , we conclude:

$\exists \langle f_s, f_{r1} \rangle, f_{r1} \in Feature_{r1}$. When $\langle f_s, f_{r1} \rangle$ changes into $\langle f_s, f'_{r1} \rangle$, according to the definition of $r1$, $\exists m' \in Map_{r1}$, such that $s_{sx1}' \subseteq Service_{r1}$, $m'(f'_{r1}) = s_{sx1}'$. m' and s_{sx1}' keep the connection available.

ii) when changes take place in role–role pattern,

let $r1, r2$ be the roles involved in the interaction, and $f_{r1} \in Feature_{r1}$. According to ψ , we conclude:

$\exists \langle f_{r1}, f_{r2} \rangle, f_{r2} \in Feature_{r2}$. When $\langle f_{r1}, f_{r2} \rangle$ changes into $\langle f_{r1}, f'_{r2} \rangle$, according to the definition of $r2$, $\exists m'' \in Map_{r2}$, such that $s_{sx2}' \subseteq Service_{r2}$, $m''(f'_{r2}) = s_{sx2}'$. m'' and s_{sx2}' keep the connection available.

According to i) and ii), proposition (2) follows.

The proof is now complete.

Theorem 1 says that a role-based service connector is adaptable. In $Connection_R$, there is a loose coupling between interacting participants, which can be changed by adjusting the service selection

function such as $sel()$. Through role features, services expose a unified interaction interface to requestor services, reducing the influences between services. When unanticipated changes occur, the connection can be reconfigured through changing the involved role features and services. In addition, a role feature can be implemented by several candidates of service providers like services in S_{sx} , which makes the connection adaptable.

2.2 Interaction Protocol

When a role is introduced, changes occur in service interaction patterns to support connection adaptation. As mentioned above, service-role and role-role patterns are the main interaction manners in service compositions where roles are involved. In order to describe them clearly, the interaction protocols are presented in a formal way.

While analyzing security protocols, I. Cervesato etc. used a multiset rewriting formalism, based on linear logic. The existential quantification in it provides a succinct way of choosing new values. Besides that, it has a bounded initialization phase, but allows unboundedly many instances of each protocol participant, making it especially qualified to analyzing finite-length protocols [11]. In this section, we present the interaction protocol of a role-based connector model with this method.

(1) Interaction in a service-role pattern

$$\begin{aligned}
& Ser^0(), R^0() \\
& Ser^0() \rightarrow \exists x. Ser^1(x), Con^1(x) \\
& R^0(), Con^1(x) \rightarrow \exists y. R^1(x, y), \exists Ser'^0(), Con^2(x, y) \\
& Ser'^0(), Con^2(x, y) \rightarrow \exists z. Ser'^1(y, z), Con^3(x, y, z) \\
& R^1(x, y), Con^3(x, y, z) \rightarrow R^2(x, y, z), Con^4(x, z) \\
& Ser'^1(x), Con^4(x, z) \rightarrow Ser^2(x, z)
\end{aligned}$$

In service-role pattern, service Ser is the requestor service, and $Ser^0()$ denotes that service Ser is in initial state 0. Then it produces message x in state 0, sends x to role R transforming into state 1 that keeps message x ; $Con^1(x)$ denotes that the connection is in state 1 that keeps message x . After role R receives x at state 0, it produces message y , then with the service selection function, role R chooses a service Ser' as server according to interaction state, and sends y to Ser' transforming into state 1. After processing message y , service Ser' produces message z that contains the results the client required and sends z to role R . After role R receives z at state 1, it delivers message z to service Ser transforming into state 2; service Ser receives z at state 1 transforming into state 2.

(2) Interaction in a role-role pattern

$$\begin{aligned}
& Ser^0(), R^0(), R'^0() \\
& Ser^0() \rightarrow \exists x. Ser^1(x), Con^1(x) \\
& R^0(), Con^1(x) \rightarrow \exists y. R^1(x, y), Con^2(x, y) \\
& R'^0(), Con^2(x, y) \rightarrow \exists z. R'^1(x, y, z), \exists Ser'^0(), Con^3(x, y, z) \\
& Ser'^0(), Con^3(x, y, z) \rightarrow \exists w. Ser'^1(x, y, z, w), Con^4(x, y, z, w) \\
& R^1(x, y, z), Con^4(x, y, z, w) \rightarrow R'^2(x, y, w), Con^5(x, y, w) \\
& R'^1(x, y), Con^5(x, y, w) \rightarrow R'^2(x, w), Con^6(x, w) \\
& Ser'^1(x), Con^6(x, w) \rightarrow Ser^2(x, w)
\end{aligned}$$

A role-role pattern is a combination of two service-role patterns. In this pattern, request and return values pass through two roles.

Above protocols describe how to configure the connector automatically. Besides that, they emphasize especially on states of connections and interaction participants. On the one hand, a role supports the dynamic selection of qualified services according to connection states and service states; on the other hand, when the connection is to be reconfigured, these states determine whether the reconfiguration is feasible. During connection reconfiguration, states of interacting participants are saved. After reconfiguring, they are restored to enable service composition to be resumed. It shows that the interaction protocol of a role-based service connector supports dynamic connection and adaptation.

After presenting the model and demonstrating its adaptability in a formal way, we now present a case study in the following section.

3 Exploring the Model with a Case Study

A role-based service connector is more adaptable than others, when changes take place in service resources and/or requirements. In order to strengthen this conclusion and illustrate how to use role-based service connectors, we briefly present a case study of representing the model in XML and applying it in project FLAME2008². Note that the case study focuses on the aspect of connection adaptabilities, other details about the case are beyond this scope.

3.1 A Real Case from FLAME2008

The Olympic Travel Planning application, which is a part of FLAME2008, is to provide pertinent information to those who watch match and tour in Beijing during Olympic Games 2008. Figure 2 presents a requirement segment of the application, which involves the following services.

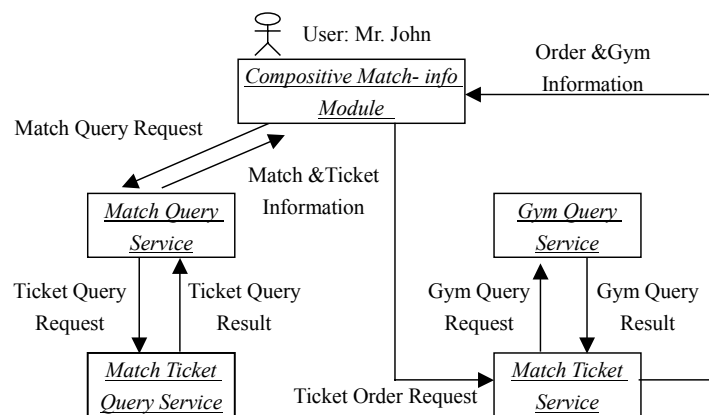


Fig. 2. A Requirement Description Segment of Olympic Travel Planning Application

- *Match Query Service*: retrieving a match schedule, getting information about the match that one wants to watch, and invoking *Match Ticket Query Service*.
- *Match Ticket Query Service*: retrieving match ticket information.

² FLAME2008 (Project Flexible Semantic Service Management Environment) is to develop grid service based applications that provide integrated, personalized information services to the public during the Olympic Games 2008. The project is supported by MOST PRC and CAS under Grand No.20012019.

- *Match Ticket Service*: ordering match tickets after making sure that there are remains, and invoking the *Gym Query Service*.
- *Gym Query Service*: retrieving information about traffic, gym location and so on.
- *Compositive Match-info Module*: coordinating above services to get information about match schedules, match tickets, gym and ordering tickets.

In the initial stage of the project, we constructed a prototype by composing services. However, those services and their connections were changing with service resources changing and requirements evolving.

- **Change I**: In the prototype, the response time of *Match Ticket Query Service* was too long to endure, so that the service had to be replaced by a new one.
- **Change II**: Previously, after ordering tickets, audiences wanted to know something about the gym where the match was to be held. So, *Match Ticket Service* interacted with *Gym Query Service*. Now, audiences want to retrieve order results after ordering. To meet the changed requirement, *Match Ticket Service* is to interact with *Order Result Query Service* (This service is a service to be added when requirements change, and it is not shown in figure 2). In this situation, both the interaction interface and services are changed.

3.2 Representations and Applications of the Model

In the initial prototype of FLAME2008, services were connected by control flow based on SOAP messages. To adapt to changes in service providers, the adaptors were modified. However, when the above-mentioned changes occurred, it was very difficult to adjust the application, for the service connections were almost unchangeable. In order to change them, we had to read through the source codes, and modify the processing logic or rebuild the module. It involved many efforts of understanding source code, coding and so on. The modified application was prone to throw exceptions yet.

Hence, we partially adopted and realized role-based service connector in the second prototype, and constructed a supporting tool named CAFISE Framework. Compared with service connections in the first prototype, role-based service connector can be adjusted smoothly with the framework that is depicted in figure 3.

3.2.1 CAFISE Framework

The CAFISE Framework includes a set of essential components and tools assisting to construct and adjust applications. From a business viewpoint, the Convergent Modeling Tool helps designers to present their requirements, and then the requirements are transformed into an executable application specification using XML, which is presented in figure 3 as Specification A. Those specifications describe the coordination among all involved services. While the CAFISE Virtual Machine interprets the specification, the required services of the Service Community are dynamically bound and invoked.

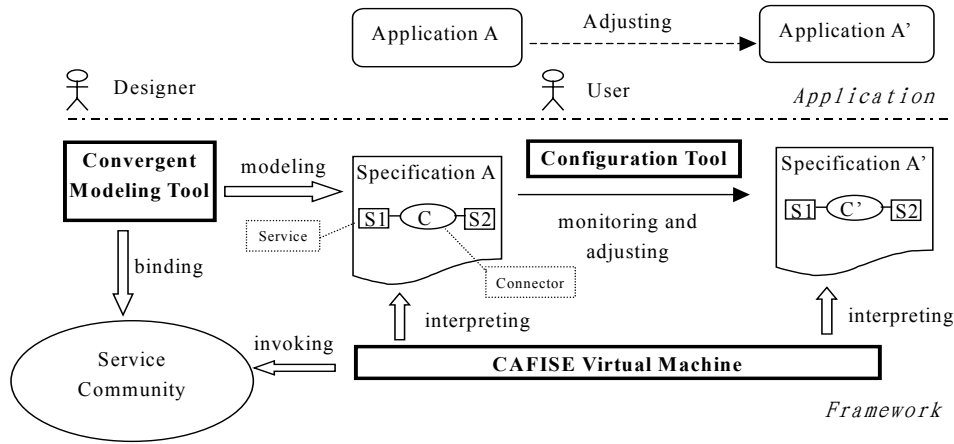


Fig. 3. CAFISE Framework

3.2.2 Reconfiguration of Connectors

The service coordination in an executable specification adopts role-based service connectors. When the aforementioned changes take place, users can reconfigure the role-based connector with the Configuration Tool. More details about the framework are presented in [12]. The following presents an example of a connector implementation and describe how to use this framework to adjust the connector. Figure 4 illustrates a role named *Ticket_querist* in XML, which is used to connect *match query service* with *match ticket query services* according to the interaction protocol listed in section 2.2. Its feature “*searcher*” describes the ports of the *Match Ticket Query Service01* and *Match Ticket Query Service 02* that are listed as feature implementations by *<Services>*. *<Selector>* specifies the algorithm for selecting the right feature implementation. In figure 4, it uses another service “*MTQSel01*” as service selection function to select the right service involved in the interaction.

```

<Role Name="Ticket querist " >
  <Feature Name="searcher">
    <Argument Name="Match"
      Type="NameString"
      PassMode="IN"/>
    <Argument Name="Time"
      Type="Timetype"
      PassMode="IN"/>
    <Argument Name="MatchTicket"
      Type="TicketVector"
      PassMode="OUT"/>
  </Feature>
  <Services>
    <DefaultURL>
      www.FLAME08app.com/search/
      Match Ticket Query Service 01
    </DefaultURL>
    <URL>
      www.FLAME08app.com/search/
      Match Ticket Query Service 02
    </URL>
  </Services>
  <Selector>
    <SelectFunction>
      www.FLAME08app.com/select/
      MTQSel01
    </SelectFunction>
  </Selector>
</Role>

```

Fig. 4. The Role Specification of Ticket_querist

```

<Role Name="Ticket querist " >
  <Feature Name="searcher">
    <Argument Name="Match"
      Type="NameString"
      PassMode="IN"/>
    <Argument Name="Time"
      Type="Timetype"
      PassMode="IN"/>
    <Argument Name="MatchTicket"
      Type="TicketVector"
      PassMode="OUT"/>
  </Feature>
  <Services>
    <DefaultURL>
      www.FLAME08app.com/search/
      Match Ticket Query Service 01
    </DefaultURL>
    <URL>
      www.FLAME08app.com/search/
      Match Ticket Query Service 02
    </URL>
    <URL>
      www.FLAME08app.com/search/
      Match Ticket Query Service New
    </URL>
  </Services>
  <Selector>
    <SelectFunction>
      www.FLAME08app.com/select
      /MTQSel02
    </SelectFunction>
  </Selector>
</Role>

```

Fig. 5. The Modified Specification of Ticket_querist

To adapt to **Change I**, a new service *Match Ticket Query Service New* was designed and registered in the Service Community. The connection was to be changed. The CAFISE Virtual Machine, its

Configuration Tool and role *Ticket_querist* made it easier to reconfigure the service connection. The CAFISE Virtual Machine interpreted the specification of role *Ticket_querist* and collected meta-data of interaction states and the connection structure, supporting the Configuration Tool to modify the connection. The Configuration Tool monitored all interaction states through meta-data collected by the virtual machine. While the number of service requests was reduced to zero, users applied the tool to change the connection meta-data. By modifying meta-data, a new service “*www.FLAME08app.com/search/ Match Ticket Query Service New*” was added, and *<SelectFuction>* was set as a new one “*www.FLAME08app.com/select/MTQSel02*”. Thus, the new service co-existed with the old ones, and the connection was changed without effects to the *Match Query Service* in a simple reconfiguration way. The modified specification of *Ticket_querist* is shown in figure 5.

Figure 6 presents the role connected match ticket service and gym query services that are the service providers in interaction. With **Change II**, *Match Ticket Service* changed to connect with *Order Result Query Service*. Adaptation of this connection required adjusting the role’s *<Feature>*, *<Services>* and *<Selector>* parts. Using CAFISE Framework, users smoothly changed the specification of *Gym_querist* to a new one that is presented in figure 7.

```

<Role Name="Gym querist" >
  <Feature Name="searcher">
    <Argument Name="Match"
      Type="NameString"
      PassMode="IN"/>
    <Argument Name="Time"
      Type="Timetype"
      PassMode="IN"/>
    <Argument Name="Gym_info"
      Type="GymVector"
      PassMode="OUT"/>
  </Feature>
  <Services>
    <DefaultURL>
      www.FLAME08app.com/search/
      Gym Query Service 01
    </DefaultURL>
    <URL>
      www.FLAME08app.com/search/
      Gym Query Service 02
    </URL>
  </Services>
  <Selector>
    <SelectFunction>
      www.FLAME08app.com/select
      Gym selector01
    </SelectFunction>
  </Selector>
</Role>

```

Fig. 6. The Role Specification of Gym_querist

```

<Role Name="Gym querist" >
  <Feature Name="result searcher">
    <Argument Name="Order Number"
      Type="String"
      PassMode="IN"/>
    <Argument Name="Result info"
      Type="ResultVector"
      PassMode="OUT"/>
  </Feature>
  <Services>
    <DefaultURL>
      www.FLAME08app.com/search/
      Order Result Query Service 01
    </DefaultURL>
    <URL>
      www.FLAME08app.com/search/
      Order Result Query Service 02
    </URL>
  </Services>
  <Selector>
    <SelectFunction>
      www.FLAME08app.com/select
      Order resultSel
    </SelectFunction>
  </Selector>
</Role>

```

Fig. 7. The Modified Specification of Gym_querist

The above case study demonstrates that a role-based service connector enables service connection adaptation smoothly. Moreover it shows that *<Feature>* and *<Services>* separates services from features and makes it feasible to modify the interaction interface and its implementation independantly. By changing the *<SelectFunction>*, the relation between interaction interface and services can be adjusted.

Table 1 lists the comparison of service connection adaptations in those two prototypes of the case. It says that these connections can be adjusted, and the adjustments all impact application behaviors with effects to semantics. However, adaptations of control flow based service connections and service adaptors involve more efforts of executants obviously. Role-based service connector can be reconfigured at runtime. And changes of the connection are incremental, which means that a new service can be incorporated into the service composition while old ones co-exist [13]. With the CAFISE Framework, the adaptation process is semi-automatic, and change impacts can be controlled. The comparison shows that the role-based service connector model provides more support to connection adaptation.

Table 1. Properties of Service Connection Adaptations in the Case

	Control flow based service connections	Service adaptors	Role-based service connectors
Adaptation executants	programmers	programmers	users
Adaptation way	by modifying source codes	by modifying source codes or customizing	by reconfiguring
Degree of automation	non-automation	non-automation	semi-automation
Adaptation time	at non-runtime	at non-runtime	at both runtime and non-runtime
Incremental changes	no	no	yes
Effects to semantics	yes	yes	yes
Change impacts control	no	no	yes

3.3 Evaluation Based on the Case Study

The case study demonstrates how to use a role-based service connector model to make service connections adaptable. In addition, it also shows that a role-based service connector has advantages in improving connection adaptability at the following aspects:

- **Communication Stability**

Communication is the essential function of a role-based service connector, which takes charge of data exchange between services involved in an interaction. Through features, roles expose an interaction interface, and allow requestors to invoke services. Role features offer a unified interface for service interaction, improving communication stability from the view of connection structure.

- **Structure Expansibility**

A role-based service connector is extensible in structure aspect. Service resources and user requirements are various and mutable. Unavoidably, service connections have to co-evolve with changes. A role-based service connector provides an extensible cadre composed of *<Feature>*, *<Services>* and *<Selector>*, which enables the connector to be extended and reconfigured according to changes.

- **Connection Adaptability**

The role-based service connector model provides essential support to connection adaptation. With the extensible cadre, it can be modified and reconfigured. Besides that, it can accommodate changes of connection to some extent through encapsulating changes in service providers. And the connection can be adapted at run time, for role-based service connector can dynamically switch service provider in the way of modifying parameters to change connection structure at run time.

4 Conclusions

Service-oriented application development in network environments meets large challenges due to open and dynamic features of the environments, which requires service connections to be adaptable.

In this paper, a role-based service connector model is presented to solve the problem. With role features, a role-based service connector offers a changeable service connection structure, which makes

connections more adaptable: by modifying the feature and related service references, the connection can be reconfigured. In addition, adjustments in a role-based connector are limited to some modifications; changes of interaction partners do not influence each other. So that, the connector enhances the flexibility of service coordination.

Through the case study of project FLAME2008, we conclude that the role-based service connector model has advantages in adaptation of service connection. Besides that, the following work should be done.

- To reduce side effects of adaptation, a run-time model of the connector should be offered to monitoring status of service connection;
- The application of the model in Web service chaining, such as BPEL4WS, is to be considered in further work.

Acknowledgements

When we wrote the paper, Dr. Agnes Voisard gave some good suggestions; Dipl.-Inf. Norbert Weissenberg corrected the writings. And Dipl.-Inf. Rüdiger Gartmann gave generous helps on paper presentation. We are grateful to them for their helps.

References

1. I. Foster, C. Kesselman, J. Nick, S. Tuecke: Grid Services For Distributed System Integration. *Computer*. vol. 35, no.6 (2002) 37-46
2. I. Foster, C. Kesselman, S. Tuecke: The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of Supercomputer Applications*. vol.15 no.3(2001) 200-222
3. F. Casati, S. Inicki, J. LiJie, S. Ming-Chien: An Open, Flexible, and Configurable System for E-Service Composition. *The Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, Milpitas, USA, June 2000
4. E. Kiciman, A. Fox: Using Dynamic Mediation to Integrate COTS Entities in a Ubiquitous Computing Environment. *The Second International Symposium on Handheld and Ubiquitous Computing*, Bristol, UK, September 2000
5. Emre Kiciman etc: Position Summary: Towards Zero-code Service Composition. *The Eighth Workshop in Hot Topics in Operating Systems*, Oberbayern, Germany, May 2001
6. S. Krishnan¹, P. Wagstrom¹, G. Laszewski: GSFL: A Workflow Framework for Grid Services. <http://www-unix.globus.org/cog/projects/workflow/>, July 2002
7. T. Andrews, F. Curbera etc.: Business Process Execution Language for Web Services Version 1.1. <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>, May 2003
8. F. Steimann: On the Representation of Roles in Object-Oriented and Conceptual Modeling. *Data & Knowledge Engineering*. vol. 35, no.1(2000)83-106
9. B. Kristensen: Object-oriented Modeling with Roles. *The 2nd International Conference on Object-Oriented Information Systems*, Dublin, Ireland, 1995
10. J. Goguen: A Categorical Manifesto. *Mathematical Structures in Computer Science*. vol. 1, no. 1(1991)49-67
11. Cervesato etc.: A Meta-notation for Protocol Analysis. *The 12th IEEE Computer Security Foundations Workshop*, Mordano, Italy, June 1999

12. Y. Han, Z. Zhao, G. Li etc.: CAFISE: An Approach to Enabling Adaptive Service Configuration of Service Grid Applications. *Journal of Computer Science and Technology*. vol. 18, no.4(2003) 484-494
13. G. Li: Adaptive software architecture and Adaptive software architecture development. Ph.D. Dissertation, Beijing University of Aeronautics and Astronautics. 2002

Building an integrated Ontology within SEWASIE system^{*}

Domenico Beneventano^{1,2}, Sonia Bergamaschi^{1,2}, Francesco Guerra¹, and
Maurizio Vincini¹

¹ Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905 - Modena
{lastname.firstname}@unimo.it

² IEIIT-CNR Istituto di Elettronica e di Ingegneria
dell'Informazione e delle Telecomunicazioni
Viale Risorgimento 2 - Bologna

Abstract. The SEWASIE (SEmantic Webs and AgentS in Integrated Economies) project (IST-2001-34825) is an European research project that aims at designing and implementing an advanced search engine enabling intelligent access to heterogeneous data sources on the web.

In this paper we focus on the Ontology Builder component of the SEWASIE system, that is a framework for information extraction and integration of heterogeneous structured and semi-structured information sources, built upon the MOMIS (Mediator envirOnment for Multiple Information Sources) system. The result of the integration process is a Global Virtual View (in short GVV) which is a set of (global) classes that represent the information contained in the sources being used. In particular, we present the application of our integration concerning a specific type of source (i.e. web documents), and show the extension of a built-up GVV by the addition of another source.

Introduction

Nowadays the Web is a huge collection of data and its expansion rate is very high. Web users need new ways to exploit all this available information and possibilities. The problem is that Web information is meaningless for a computer and so it is very hard to find out what we are looking for. In this context, the need of *a new vision of the Web*, the **Semantic Web**³, arises. Within the Semantic Web, resources could be annotated with machine-processable metadata providing them with background knowledge and meaning. This new scenario creates many expectations amongst the users and information providers but new issues have to be solved before achieving optimum results. One of the main components in this context is the **ontology**; this “*explicit specification of a*

^{*} This research has been partially supported by EU IST-SEWASIE

³ <http://www.w3.org/2001/sw/>

conceptualization” [11] might allow information providers to give a shared meaning to their documents. Many studies are defining *languages* and *standards* that can help domain experts in the delicate task of expressing their knowledge in a formal way⁴. Another fundamental issue is the “*dynamics*”. The web environment is very changeable, it is continuously updated, modified and the users need to rely on the data they retrieve from the net. Ontologies evolve, and therefore we have to address the problem of managing the dynamics with respect to the ontologies [13, 14].

SEWASIE (SEmantic Webs and AgentS in Integrated Economies) (IST-2001-34825) is a research project funded by EU on the action line “Semantic Web” (May 2002/April 2005 - <http://www.sewasie.org/>). The goal of the SEWASIE project is to design and implement an advanced search engine enabling intelligent access to heterogeneous data sources on the web via semantic enrichment to provide the basis of structured secure web-based communication. A SEWASIE user has at his disposal a search client with an easy-to-use query interface able to extract the required information from the Internet and to show it in an easily readable format.

In this paper we focus on the Ontology Builder component of the SEWASIE system, that is a framework for information extraction and integration of heterogeneous structured and semi-structured information sources, built upon the MOMIS (Mediator envirOnment for Multiple Information Sources) system [1, 2, 6].

The Ontology Builder implements a semi-automatic methodology for data integration that follows the *Global as View* (GAV) approach [15]. The result of the integration process is a global schema which provides a reconciled, integrated and virtual view of the underlying sources, the GVV (Global Virtual View). The GVV is composed of a set of (global) classes that represent the information contained in the sources being used and the mappings establishing the connection between the elements of the global schema and those of the source schemata. A GVV, thus, may be thought of as a domain ontology [12] for the integrated sources. We represent the ontology by means an object language, called ODL_{J3} , which is an evolution of the OODBMS standard language ODL. Moreover, ODL_{J3} permits the definition of integrity constraints (in the form of *if then* rules) that are translated, together with the schema properties, into a description logics OLCD (Object Language with Complements allowing Descriptive cycles) [4, 6]. In this way, inference tasks typical of Description Logics that are useful for the GVV creation process can be exploited. The Ontology Builder system relies on a logic layer, ODL_{J3} is the language to represent the ontology properties and OLCD to perform reasoning over the data, like other approaches in the literature (DAML+OIL⁵).

The outline of the paper is the following: section 1 describes the SEWASIE architecture, while in section 2 we depict the Ontology Builder and the approach

⁴ OntoWeb - Ontology-based information exchange for knowledge management and electronic commerce, <http://ontoweb.aifb.uni-karlsruhe.de/>

⁵ <http://www.w3.org/TR/daml+oil-reference>

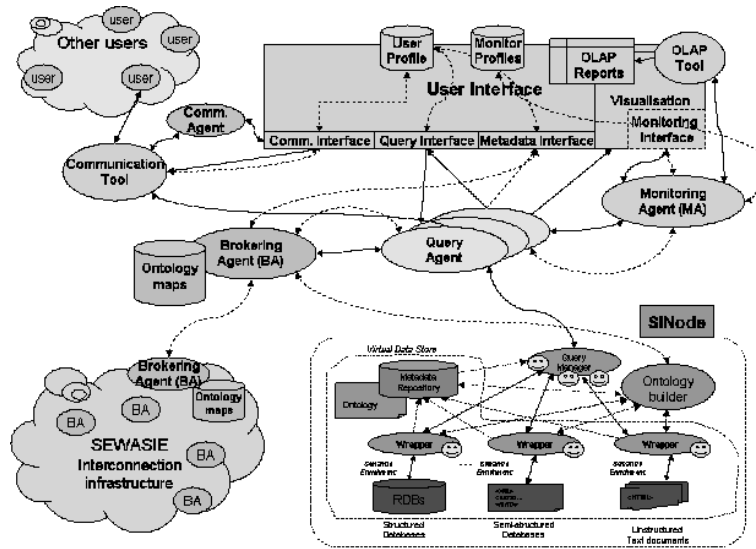


Fig. 1. The SEWASIE architecture

for creating a domain ontology from scratch and shows the result of the integration process (GVV). Section 3 describes the semi-automatic annotation process of the GVV. Section 4 presents the methodology to support the GVV extensions in the case of the addition of a new source. Finally, section 5 concludes the paper.

1 The SEWASIE Architecture

The first basic idea underlying the SEWASIE architecture is that *semantic enrichment of data sources is the next step towards building information systems that are really useful*. However, the addition of semantics to data sources is a formidable task and it may be achieved only if info seekers and info providers may reach each other across a middle ground. This requires a *common* language and strategy, and the tools that actually flesh them both out.

The second idea is that *we have to deal with two levels of knowledge*. We envision a multi-level architecture, composed of nodes (the SINodes) integrating information coming from communities with strong ties, and at a wider level the relationships among distinct SINodes are established by means of weaker semantics mappings. The latter is maintained by an infrastructure of *brokers*, which will provide the entry points to the system and some routing of the queries towards the relevant information nodes.

A search system architecture satisfying the aforementioned ideas and desiderata is shown in figure 1.

The **information nodes** (SINodes) are mediator-based systems, each including a Virtual Data Store, an Ontology Builder, and a Query Manager. A Virtual Data Store represents a virtual view of the overall information managed within any SINode and consists of the managed information sources, wrappers, and a metadata repository. The managed Information Sources are heterogeneous collections of structured, semi-structured, or unstructured data, e.g. relational databases, XML or HTML documents. A Wrapper implements common communication protocols and translates to and from local access languages. According to the metadata provided by the wrappers, the Ontology Builder performs semantic enrichment processes in order to create and maintain the Ontology of the SINode. The Metadata Repository holds the ontology and the knowledge required to establish semantic inter-relationships between the SINode itself and the neighboring ones. A Query Manager provides the functionalities for solving a query within an SINode and constitutes the SINode interface to the network.

The **brokering agents (BAs)** are the peers responsible for maintaining a view of the knowledge handled by the network, as well as the information on the specific content of SINodes which are under direct control (of each brokering agent). These agents are intermediaries which have direct control over a number of SINodes, and provide the means to publish a manifesto within the network of the locally held information with a semantic profile.

The **query agents (QAs)** are the carriers of the user query from the user interface to the SINodes, and have the task of solving a query by interacting with the brokering agent network. Starting from a user- or task- specified brokering agent, they may access other BAs, connect with other information nodes, collect partial answers, and integrate them.

The **user interface** is the group of modules which work together to offer an integrated user interaction with the semantic search system. This interface needs to be personalized and configured with the specific user profile and a reference to the ontologies which are commonly used by this user.

2 The Ontology Builder

The process of semantic enrichment of the sources constituting a SINode is a crucial step towards building the overall SEWASIE structure. The process is human assisted and based on a tool, the Ontology Builder. The underlying strategy and framework are based on ODL_{I^3} , the ontology description language, and basic lexical ontologies to bootstrap. The final result is a Global Virtual View encompassing all the sources within the SINode.

In this section, we describe the information integration process for building the GVV of set of web pages (see Figure2 for the whole process representation).

2.1 ODL_{I^3} + OLCD

For a semantically rich representation of source schemas and object patterns, the Ontology Builder uses an object-oriented language called ODL_{I^3} [6]. ODL_{I^3} is

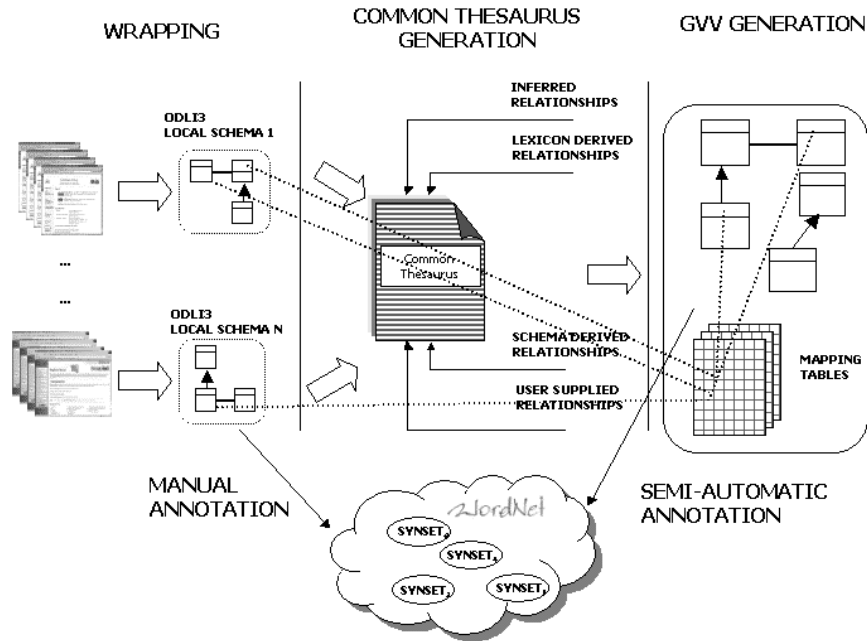


Fig. 2. An overview of the ontology integration process

an extension of the ODL language⁶ and can be used to describe heterogeneous schemas of structured and semistructured data sources. In particular, ODL_{I3} extends ODL with the following relationships expressing intra- and inter-schema knowledge for the source schemas:

- SYN (synonym of) is a relationship defined between two terms t_i and t_j that are synonyms in every involved source.
- BT (broader terms) is a relationship defined between two terms t_i and t_j , where t_i has a broader, more general meaning than t_j . BT relationships are not symmetric. The opposite of BT is NT (narrower terms).
- RT (related terms) is a relationship defined between two terms t_i and t_j that are generally used together in the same context in the considered sources.

Other main additions are the *Integrity constraint rules*, introduced in ODL_{I3} in order to express, in a declarative way, *if then* constraint rules at both intra- and inter-source level.

By means of ODL_{I3} it is possible to describe both the sources (the input of the synthesis process) and the GVV (the result of the process) by using the same language.

⁶ http://www.service-architecture.com/database/articles/odmg_3.0.html

Due to the fact that the ontology is composed of concepts (represented in ODL_{I^3} with Global Classes) and simple binary relationships, the translation of ODL_{I^3} descriptions into one of the Semantic Web standards such as RDF, DAML+OIL, OWL is a straightforward process. In fact, from a general perspective an ODL_{I^3} concept corresponds to a *Class* of a the Semantic Web standard, and ODL_{I^3} relationships are translated into *properties* (in particular the BT/NT ODL_{I^3} relationships are *subclassof* in the Semantic Web standards). Analyzing syntax and semantics of each standard, further specific correspondences may be established. For example, there is the correspondence with the DAML+OIL *Class*, the simple domain attributes correspond to DAML+OIL *DataTypeProperty* concept and complex domain attributes correspond to DAML+OIL *ObjectProperty* concept. Moreover, classes are wrapped in both the approaches into description logics. For a more detailed description of ODL_{I^3} /OLCD translation see [6]. For a description of the OLCD description logics see [4, 3]

2.2 Wrapping: extracting data structure for sources

The first step of the ontology development process is the construction of a semantic representation of the information sources, i.e. the conceptual schema of the sources, by means of the common data language ODL_{I^3} . To accomplish this task, we encapsulate each source with a wrapper that logically converts the underlying data structure into the ODL_{I^3} information model. Therefore, the wrapper architecture and interfaces are crucial, because wrappers are the focal point for managing the diversity of data sources.

For conventional structured information sources (e.g. relational databases, object-oriented databases), a schema description is always available and can be directly translated.

For semistructured information sources, a schema description is in general not directly available at the sources. In fact, a basic characteristic of semistructured data is that they are "self-describing", hence the information associated with the schema is specified within data. Thus, in order to manage a semi-structured source a specific wrapper has to implement a (semi-) automatic methodology to extract and explicitly represent the conceptual schema of the source. We developed a wrapper for XML/DTDs files.

Information is available on the Web mainly in HTML format that is human-readable but cannot easily be automatically accessed and manipulated. In particular, HTML language does not separate data structure from layout. Thus, in order to manage these kind of sources, we need a further preliminary step of extraction: by means of a commercial tool we translate the content of a web page (data and data structure) into a XML file, then we exploit the previously developed wrapper XML/DTD to acquire the source descriptions.

We have tested many research and commercial tools, such as Lixto [10], RoadRunner [8], Andes [16], and we select Lixto as the most suitable for our approach. By providing a fully visual and interactive user interface, Lixto assists the user to create a wrapper program in a semi-automatic way. Once the wrapper is built, it can be applied automatically to continually extract relevant

information from a permanently changing web page and translate it into a XML file to be exploited by the XML/DTD wrapper.

2.3 Running example

We consider the creation of an ontology of two web sources related to the University domain. By means of a Lixto generated wrapper, the source content is translated into XML files according to the DTDs sketched in Table 1.

University Site (UNI)	Computer Science Site (CS)
<pre> <!ELEMENT UNI(People*)> <!ELEMENT People(Research_Staff* School_Member*)> ... <!ELEMENT Research_Staff(name, e-mail, Section*, Article*)> <!ELEMENT Section(name, year. period)> <!ELEMENT Article(title, year, journal, conference)> <!ELEMENT School_Member(name, e-mail)> <!ELEMENT name (#pcdata)> ... </pre>	<pre> <!ELEMENT CS(Person*)> ... <!ELEMENT Person(Professor* Student*)> <!ELEMENT Professor(first_name, last_name, e-mail, Publication*)> <!ELEMENT Student(name, e-mail)> <!ELEMENT Course(denomination, Professor)> <!ELEMENT Publication(title, year, journal, editor)> <!ELEMENT School_Member(name, e-mail)> <!ELEMENT name (#pcdata)>... </pre>

Table 1. A fragment of the University (UNI) and Computer Science (CS) DTDs

By means of the XML/DTD wrapper, the obtained DTDs are translated into ODL_{I3} descriptions. An example of the classes obtained in this step is shown in Table 2.

2.4 Annotation of a local source with WordNet

With reference to the Semantic Web area, where generally the annotation process consists of providing a web page with semantic markups w.r.t. an ontology, in our approach we markup the metadata descriptions extracted by the wrappers, i.e. the ODL_{I3} schemata, and the reference lexical ontology is WordNet.

The WordNet database contains 146,350 lemma organized in 111,223 synonym sets. WordNet's starting point for lexical semantics comes from a conventional association between the forms of the words – that is, the way in which words are pronounced or written – and the concept or meaning they express. These associations give rise to several properties, including synonymy, polysemy, and so forth. The correspondence between the words form and their meaning is

University Site (UNI) ... Interface Research_Staff (Source Un_site.dtd) { attribute string name; attribute string email; attribute set<Section> section; attribute set<Article> article; } Interface Article (Source Un_site.dtd) { attribute string title; attribute string journal; attribute string conference; attribute string year; } ...	Computer Science Site (CS) ... Interface Professor (Source Sc_site.dtd) { attribute string first_name; attribute string last_name; attribute string email; attribute set<Publication> publication; } Interface Publication (Source Sc_site.dtd) { attribute string title; attribute string year; attribute string journal; } ...
---	--

Table 2. A piece of the University (UNI) and Computer Science (CS) sources in ODL_{T3}

represented in the so-called Lexical Matrix M (see table 3), in which the words meaning are reported in rows (hence each row represents a synset) and columns represent the words form (form/base lemma).

	WF1	WF2	WF3	...	WF _n
M1	E1,1	E1,2			
M2		E2,2			
M3			E3,3		
...				...	
M _m					E _{m,n}

Table 3. WordNet word form and meanings

Thus, entry E1,1 implies that word form F1 can be used to express word meaning M1. If there are at least two entries in the same column then the corresponding word form is polysemous (i.e. it can be used to represent more than one meaning, exactly two in this case); if there are at least two entries in the same row then two word forms are synonyms relative to a context.

Given a word form F, its *i*-th meaning will be denoted by F#*i*. For example, the word form `course` has 8 meanings in WordNet; the first one is `course#1 = "education imparted in a series of lessons or class meetings"`.

In the phase of a local source annotation, the integration designer has to manually choose the appropriate WordNet meaning for each element of the conceptual schema provided by the wrappers. The annotation phase is composed of two different steps:

1. **Word Form choice.** In this step, the WordNet morphologic processor aids the designer by suggesting a word form corresponding to the given term. More precisely, the morphologic processor stems (i.e. converts to a common root form) the term and checks if it exists as word form.
2. **Meaning choice.** The designer can choose to map an element on zero, one or more senses. Notice that the user can only choose a sense among the existing ones in WordNet, and that is he is not allowed to extend it with his new meanings.

Notice that, for a compound descriptive term, our tool extracts the component terms and all these terms are processed by the WordNet morphologic processor. For example if the attribute name is `shipment_received_date` then the terms `shipment`, `received`, and `date` are proposed to the designer. If a term is not available as word form (this can happen, for example, for an abbreviation), if there is an ambiguity, or the selected word form is not satisfactory, the designer can choose another word form of WordNet or manually search for a meaning of the term. A term that doesn't find a meaning within WordNet is considered as unknown term and no lexicon relationship will be derived for it (see next section).

This phase assigns a name, *LEN* (this name can be the original one or a word form chosen from the designer), and a set (that might be empty) of meanings, *LEM_i* (a class or attribute meaning is given by the disjunction of its set of meanings), to each local element (class or attribute) *LE* of the local schema:

$$LE = \langle LEN, \{LEM_1, \dots, LEM_k\} \rangle, k \geq 0$$

For example:

```

CS.Course           = < course, {course#1} >
UNI.Professor       = < professor, {professor#1} >
UNI.School_Member  = < student, {student#1} >
UNI.School_Member.name = < name, {name#1} >

```

where

```

course#1 = 'education imparted in a series of lessons or class meetings'
professor#1 = 'someone who is a member of the faculty at a college or university'
student#1 = 'a learner who is enrolled in an educational institution'
name#1 = 'a language unit by which a person or thing is known'

```

2.5 Common Thesaurus Generation

The Ontology Builder constructs a Common Thesaurus describing intra and inter-schema knowledge in the form of relationships SYN, BT, NT, and RT.

The Common Thesaurus is constructed through an incremental process in which relationships are added in the following order:

1. *schema-derived relationships*: relationships holding at intra-schema level extracted by analyzing each schema separately;

2. *lexicon-derived relationships*: These originate from the annotation of the schemas respect the lexical ontology. WordNet defines a large variety of semantic relations between its meanings. A lexicon relationship between terms for the common thesaurus is derived from a semantic relation in WordNet between the annotated meanings of the terms according to the following correspondences:

Synonymy: corresponds to a SYN relation
Hypernymy: corresponds to a BT relation
Hyponymy: corresponds to a NT relation
Holonomy: corresponds to a RT relation
Meronymy: corresponds to a RT relation
Correlation: corresponds to a RT relation

3. *designer-supplied relationships*: new relationships can be supplied directly by the designer, to capture specific domain knowledge. This is a crucial operation, because the new relationships are forced to belong to the Common Thesaurus. This means that, if a nonsense or wrong relationship is inserted, the subsequent integration process can produce a wrong global schema;
4. *inferred relationships*: Description Logics techniques of ODB-Tools [5] are exploited to infer new relationships, by means of subsumption computation applied to a “virtual schema” obtained by interpreting BT/NT as subclass relationships and RT as domain attributes.

In our running example, some of the relationships automatically obtained and proposed at the integration designer are the following:

```

schema derived:  CS.Professor NT CS.Person
schema derived:  CS.Student NT CS.Person
lexicon derived: UNI.School_Member NT CS.Person
lexicon derived: UNI.Article NT CS.Publication
designer-supplied: UNI.Research_Staff SYN CS.Professor
inferred:  UNI.Research_Staff NT CS.Person
inferred:  UNI.Research_Staff RT UNI.Article
  
```

If the designer accepts and confirms the above relationships, they are included in the Common Thesaurus.

2.6 Global Virtual View generation

The proposed methodology allows us to identify similar ODL_{J3} classes, that is, classes that describe the same or semantically related concept in different sources. To this end, *affinity coefficients* (i.e., numerical values in the range $[0, 1]$) are evaluated for all possible pairs of ODL_{J3} classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (*Name*

Affinity coefficient) and their attributes (*Structural Affinity* coefficient) and are fused into the *Global Affinity* coefficient, calculated by means of the linear combination of the two coefficients. For a detailed description of the affinity coefficient evaluation, the reader can refer to [7]. Global affinity coefficients are then used by a hierarchical clustering algorithm [9], to classify ODL_{I3} classes according to their degree of affinity. The output of the clustering procedure is an affinity tree, where ODL_{I3} classes are the leaves and intermediate nodes have an associated affinity value, holding for the classes in the corresponding cluster. Clusters for integration (candidate clusters) are interactively selected from the affinity tree using a threshold based mechanism whose parameter are set by the designer. Regarding the quality of our clustering results the reader can refer to [6] where a deep discussion of the experimentations results of the use of the strengthened terminological relationships and affinity-based clustering is reported.

The generation of **Global Classes** out of selected clusters is a synthesis activity performed interactively with the designer: a Global Class GC_i definition is built for each cluster Cl_i . The GVV generation consists of two phases. First, the system automatically associates a set of global attributes with GC_i , corresponding to the union of local attributes of the classes belonging to Cl_i . Then, the system proposes to the designer the restriction of the global attributes set by exploiting the Common Thesaurus lattice that contains SYN relationships and BT/NT relationships among local attributes.

For each global class, a persistent **Mapping Table** MT storing all the mappings is generated; it is a table whose columns represent the set of local classes which belong to the cluster and whose rows represent the global attributes. An element $MT[GA][LC]$ represents the set of attributes of the local class LC which are mapped into the global attribute GA : the value of the GA attribute is a function of the values assumed by the set of attributes $MT[GA][LC]$. Some simple and frequent cases of such function are the following:

- *identity*: the GA value is equal to the LA value; we denote this case as $MT[GA][LC] = LA$
- *conjunction*: the GA value is obtained as a conjunction of the values assumed by a set of local attributes LA_i of the local class LC ; we denote this case as $MT[GA][LC] = LA_1 \text{ and } \dots \text{ and } LA_n$
- *constant*: GA assumes into the local class LC a constant value set by the designer; we denote this case by $MT[GA][L] = \text{const}$
- *undefined*: GA is a set undefined into the local class LC ; we denote this case as $MT[GA][L] = \text{null}$.

In our running example the integration process gives rise to three global classes:

Global1: (UNI.Section, CS.Course)

Global2: (UNI.Article, CS.Publication)

Global3: (UNI.Research.Staff, UNI.School.Member, CS.Professor, CS.Student)

For each global class a Mapping Table is generated. For example the Mapping Table for Global2 is:

	UNI.Article	CS.Publication
Title	Title	Title
Year	Year	Year
Journal	Journal	Journal
Conference	Conference	null
Editor	null	Editor

Table 4. Mapping Table of the global class Global2 (Publication)

3 Global Virtual View Annotation

In this section, we propose a semi-automatic methodology to annotate a GVV, i.e. to assign a name, GEN , and a set (that might be empty) of meanings, GEM_i (a class or attribute meaning is given by the disjunction of its set of meanings) to each global element (class or attribute) GE :

$$GE = \langle GEN, \{GEM_1, \dots, GEM_p\} \rangle, p \geq 0$$

3.1 Global Class Annotation

In order to semi-automatically associate an annotation to each global class, we consider the set of all its “broadest” local classes, w.r.t. the relationships included in the Common Thesaurus, denoted by GC_B :

$$GC_B = \{LC \in GC \mid \neg \exists y \in GC, (LC \text{ NT } y) \vee (y \text{ BT } LC)\}$$

In our example:

	GC	GC_B
GC ₁	CS.Course, UNI.Section	CS.Course, UNI.Section
GC ₂	CS.Publication, UNI.Article	CS.Publication
GC ₃	CS.Professor, CS.Person, UNI.School_Member, UNI.Research_Staff, CS.Student	CS.Person

On the basis of GC_B , the designer will annotate the global class GC as follows:

- **name choice:** the integration designer is responsible for the choice of the GC name: the system only suggests a list of possible names. The designer may select a name within the proposed list or select another name not inside the list. In particular, concerning the *name* and according to the role of the global class name (to allow the designer to identify the Global Class and its contents), we consider the name as a label. Therefore, a name might not be a word form of WordNet. For example, regarding Global Class GC1 (see Table 5), the designer selected the name *course* between the suggested *Course* and *Section*. Regarding GC3 the designer chose a more significative name (*University_Member*) instead of the proposed generic *person*.

- **meaning choice**: the union of the meanings of the local class names in GCB are proposed to the designer as meanings of the Global Class. The designer may change this set, by removing some meanings or by adding other ones.

With respect to our example, the proposed annotations are the following:

GC	Names	Meanings
GC1	course or section	course#1
GC2	publication	publication#1
GC3	University_Member	person#1

Table 5. University GVV annotation

3.2 Global Attributes Annotation

We extend the previously used approach for names and meanings of the attributes. Given a global attribute GA of the global class GC , we consider the set LGA of local attributes, which are mapped into GA :

$$LGA = \{LA | \exists LC \in GC, LA \in LC \wedge MT[GA][LA] \neq \text{null}\}$$

and the set of all its “broadest” local attributes, denoted by LGA_B :

$$LGA_B = \{LA \in LGA | \neg \exists y \in LGA, (LA \text{ NT } y) \vee (y \text{ BT } LA)\}$$

On the basis of LGA_B , the designer will annotate the global attribute as described for global classes. Moreover, according to mapping function, we may develop some specific policy to automatically select meanings.

4 Adding a new source

Supporting the evolution of an ontology represents a challenging issue (to be faced). Many interesting solutions have been developed with regard to this topic [13, 14] and an outstanding idea is to exploit multiple variants of the same ontology to cope with changes. This approach, called *Ontology Versioning*, is different from our proposal where a single ontology is kept consistent with the sources which refer to.

Within Ontology Builder if new sources are added/deleted, or if some changes occur in the sources, the corresponding GVV has to change. The integration process is expensive both for the designer and for the system. For this reason, we propose a methodology for integrating a new source, which exploits the previous integration work, i.e., a built-up GVV, without restarting the integration process from scratch.

In the GVV building approach all the sources to be integrated contribute with the same weight to the process. Therefore, if we consider an already built GVV and we have to insert a new source which refers to the same ontology, we can assume that this source brings less semantics than the GVV itself. For this reason, we devise an integration process of a new source that starts from the obtained GVV and tries to integrate a new source in the GVV.

In the following, we show how the evolution of a GVV caused by the insertion of a new source can be strongly simplified by having available the lexicon-based knowledge of the GVV annotation.

4.1 Integration of a new source in a GVV

The insertion of a new source is managed as an integration process between two schemata: the GVV and the new source schema; in other words, the global classes of GVV are considered as local classes and are integrated with the local classes of the new source.

We show the approach analyzing all the integration phases of the GVV with the new source. We introduce the following notation:

gcNew the global class of the new integrated schema has a name, **gcNewName** and a set of global attributes **gcNewAtt_i**,

gcOld the global class of the old integrated schema has a name, **gcOldName** and a set of global attributes **gcOldAtt_j**,

lcNew the local class of the new source has a name, **lcNewName** and a set of local attributes **lcNewAtt_k**.

According to the integration methodology, we have to create a Common Thesaurus of the involved sources. In this case, the Common Thesaurus will contain schema-derived relationships extracted from the analysis of the new source and intra-schema lexicon-derived relationships obtained by the annotation of the new source. Further, the GVV global classes have to be semantically enriched according to the semi-automatic annotation method shown in section 3. The interesting point is that the annotation of GVV allows us to discover inter-schema lexical relationships which enrich the Common Thesaurus.

The next step is the cluster generation followed by Global Classes and mapping tables generation. This phase has to provide mapping rules among Global Classes and new or old local classes. In order to achieve this result, we substitute here old Global Classes with the respective Local Classes. In this way, new Global Classes that represent old Local Classes and new Local Classes as are built. Thus we have:

$$\mathbf{gcNew} = \{\mathbf{gcOld}_1, \dots, \mathbf{gcOld}_p, \mathbf{lcNew}_1, \dots, \mathbf{lcNew}_n\}$$

the resulting rewriter step is:

$$\mathbf{gcNew} = \{\mathbf{lcOld}_{1_1}, \dots, \mathbf{lcOld}_{1_z}, \dots, \mathbf{lcOld}_{p_1}, \dots, \mathbf{lcOld}_{p_n}, \mathbf{lcNew}_1, \dots, \mathbf{lcNew}_n\}$$

With Global Class generation, we observe that, using the same clustering parameters, an old Global Class $lc_1, \dots, lc_i, \dots, lc_n$ changes only if the integration process inserts one or more new local classes ($lcNew_i$) into the Global Class. Therefore, we observe that the following cases are possible:

a) A new global class $gcNew$ is composed of only one old global class ($gcOld$) and one or more new local classes ($lcNew_i$):

$$gcNew = \{gcOld, lcNew_1, \dots, lcNew_i, \dots, lcNew_n\}$$

The new global class ($gcNew$) may have new global attributes generated from the semantic contribution of new local classes. New mapping rules are defined among a global attribute and its corresponding local attribute(s). In this case, global attributes belonging to the $gcOld$ ($gcOldAtt_i$) may map both local classes of the old Global Class and new local classes (see the columns associated to $lcNew_t$, for example). New global attributes can only map new local Classes (null mappings in the following table).

So we can say that meanings associated to each global attribute are:

- The meaning of old global attributes have to be enriched with the meanings of the new local classes mapped by these attributes;
- The meaning of new global attributes have to be set according to the rules defined before (see 3.2).

	$lcOld_1$...	$lcOld_k$	$lcNew_1$	$lcNew_t$	$lcNew_n$
$gcOldAtt_1$	the same mappings as in $gcOld$			new mappings		
...						
$gcOldAtt_m$						
$gcNewAtt_1$	null mappings					
...						
$gcNewAtt_p$						

Table 6. New mapping table example.

b) A global class of the new integrated schema is composed of only new local classes:

$$gcNew = \{lcNew_1, \dots, lcNew_i, \dots, lcNew_n\}$$

This situation describes the case in which the GVV is extended without interfering with the previous one.

The new global class ($gcNew$) has a name ($gcNewName$) and a set of new global attributes ($gcNewAtt_i$), where each new global attribute maps only new local attributes. The names and meanings of the global attributes are defined following the rules stated before (see 3.2).

c) A global class of the new integrated schema is composed of more than one global class of the GVV and at least one local class of the new source we are integrating.

$$\text{gcNew} = \{\text{gcOld}_1, \dots, \text{gcOld}_p, \text{lcNew}_1, \dots, \text{lcNew}_i, \dots, \text{lcNew}_n\}$$

In this case the previous GVV is modified; side effects can influence the applications based on the previous schema. The new global class (`gcNew`) has a name (`gcNewName`) and a set of new global attributes (`gcNewAtti`).

5 Concluding remarks

In this paper, we presented a methodology for supporting the semi-automatic building, annotation and extension of a domain ontology obtained by integrating web documents with the Ontology Builder component of the SEWASIE System. Talking about the evolution issue and, in particular, the addition of a new source, we had to face two different problems: the system overload to maintain the built ontology corresponding to the involved sources, and, the insertion of a new source that may modify the existing ontology, with a side effect to each application based on the ontology.

We tried to solve both problems and the most relevant advantages of our methodology of integrating a new source into a GVV is that the process is less expensive than starting from scratch and it is done starting from semantically annotated results of previous integration processes. Possible limitations are:

- mistakes of the previous integration process might propagate to the new GVV;
- the new GVV is based on the previous one, and so it might not perfectly represent all the sources.

Acknowledgements

This work is supported in part by the 5th Framework IST program of the European Community through project SEWASIE within the Semantic Web Action Line. The SEWASIE consortium comprises in addition to the authors' organization (Sonia Bergamaschi is the coordinator of the project), the Universities of Aachen RWTH (M. Jarke), Roma La Sapienza (M. Lenzerini, T. Catarci), Bolzano (E. Franconi), as well as IBM Italia, Thinking Networks AG and CNA (Association of SMEs) as user organizations.

References

1. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: The momis project demonstration. In *VLDB 2000, Proc. of 26th International Conference on Very Large Data Bases, 2000, Egypt, 2000*.

2. D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. The MOMIS approach to information integration. In *AAAI International Conference on Enterprise Information Systems (ICEIS 2001)*, 2001.
3. D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10:576–598, July/August 1998.
4. D. Beneventano, S. Bergamaschi, and C. Sartori. Description logics for semantic query optimization in object-oriented database systems. *ACM Transaction on Database Systems*, 28:1–50, 2003.
5. D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. ODB-Tools: A description logics based tool for schema validation and semantic query optimization in object oriented databases. In *Proc. of Int. Conf. on Data Engineering, ICDE'97*, Birmingham, UK, April 1997.
6. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.
7. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Data and Knowledge Engineering*, 13(2), 2001.
8. Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: automatic data extraction from data-intensive web sites. In *SIGMOD Conference*, 2002.
9. B. Everitt. Cluster analysis. *Heinemann Educational Books Ltd*, 1974.
10. R. Baumgartner S. Flesca and G. Gottlob. Visual web information extraction with lixto. In *the 27th International Conference on Very Large Data Bases (VLDB 2001)*. Roma, Italy, September, 2001.
11. T. R. Gruber. *A translation approach to portable ontology specifications.*, volume 5. 1993.
12. N. Guarino. Formal ontologies and information systems. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'98)*, Trento, Italy, june 1998.
13. J. Heflin and J. Hendler. Dynamic Ontologies on the Web. In *In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 443–449. AAAI/MIT Press, 2000.
14. M. Klein and D. Fensel. Ontology Versioning on the Semantic Web. In *First Intl' Semantic Web Working Symposium. 2001*, 2001.
15. M. Lenzerini. Data integration: A theoretical perspective. In Lucian Popa, editor, *Proc. of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 233–246, Madison, Wisconsin, USA, 2002. ACM.
16. Jussi Myllymaki. Effective web data extraction with standard xml technologies. In *WWW*, pages 689–696, 2001.

Ontologies : A contribution to the DL/DB debate.

Nadine Cullot¹, Christine Parent³, Stefano Spaccapietra², and Christelle Vangenot²

¹ LE2I Laboratory, University of Burgundy,
BP 47870, 21078 Dijon Cedex, France
`nadine.cullot@u-bourgogne.fr`,

² Database Laboratory, Swiss Federal Institute of Technology,
CH-1015 Lausanne, Switzerland

`stefano.spaccapietra@epfl.ch, christelle.vangenot@epfl.ch`

³ University of Lausanne, CH-1015 Lausanne, Switzerland
`christine.parent@unil.ch`

Abstract. The move to global economy has emphasized the need for intelligent information sharing, and turned ontologies into a kernel issue for the next generation of semantic information services. The push towards an effective use of ontologies as a means to achieve semantic interoperability is, in our opinion, shifting the focus from purely taxonomic ontologies to more descriptive ontologies. These would namely provide agreed descriptions of the data structures representing the complex organization of objects and links of interest within the targeted domain. This paper analyzes the requirements for such descriptive ontologies, and contrasts the requirements to the functionality provided by some current representative approaches that have been proposed for ontology management. Selected approaches originate from research in artificial intelligence, knowledge representation and database conceptual modeling. The paper concludes that extending rich semantic data models with support for reasoning is an interesting alternative to extending description logics with data management functionality.

1 Introduction

Information sharing, rather than information processing, is what characterizes information technology in the 21st century. Consequently, ontologies gain increasing attention, as they appear as the most promising solution to enable information sharing both at a semantic level and in a machine-processable way. Ontologies by definition provide an encoded representation of a shared understanding of terms and concepts in a given domain, as agreed by a community of people. But ontologies are not all alike. At least three orthogonal criteria may be used to differentiate among them.

Ontology Focus. Wordnet (<http://www.cogsci.princeton.edu/wn/>) is the most well-known representative of first-generation ontologies that basically provide

definitions of terms and are intended to be used as sophisticated thesauri. Their structure shows terms organized into a subsumption hierarchy (each term conveying the definition of a more specialized concept than its parent term), and linked by other relationships to express synonymy, composition, etc.... This kind of ontology, usually referred to as taxonomic ontologies, is useful in information-sharing infrastructures to provide a reference vocabulary for aligning names denoting data in different data sets. Other ontologies reach beyond terminology, defining conceptualizations that include the representation of properties of concepts and their interrelationships. These "descriptive" ontologies resemble database schemas, showing concepts interconnected by a variety of semantic associations to achieve a semantically rich representation of the intended domain. An example (out of the many existing ones) is ImMunoGeneTics, an international medical ontology (<http://imgt.cines.fr>). These ontologies are useful in information sharing to align existing data structures (not just terms) into an integrated description of the corresponding domain, or, in a top-down perspective, to provide patterns for the definition of new, specialized ontologies or database schemas.

Ontology Scope. The term scope here refers to the intended use of the ontology. Ontologies may be designed and used for purely explanatory purposes, i.e., as a service to enable the understanding of some domain. Such ontologies usually come without associated instances, as these are at a level of detail whose representation is most often not relevant. Ontologies may also serve as a means to actually support some data management services. Such ontologies have associated instances, stored in either a database or a semi-structured data set in e.g. a web server. In the latter case, the ontology plays the role of a database schema in guiding access to the data in the web server. In the former case, the ontology may either be used just to assist in the design of the database schema by providing background semantic information, or as an operational component in the information management architecture, providing access functionality that is additional to those typically provided by a DBMS. A characteristic example of additional functionality is the management of incomplete data. The split between explanation (non instantiated) and management (instantiated) ontologies is orthogonal to the split between taxonomic and descriptive ontologies. A taxonomic ontology serves a data management goal if it contains, for instance, references to databases where the user may find data corresponding to a given term or concept, thus facilitating information retrieval. Descriptive ontologies defined by some standardization body for a given application domain are explanatory. They hold generic abstractions of a domain, not aiming at managing a specific database. However, descriptive ontologies, looking very much alike a database schema, are natural candidates for being used for data management.

Ontology Context. Ontologies traditionally convey a single, monolithic conceptualization, supposedly stating the truth, i.e., how the described domain should be understood. But many different conceptualizations may exist for the same real word, each one defining a view shared by some user community. Contextual ontologies have been proposed to support alternative views of the world.

They provide definitions and descriptions that are context-dependent, thus supporting use from inhomogeneous user communities. The advantage of a contextual ontology, versus the alternative to have independent ontologies (or a hierarchy of ontologies), is that in a contextual ontology it is very easy to support navigation among contexts, i.e. dynamically moving from one context to another. Another advantage is to have a multi-context vision of the world available as a single consistent whole. This is particularly important when updating ontologies is considered and update propagation from one context to another is desirable.

While the traditional use of ontologies is more on the taxonomic and explanatory side, there is a tangible push to move towards descriptive, and even management-oriented ontologies, in order to use them as a key component in data integration frameworks. In such frameworks ontologies do not refer anymore to arbitrary, abstract perceptions of the real world. They describe some defined subset of the real world that is actually represented in the stored data. The increase in similarity between ontologies and database schemas [1] arises a legitimate question about the appropriateness and effectiveness of using a conceptual modeling approach (that has proven to be the best to elaborate a semantically rich description of the data in a database) to describe the conceptualization that is the subject of an ontology. In other terms, could the conceptual modeling know-how provide an interesting alternative to traditional description logic (DL) based approaches? This debate is still open. Some strong proposals from the database [2] and knowledge representation communities [3] already challenge the corpus of work more directly framed in the context of description logic and its reasoning capabilities (DAML+OIL [4]). The purpose of this paper is to contribute to the debate a detailed analysis of requirements for the description and use of what we believe will become the leading ontology framework, i.e. descriptive and, in the longer term, management ontologies. The paper also analyzes differences and complementarities between proposed approaches to ontology description and use. We conclude that enhancing conceptual models with reasoning support may be the best way to make ontologies operational in data integration frameworks.

The next section briefly introduces selected related work and representative proposals for ontology management. It also introduces the conceptual model we propose. Its facilities are illustrated using a simplified example (borrowed from [2]) of a scientific conference ontology. The same example is used throughout the paper. Section 3 discusses ontology requirements, focusing on differences with traditional database requirements. The highlighted issues are detailed in the sequel of the paper. Section 4 focuses on the comparison of the data models. Section 5 deals with how instances are handled. Section 6 analyzes constraint specification and consistency checking. Query languages are discussed in section 7. Some additional features for ontologies are described in section 8, before the concluding remarks.

2 Related work

Interactions between the ontology and conceptual modeling domains is a topic of rapidly growing interest for both communities, as witnessed by looking, for instance, at the number of ontology-oriented contributions at the last ER conference on Conceptual Modeling. The present workshop on Semantic Web and Databases is another clear sign of interest into these interrelationships.

A well-worth-reading starting point when targeting a comparison of models for ontologies is the paper by D. McGuinness [5], which provides interesting insights into a historical perspective of description logic developments, explaining how emerging applications such as those on the web have motivated the use of description logics.

A. Borgida and R.J. Brachman [6] adopt a conceptual modeling perspective to discuss ontology modeling issues, looking in particular at object-based aspects and DL. They highlight some weaknesses of DL in representing structured values, some kinds of constraints, and some forms of "inheritance" (for materialization) and meta-information for conceptual modeling. Conversely, they identify strengths of DL as its specific features to specify primitive and defined concepts, necessary and sufficient conditions, and its reasoning tools. We pursue the same comparative analysis by addressing more specifically ontology requirements and complementing the comparison of the database conceptual models and description logic approaches for modeling with two other issues: Instances handling and querying.

The comparative analysis of Entity Relationship (ER) models and DL proposed in [7] develops the transformation of ER schemas into knowledge bases. The chosen description logic, \mathcal{DLR} [8], is a generalization of description logic for n-ary relations. The authors argue that the semantics of ER models can be captured by \mathcal{DLR} . They also address querying issues, showing that DL queries can only return subsets of existing objects, while database queries may also create new objects. Desirable extensions of standard DL queries are discussed. Our work also focuses on the comparison of conceptual models and DL. We complement their analysis by adopting the reverse viewpoint: We aim at identifying desirable extensions of conceptual models to better address ontological issues. We actually follow the path lead by R. Meersman [1], who argues that methods and techniques originally developed for database conceptual modeling and large databases management could be relevant for ontologies.

His DOGMA project is one of those we explicitly discuss in this paper. Indeed, to substantiate our analysis on trends in ontology management, we have looked in detail at proposals that we felt were good representatives of the alternative approaches from the research communities in artificial intelligence, knowledge representation, and databases.

Artificial Intelligence Approach and Reasoning. Description logics and their associated inference techniques have been extensively used as formal theories on which several ontology languages have been defined. We have chosen RACER [9] to represent this research area, because it has a wide range of applicability as it includes instance management facilities. RACER is a description logic rea-

soning system based on the *SHIQ* logic [10], [11]. RACER separates the formal description of the ontology schema (denoted as the TBox) from the description of individuals (in the ABox). RACER modeling constructs include:

- *Concepts*. They are atomic types defined by their names. Logical expressions may be attached to them, thus allowing designers to define ⁴: a) subsumption hierarchies, e.g. (*implies Author Person*), b) constraints associated to a concept, e.g. (*implies Author (at-least 1 Writes)*), c) stand-alone constraints, e.g. (*disjoint Person Committee Review Paper Topic*), and d) a new concept, using a logical assertion, e.g. (*equivalent PCAuthor (and Author Reviewer)*). Concepts defined by a logical assertion are called "defined concepts", as opposed to the other ones called "primitive concepts".
- *Roles*. They define binary relationships between a domain concept and a range concept, e.g., *roles (Writes: domain Author :range Paper)*. Roles played by a concept can be qualified using quantified restrictions (*some, all*) and numeric restrictions (*at-most, at-least, exactly*). Roles may be transitive, symmetric, and functional. They can have an inverse as well as super-roles.
- *Domain of values*. Integer and real named domains of values can be defined. But attributes, i.e. binary links from a concept to a domain of values, cannot be defined within the schema. Attribute values are dynamically defined and associated to instances of concepts. Lastly, RACER allows users to define in an intentional way, i.e. by a logical expression, characteristics of instances. For example, one can state that paper p100 has been written by exactly one author: (*instance p100 (and Paper (exactly 1 WrittenBy))*).

Artificial Intelligence Approach and Knowledge Representation. We chose KAON [3] as a representative proposal transferring a knowledge representation know-how into the ontology domain. KAON is an ontology and semantic web framework allowing the design and management of ontologies. It includes an ontology modeling language based on RDF(S) with some proprietary extension and a conceptual query language. KAON supports modularization through the recursive definition of sub-models. Each sub-model has (similarly to RACER) two components:

- An ontology structure, holding definitions of concepts, oriented binary relationships between concepts, and attributes. Relationships may be symmetric, transitive and have an inverse. Minimum and maximum cardinality constraints for relationships and attributes may be specified. Concepts and relationships can be arranged in two distinct generalization hierarchies.
- An instance pool, holding concepts and relationship instances and attribute values. Specific to KAON is the possibility to have spanning objects, i.e. a real world entity being represented both as a concept and as an instance.

Database Approach. DOGMA [2] is an ontology engineering framework based on the ORM (Object-Role-Modeling) conceptual model[12]. ORM is a binary relationship data model. DOGMA splits the ontology into two parts:

⁴ Examples refer to the Conference ontology illustrated in Figures 2 and 3 in RACER (and in Figure 1 in MADS).

- The ontology base, holding the data structure. Its definitions may be contextualized using a context name.
- A set of ontological commitments. A commitment is a set of integrity constraints (e.g., definition of identifiers, cardinalities) that govern the ontology for its use in a specific application. The idea is that the ontology base holds generic knowledge about a domain, while its association to a commitment set specializes the ontology for a given application within the domain.

Our approach to ontology modeling also belongs to the database inspired track. While DOGMA (as description logics and KAON) organizes the world as a collection of object tokens associated to properties and interrelated by binary relationships, we favor a more synthetic view, as supported by complex object data models (e.g., UML, extended ER models, semantic models). MADS [13] is such a data modeling framework. MADS is a spatio-temporal conceptual model that handles complex objects (i.e., objects with a multi-level attribute structure, where an attribute can be composed of other attributes), n-ary relationships with attributes, generalization hierarchies, multi-instantiation, as well as spatial, temporal and contextual features (context is materialized by stamping definitions, values and instances to express for which context they are relevant [14]). Both object and relationship types are first class constructs. MADS has associated data manipulation languages. The MADS framework includes a visual schema editor, a visual query editor and the associated mappings onto existing DBMS. It provides users with an integrated environment where they can work at the conceptual level for both designing and querying the database. Figure 1 uses traditional ER diagrammatic techniques to show a MADS data structure (without space, time, and contextual features) for our running example about activities and contributors of a scientific conference.

3 Ontology Requirements

This section holds introductory discussions of the four major components of an ontology management approach: How the conceptualization is described, how associated instances are managed, how reasoning is performed, and how data is queried. The discussion points at similarities and differences between ontology requirements and requirements for traditional databases. Sections 4 to 7 look in more detail into each issue.

Data Modeling. As we believe future ontologies will be descriptive rather than purely taxonomic, we assume the conceptualization includes the definition of relevant data structures. For instance, Figure 1 can be interpreted as illustrating an ontology data structure for management of conference reviews. Representing the knowledge that "papers are assigned to reviewers" as a data structure showing a relationship type linking the two complex object types defining papers and reviewers, is semantically richer than embedding the same semantics in the separate definition of three terms (paper, reviewer, assignment) in a taxonomic ontology. On the contrary, binary data models à la DOGMA, KAON, and description logics may provide a good solution for taxonomic ontologies: Concepts

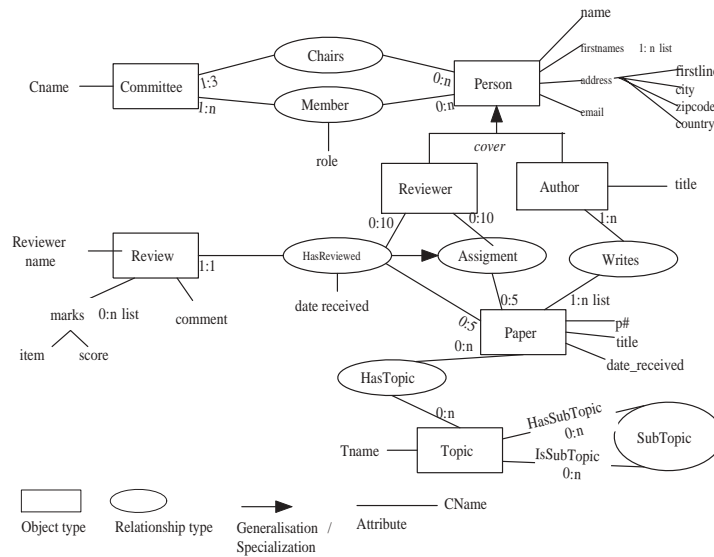


Fig. 1. MADS schema for the scientific conference ontology.

may represent terms and binary links may represent the classic taxonomic relationships, e.g., synonym, homonym, hyponym, holonym. Defining a semantic data structure is also an extremely efficient support for visualizing in an intelligent and intuitive way how the domain of interest is articulated into its many concepts. The capability to visualize the structure of a domain has always been one of the best selling arguments for conceptual models and their acceptance by users. It also has been concerns of knowledge representation systems [15]. Of course, this argument is irrelevant if the ontology is automatically built using some emergent semantic technique, or if the ontology is explored by agents only.

Aiming at expressiveness of concepts for the representation and definition of data structures, powerful conceptual data models naturally appear as the best candidate. They have been purposely and carefully developed to enable building representations that are as rich and as close as possible to human perception. They have proven to be quite successful with users. The same benefits can be expected in using them to build ontologies. A number of researchers [16], [17], [1] have already argued in favor of "highly intuitive" ontology models with a "frame-like look and feel" or "database schema" alike. We support this viewpoint. Nevertheless, ontologies may require even higher expressiveness than conceptual models, as, beyond modeling, they aim at supporting reasoning on the description of the domain of interest. As will be shown hereinafter, this requires extending current conceptual data models with some additional features.

Instance Handling. Instances always exist in a database (except during the design phase), the main purpose of a DBMS being to provide efficient services for storing and handling the instances. As seen in section 1, instances do not

necessarily exist in an ontology. While in a database framework there is a clear separation between the schema (metadata) and the instances (data), and the schema definition is completed before instances are created, this separation is not always enforced in ontological frameworks, where instances may be created anytime. The database approach is normative, in the sense that the database schema defines how the world is, and instances are accepted only if they fully comply with the definitions and constraints stated in the schema. The ontology approach is only partly normative, as it accepts instances as long as they do not explicitly contradict the knowledge already in the ontology, without requiring that all expected data being present. When an inference mechanism finds that an instance should hold a characteristic that is not present, the ontology assumes that the instance does hold it. In other words, databases work with a closed-world assumption, while ontology systems apply an open-world assumption.

Reasoning. The ontology world seems to follow a collaborative approach, where the conceptualization at hand may continuously evolve through updates from a community of users, without a normative policy or sequence ruling the process. For example, the specification of a concept (e.g., the *Paper* object type in the conference example) may be changed anytime, irrespectively of the fact that the ontology holds instances of the concept. In the ontology approach, the specification of a concept defines the condition that its instances must verify. At any time instances are classified in concepts according to these specifications. If the specification of the concept or the characteristics of the instance are modified the classification of the instance is automatically updated. This flexibility is enabled by the existence of powerful reasoning mechanisms. An even higher flexibility is provided in ontology approaches that support so-called spanning objects [18] i.e. objects that are both at the instance and at the type level (these objects are both source and target of instance-of links). For example in [18], an Ape may be an instance of the Species type and a type for ape objects. Although theoretically possible by introducing a meta-schema level, spanning objects are not supported by database technology, which for pragmatic reasons limits its interest to the two basic levels, schema and data.

Queries. Querying in databases is used to retrieve data. Queries are expressed on the schema, which is supposed to be known to users that want to formulate a query. Ontology users are more prone to start their search for data by wondering about what information is actually held by the system. These users (or agents) will first query the ontology schema, to identify what relevant information exists, and then proceed to query the data to extract the desired information from the underlying databases.

4 Data Modeling

In the previous section, we have argued that, due to strong similarity between descriptive ontologies and database schemas, conceptual data models are good candidates for ontology modeling. In this section we analyze differences between constructs in conceptual models and in current ontology proposals.

Object Structure. Ontology models, as we have seen, adopt a binary (also termed functional) approach. Objects are mere tokens (i.e., objects with only an identity and no value) that gain their semantics through binary relationships with other objects or value domains. The known disadvantage of the approach is that a real world entity is scattered into its most elementary pieces and the vision of the thing as a whole is lost. Conceptual models, like MADS, that support complex (NF2like) object structures can represent each real world entity as a single object. This greatly reduces the complexity of the schema. Figures 1, 2 and 3 show that, even for an over-simplified example, the difference in readability is important. The MADS diagram in Figure 1 only needs 7 object types, while the equivalent DL diagram needs 27 objects types. The latter also doubles the number of relationship types if its inverse roles have to be represented. The gain in semantic expressiveness induced by complex objects is worth the additional challenge in implementation.

Object Identity. There is a general agreement that object instances should have a unique object identity. Originally, object identity is system defined and not visible to users. Some ontology approaches (including KAON and RACER) leave it up to users to define the identity of each object. In our opinion, this policy hardly scales up to the very large sets of instances that may be expected in future ontologies.

Generalization Hierarchy. Is-a links, with population inclusion semantics and property inheritance, and generalization hierarchies (or lattices) are standard constructs in both ontology languages (where the term subsumption is often preferred to the term is-a) and conceptual models. Notice that rules for generalization hierarchies in conceptual models may differ significantly from object-oriented models rules. MADS, for instance, allows an object instance to dynamically gain (or loose) membership in (or from) other classes. MADS also supports multi-instantiation, i.e. a real world entity can be represented by several instances belonging to different classes. For example, a person can be both an author and a reviewer. A generalization hierarchy may similarly be defined on relationship types. DL models follow a similar approach. However, they have different default assumptions. In KAON and DL models (e.g. RACER), by default any two concepts may contain common instances. In conceptual models, like MADS, by default two object (or relationship) classes with no common ancestor in the generalization hierarchy (but the root) are disjoint. With the permissive approach of DL, non-careful users may unwillingly create unwanted multi-instantiations that are automatically deduced by the inference engine from their assertions.

Defined constructs, views and derivations. The main goal of ontologies, supporting precise definitions of concepts in relation to other concepts, is fulfilled by the possibility to define concepts using an intentional formula. For example, based on the Conference ontology, one may want to define new concepts such as *PCPaper* (to represent papers submitted by at least one member of a committee), *ChairPerson* (persons chairing a committee), and *SwissAuthor* (authors from Switzerland). In DL these defined concepts are managed exactly in the

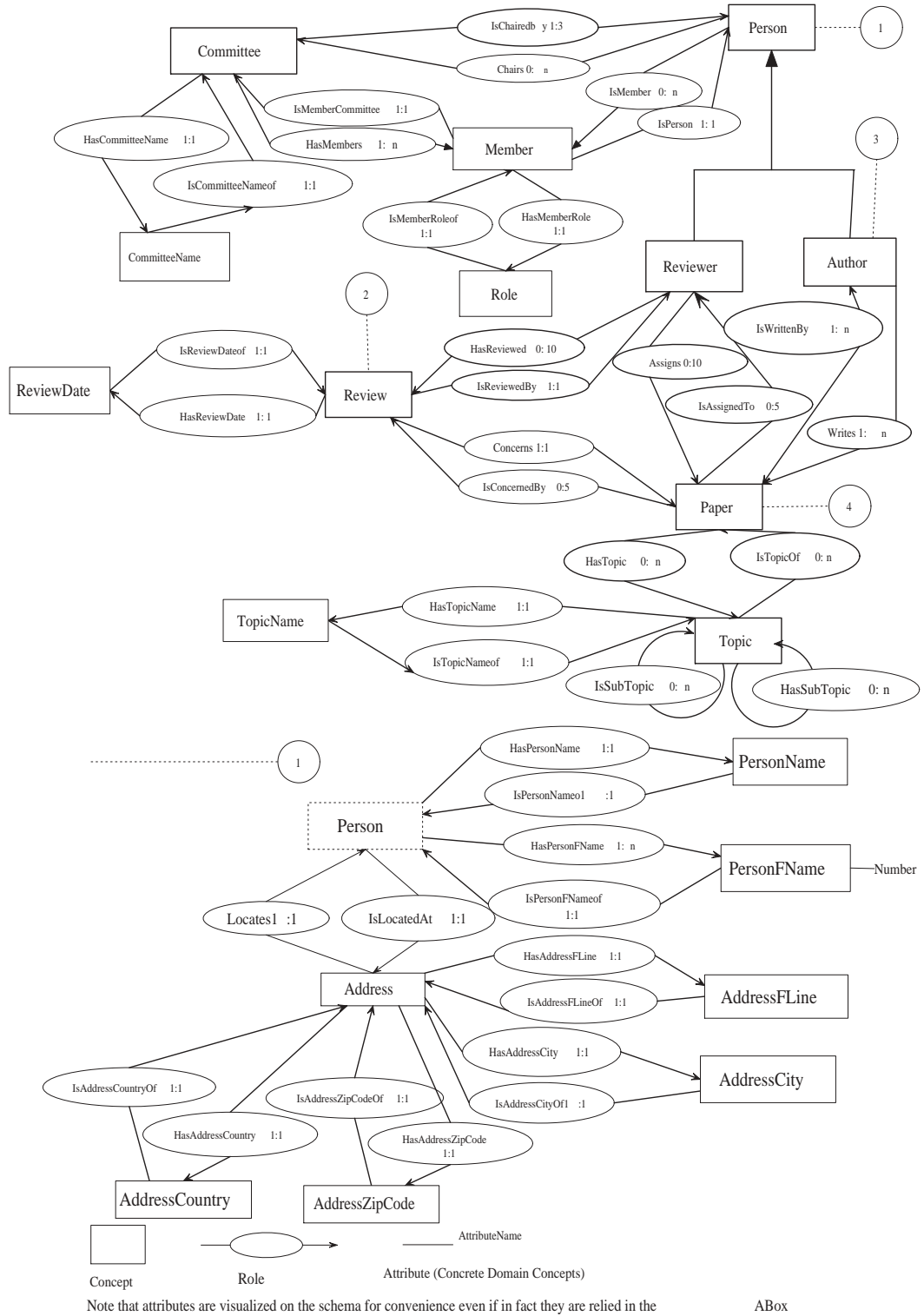


Fig. 2. Scientific Conference Ontology Snapshot with DL formalism (Part1)

same way as the other concepts (the primitive ones), which means, among other things, that they belong to the same generalization hierarchy.

Databases, interested in redundancy-free schemas and data, are not prone to support defined constructs. Nevertheless, some DBMS provide a somehow similar functionality through views and derived constructs. Views are relations (or object classes) defined by a query. Their goal is to provide users with another presentation of a subset of the database in order to make user querying easier. Views do not belong to the database schema. They form an external level that acts as an interface for the users. In terms of supporting derivation mechanisms, some conceptual models allow designers to associate to an object class (or relationship class or attribute) an expression that automatically generates the instances of the class (or the values of the attribute). The designer also explicitly defines the structure of the derived class and its position in the generalization hierarchy. Only the identities of the instances (the values, in case of an attribute) are automatically inferred. An example of a derived construct is a cluster of is-a links defined by a classifying attribute whose value determines which sub-class each instance belongs to. Another example is MADS support of derived topological relationships. For instance, a derived inclusion relationship *On* may be defined to link two spatial objects classes *Parcel* and *House*: Each time the geometry of a house is inside the geometry of a parcel, the system automatically generates an instance of the *On* relationship that links the house and the parcel.

Let us compare these three mechanisms using the following criteria:

- *Which modeling constructs can be defined, derived or can be a view?* Description logics support defined concepts and roles. Both relational and object-oriented DBMS support views for their main construct only: tables or object classes. MADS supports derived spatial and temporal relationships and derived attributes. Defined object classes could be implemented by queries whose resulting type would be added to the database schema. Description logics, RACER, and KAON also support some kind of derived relationships: Instances of transitive, symmetric, and inverse roles are automatically inferred. DOGMA supports symmetric cyclic relationships.
- *What is the status of the defined, derived construct, or view?* In DL there is no difference between defined and primitive concepts except their definition. In databases the specificity of a derived construct is that it cannot be instantiated directly by users. View is a special construct that does not belong to the set of constructs of the data model.
- *How powerful is the defining formula, derivation expression, or the query defining a view?* As the data model of DL is based upon token objects, DL formulas have to define identities only, while in databases derivation expressions and queries for views have to define identities with the associated structured values. Another difference is that DL formulas define only sets of existing instances, while queries can be either object preserving (defining new representations for existing objects) or object generating (creating new objects with new oids). Examples of such queries for the Conference

ontology are, respectively: *PCPaper*, the set of papers written by at least a member of a committee, and the new relationship class *ReviewerAuthor* that links each author to each of his/her reviewers. Borgida [19] showed that DL formulas have a limited power compared to databases query languages. They are equivalent to first order logic with 3 variables. For instance, it is easy to write a query that finds the papers that have been assigned to one of their authors (an error of assignment), while in DL it is impossible to write the equivalent formula. As for derivation expressions, they vary according to the data model. Often they are predefined and therefore are expressions with limited power.

In conclusion, one could roughly say that conceptual models are better at designing primitive concepts because they can describe more complex structures, closer to the real world, and because they support appealing visual diagrams and design tools.⁵ They also support derived constructs whose instances can be automatically inferred. But, contrarily to models based on DL, they do not support defined constructs that designers can define by a logical formula without knowing where they will fit in the generalization hierarchy or even knowing the generalization hierarchy.

5 Instance Handling

Ontologies may include instances, as databases routinely do, as part of their domain of interest. To realistically manage large sets of instances, storage and transaction management mechanisms that support security, concurrency, reliability, query optimization, and scalability are needed. As this is exactly what a DBMS provides, DOGMA, KAON, and MADS delegate such services to an underlying DBMS. KAON, for example, stores ontology instances in a relational database [18]. DOGMA, KAON, and MADS are built as a layer in between users and the DBMS, providing an ontological or conceptual modeling perspective on the data. RACER, instead, uses a proprietary file system, which limits portability and does not provide all of the above services. Instance management includes manipulation facilities such as insertion, deletion and updating. They should be accessible both via some user-oriented assertional language (à la SQL, for example) and via some API providing one instance at a time access. Both types of DML are fully supported by a DBMS. RACER provides only elementary facilities. Attribute values are treated as objects, which requires three operations to define a value for a simple attribute of an instance: (1) creation of an object-value, (2) assigning the value to the object-value, and (3) linking this object-value to the instance. DOGMA, KAON, and MADS offer (or plan to offer) a conceptual DML that corresponds to the data modeling paradigm they

⁵ In order to achieve readability and understandability by users, visual diagrams purposely limit their expressive power to a subset of the concepts in the conceptual model. They are complemented with textual specifications that complete the description of the schema.

use. Users can, as in RACER, load and manipulate instances that obey the rules of the ontology, without having to consider the representation format used to store instances.

Ontologies, however, need additional manipulation facilities. DBMS users are expected to know the schema and issue manipulation requests that conform to the schema and the associated constraints. For example, to insert a new instance users have to explicitly specify its type and to provide its value and links. Moreover the value and links have to obey the format and constraints defined for this type. Description logics assume users (humans or agents) may be only partly (if at all) aware of the schema (concepts and role definitions). A DL schema hence acts like a set of sufficient conditions that define the membership of the instances to concepts (or roles). Therefore, DL systems allow users to insert a new instance giving only an intensional definition (i.e., a logical formula) characterizing the target concept. The reasoner then computes the concepts the instance belongs to. Finding the most specific concepts an individual belongs to is called *realization*. For example, RACER supports the definition of an instance by specifying its properties *instance Mary (some Writes Paper)* and realization allows finding its most specific concepts. The system will deduce that the instance *Mary* is an instance of *Author*. Realization may be much more complex than in this very simple example.

Database systems are not meant to provide such looseness in instance manipulation. As already stated, they are normative. They follow the closed world assumption, stating that only information that is present in the database (or derivable by explicitly defined derivation rules) is valid. Consequently, they do not need sophisticated reasoners to infer additional information. DL systems naturally adhere to the open world assumption, which assumes that present data is just the explicitly known subset of the valid data, and more valid data may be inferred by sophisticated reasoning. Thus, if an assertion implies a deduced fact that is consistent with all known assertions and instances, then the fact is assumed to be true even if it is not present in the instance set. Otherwise stated, an insertion in DL is always treated as the insertion of incomplete information. For example, a database will accept the insertion of the above instance *Mary* in *Author* only if it comes with the insertion of (at least) one instance of the *Writes* relationship involving *Mary*. RACER accepts the insertion of the single *Mary* instance, deducing that the paper written by *Mary* is presently unknown (see [20], for more on this discussion). In addition, *Recognition* is performed when an already existing instance of a concept acquires (or loses) a characteristics and therefore gets (or loses) a new instantiation in another concept. For example, on the insertion of a new role instance linking a *Person* instance *p* to a *Paper* instance, RACER infers that *p* is also an instance of *Author*.

6 Constraints

In the database world, constraints significantly enrich data description. They state rules that apply in the real world of interest (e.g., one paper has at least

one author) and rules that define the conditions for real world phenomena to be or not to be of interest for the intended application (e.g., a committee is not registered in the database before at least one of its members is registered in the database). Constraints, as other schema definition statements, are understood in the DB world as normative specifications. They entail consistency-checking mechanisms, to verify that instances in the database satisfy all constraints. DL languages are also able to express rules similar to database constraints (e.g., min-max cardinalities). However, only some of them actually constrain the instances in the A-Box. For example, a maximum cardinality specification acts as a constraint as an attempt to create more instances than allowed would result in an inconsistency that is detected and rejected. On the other hand, a minimum cardinality specification only acts as a descriptive feature, as a DL system would accept, e.g., the creation of a paper without an author, simply assuming that the information in the instance base (the A-Box) is temporarily incomplete. We believe that the possibility to define normative constraints, as in the DB approach, is a desirable feature also for ontologies.

There are such a variety of constraints that data models almost necessarily only include part of them in their constructs. Implicit *model constraints* rule the use of modeling constructs and are built-in in the data model, i.e. they act as syntactic constraints that are automatically enforced by the data management interface. For instance, in RACER a role definition has to specify a domain and a range; in conceptual data models a relationship type is only allowed to link object types, and a relationship instance is not allowed to have pending roles; in both DL and DB models cycles of is-a links are forbidden. Explicit definition of constraints is used to describe the semantics of the domain/ontology. According to the approach, they come in two different ways. 1) *Embedded constraints* are expressed using dedicated constructs in the data model. Examples include cardinality and identifier specifications (e.g., the NOT NULL and PRIMARY KEY clauses in relational DBMS), set constraints on groups of roles or is-a links (e.g., disjunction, inclusion, cover, partition), and simple integrity constraints (e.g., using the CHECK clause in relational DBMS). 2) *Integrity constraints* are those that are not directly supported by clauses in the model itself, and thus have to be explicitly expressed using a complementary technique, such as a generic declarative language (e.g., first order logic), a generic programming language (e.g. stored procedures or methods), or triggers. As DL axioms can define a large range of embedded constraints, DL approaches do not resort to additional mechanisms for integrity constraints definition. On the contrary, DB approaches rely on such mechanisms. DL also differs from DB approaches in that DL allows associating a constraint to an instance, while DB considers constraints as meta-information and always associates them to types (thus constraining under the same rule all instances of the type). In terms of expressiveness of the language to define constraints, first order logic (FOL) is the closest to full expressive power using a declarative approach. DL languages usually support a more limited expressive power. For instance, RACER cannot express key constraint involving multiple attributes and ad hoc constraints such as "an author cannot

review his/her papers”. More expressive DL such as \mathcal{DLR} supports this kind of constraints.

A meta-question about constraints and ontologies is whether constraints should be included in ontologies at all. Most frequently, constraints are intertwined in the T-Box with the data description statements. Meersman and his group [2] take the opposite view that all constraints should be separated from data description and defined in a commitment layer. The supporting argument is that constraints are application specific, while the ontology should be application independent. We agree that having a commitment layer is the appropriate way to handle application-specific semantics. Nevertheless, there are in our opinion constraints that belong to the ontological world, i.e. that form an essential component in the description of the semantics of things. For instance, the fact that in a conference management ontology a review of a paper should never be assigned to one of the authors of the paper is a constraint that is unanimously agreed upon. It is a constraint that is tightly linked to the semantics of a review (defined as a critical appreciation of a work by a person not involved in the work). On the contrary, whether a conference committee has one or more chairpersons varies from one conference to the other. The ontology could state a 1:N cardinality (to make sure a committee has at least a chair), leaving to the commitment layer to refine the cardinality to 1:2 or 1:3 or whatever else fits the conference specific requirements.

Checking the consistency of the set of constraints and checking the consistency between the constraints and the schema are tasks that can be performed automatically by the reasoners available in description logics. Constraints are expressed in the same formalism as the other description clauses; hence they are naturally involved in the inferring. The same functionality is not provided by current database technology, where applicable reasoning techniques cannot grasp the semantics hidden in the external language expressions used to define integrity constraints.

Checking the consistency of schema specifications is also an issue where DL and databases take different approaches. In databases, it is not possible to validate a schema that does not obey model constraints. As there is no defined construct, there is no need for sophisticated reasoning in checking the consistency of the schema. Reasoners are necessary in DL to check the consistency of primitive, defined concepts, and other axioms. They define a valid schema as a schema such that it is possible to find one instance of the ontology that has at least one instance in each one of its concepts. While database users can never actually use an invalid schema, in RACER, for example, users can request a consistency check anytime and they can continue working on an inconsistent schema.

7 Ontology Querying

Like databases, ontologies are queried by different categories of users, with different needs:

- Ontology administrators, whose role (like DBAs, database administrators) is to design and maintain the ontology schema and monitor its evolution. While a DBA is seen as a central authority, ontology creation and evolution is often seen as a more cooperative activity, distributing the task among many people. Hence schema querying is likely to be a more intensively used functionality, with the schema continuously and incrementally growing with many defined concepts.
- End-users, who will face a large and complex schema that they may not know well. They also will query the schema to know what is in the ontology. They may also write mixed queries to access both the schema and the instances. For example, a user of a geographic ontology describing the various states of a country may ask for all information (list of properties with their description and value) about rivers and synonyms of rivers. Therefore, both the schema and the instance base should be accessible through the same query language, possibly within the same query.
- Application developers, who need to gain access to the ontology and its services via an API.

Requirements for instantiated descriptive ontologies include the usual services supported by DBMS, namely a generic assertional query language with associated tools for automatic query optimization. The expressive power, and its optimization possibilities, of the language are bound to the characteristics of the associated ontology model. For instance, queries in a language for a binary model, like those of KAON and RACER, will return types or instances of the elementary constructs of the model: concepts and binary relationships. On the other hand, queries on semantic models with structured objects, like MADS, will return structured instances, i.e. more informative and more condensed results. Therefore, for descriptive ontologies frameworks, where understanding the data structure may be a challenge, semantic query languages returning structured objects are likely to perform better than languages for binary models.

Another requirement is that the same query language should support querying both the schema and the instances in the same way. Models that host the description and the instances of the ontology within the same structure automatically fulfill this requirement as it is possible in RACER or KAON. For models that clearly separate the schema from the instances, like database models do, a solution is to store the schema as instances of a meta-schema described with the same model as the ontology. Such a solution is currently provided by relational DBMS that support a data dictionary made up of a set of predefined tables that describe the tables of the application schema.

Several functionalities should be provided for schema querying. Exploration of the schema is the first one. When the schema contains concepts defined by logical formulas, reasoning comes as the second one.

Schema exploration. The query language should allow getting all information existing in the schema. Examples of such queries could be:

- Give the characteristics of a relationship (transitive, symmetric, inverse)
- Give the relationships going from (or to) a concept.

Queries of this type can be formulated in the RACER language using the elementary predefined functions that are provided for navigation inside the schema: *describe-concept*, *describe-role*, *reflexive?*, *symmetric?*, *transitive?*, *feature?*, *role-inverse*, *role-domain*, *role-range*. KAON also provides similar functions, such as *Properties_From*, *Properties_To*, *Domain_Concept*, *Range_Concept*, *SubConcepts*, *SuperConcepts*.

In MADS, this can be done by defining the schema of the meta-model of MADS, and querying this meta-base with one of the generic languages of MADS (visual or textual algebraic). However, for humans a much simpler way to explore the schema is to visualize and browse it using the MADS schema editor.

Reasoning on the schema. Users of ontologies that contain non-primitive concepts defined by logical formulas need a schema query language with new functions for helping them in their understanding of the defined concepts. Typical functions of this type are (here by concept we mean any kind of concept, be it primitive or defined):

Are two concepts equivalent or disjoint? Does a concept (or relationship) subsume another one? Classify the whole set of concepts. What are the superior sub-concepts (at any level) of a concept?

These functions require an inference engine for their evaluation. This justifies the choice of formal models, such as DL, that have powerful tools to classify concepts using the subsumption mechanism.

Instances querying. Databases and DL offer complementary functionality for instance querying. Databases systems usually provide powerful assertional query languages complemented with efficient query optimization tools. These languages, like the ones of MADS, support object preserving as well as object generating queries. They also allow users to define new structures for existing objects by pruning existing properties or computing new, derived properties. On the other hand, DL systems support a set of simple functions for accessing instances and derived facts computed by their inference engines, like the closure (resp. inverse, symmetry) of the transitive (resp. inverse, symmetric) relationships. Moreover DL systems, like RACER, that allow users to associate logical formulas to instances, provide a new reasoning function: "To which most specific concepts does this instance belong?"

8 Additional Requirements

Up to now we have only discussed traditional requirements as addressed by current ontology frameworks. This section highlights some additional features that we believe will in the short term gain importance for the full development of ontologies. The first and most evident additional feature is the possibility to associate temporal specifications to the concepts and roles of an ontology. The semantics of terms and concepts evolves in time, new terms and new concepts appear while other become obsolete. It is therefore important that each item in an ontology be qualified using a temporal specification that says when the definition of this item is valid. How to define and implement such lifecycle spec-

ifications for concepts and roles, as well as time-varying attributes, has been thoroughly investigated by the temporal database community. Results should simply be taken over to ontologies. Similar considerations may apply to spatial specifications, well-known in the world of geographical information systems (GIS). They may describe, for instance, the geographical coverage of a given term (e.g. "car" to denote a car holds in Quebec but not in Paris). Moreover, research in data visualization has shown that ontologies may be displayed as a concept space, where spatial concepts such as distance, neighborhood, inclusion, or orientation may fully apply. Spatial information supports storing the position and topology of concepts in such abstract spaces. Spatial information will also play an important role for ontologies where geographical aspects are part of the domain of discourse. To support its description, concepts and techniques may be borrowed from GIS research. Finally, as we have importance as the actual use of ontologies becomes practically relevant. The need for context information is recognized in the ontology literature, but the current status shows limited results and significant advances may still be foreseen in this domain. How to define a context, how to analyze interrelationships between contexts, how to characterize constraints on contexts, are examples of open research issues. RACER and KAON currently support none of these additional features. Some extensions of DL have been proposed for spatial and temporal modeling [21],[22]. DOGMA includes context information. MADS supports space, time, and context description and manipulation.

9 Conclusion

Ontologies are promised to a brilliant future. As a consequence, usability criteria will assess their success. In our opinion, this means that focus will be on more informative ontologies, showing, in addition to terms and concepts of a domain, how domain data are semantically structured and interrelated. We termed these ontologies descriptive, as opposed to first-generation taxonomic ontologies. The paper investigated requirements for the design and management of descriptive ontologies, and contrasted the requirements with the functionality currently provided by database conceptual models. Proper identification of the requirements has been supported by an analysis of some recent representative proposals for ontology systems (namely, RACER, KAON, and DOGMA). The rationale for this work is the close resemblance between requirements for database design and those for ontology design. We attempted to highlight similarities as well as significant differences in the approaches. Most differences appear to be linked to the current state of art in both domains. These ones may disappear thanks to further research. However, important differences (such as closed versus open world assumptions) are inherently due to the different goals of ontology and database services. Ontologies are meant to describe and explain the world, while databases are meant to describe that part of the world whose representation has to be managed for some application purpose. Overcoming differences is a meaningful way to benefit one domain with results from the other domain. Pre-

vious work has investigated how to extend description logics to provide more data semantic services, or how to map description logic specifications into conceptual model specification, and vice versa. Our aim has been to identify the enhancement that conceptual models would need to make them fit the requirements of ontologies. Briefly stated, the necessary enhancements have obviously to do with supporting reasoning. A major addition is the support of intentionally defined concepts à la DL. This somehow resembles view definition and queries in databases, but views are not part of the database schema and queries raise a number of open issues (e.g., how to place the query object type in the generalization hierarchy in order to explicit the semantic relationship between the new type and the existing types). A minor addition is the explicit definition of the specialization criterion in is-a clusters, so that the system can compute the appropriate sub-class for an object whose value changes or is first created. More additions that we feel important to match coming requirements are provision for spatio-temporal data modeling and context management. We have currently defined and implemented a conceptual data model, MADS, that supports advanced data structure, time, space, and context modeling requirements, as well as query placement to some extent. As a further step towards ontologies, we are extending the model to support imprecise information, where incompleteness is seen as a form of imprecision. This is intended to allow building ontology services above the conceptual services currently provided by prototypes developed within the MurMur IST project [23].

References

1. Meersman, R.: Ontologies and Databases: More than a Fleeting Resemblance. In: OES/SEO Workshop Rome. (2001)
2. Jarrar, M., Meersman, R.: Formal Ontology Engineering in the DOGMA Approach. In Meersman, R., Tari, Z., et al., eds.: *CooPIS/DOA/ODBASE*, Springer-Verlag, LNCS 2519 (2002) 1238–1254
3. KAON: KAON - The Karlsruhe Ontology and Semantic Web Tool Suite (2003) <http://kaon.semanticweb.org/>.
4. Horrocks, I.: DAML+OIL: A reason-able Web Ontology Language. In Jensen, C., et al., eds.: *EDBT 2002*, Springer-Verlag, LNCS 2287 (2002) 2–13
5. McGuinness, D.: Description Logics Emerge from Ivory Towers. In: *Proceedings of the International Workshop on Description Logics*, Stanford, CA (2001)
6. Borgida, A., Brachman, R.J.: Conceptual Modelling with Description Logics. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: *Description Logic Handbook*, Cambridge University Press (2002) 349–372
7. Borgida, A., Lenzerini, M., Rosati, R.: Description Logics for Databases. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: *Description Logic Handbook*, Cambridge University Press (2002) 462–484
8. Calvanese, D., Giacomo, G.: Expressive Description Logics. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: *Description Logic Handbook*, Cambridge University Press (2002) 178–218
9. Haarslev, V., Möller, R.: RACER System Description. In Goré, R., Leitsch, A., Nipkow, T., eds.: *Proceedings of International Joint Conference on Automated Reasoning (IJCAR'2001)*, Springer-Verlag (2001) 701–705

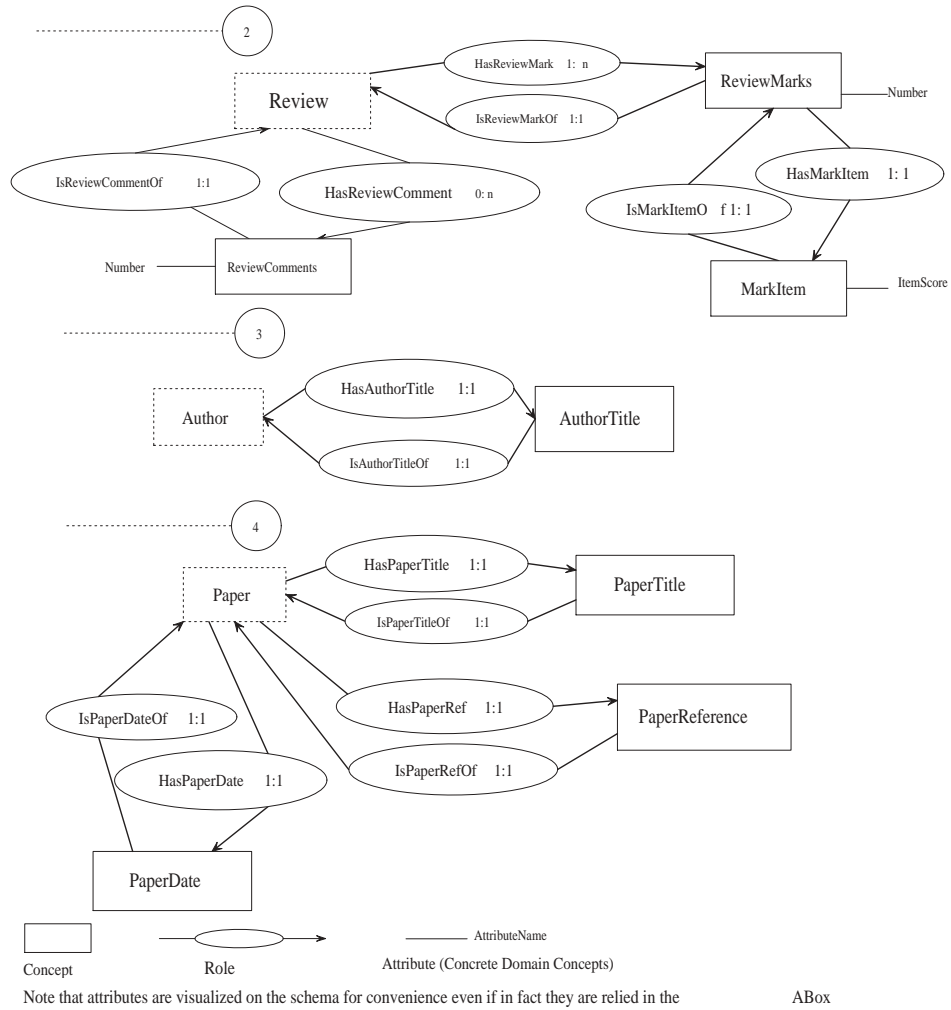


Fig. 3. Scientific Conference Ontology Snapshot with DL formalism (Part2)

10. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with Individuals for the Description Logic *SHIQ*. In MacAllester, D., ed.: Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17). LNAI 1831, Springer-Verlag (2000) 482–496
11. Horrocks, I., Sattler, U.: Optimised Reasoning for *SHIQ*. In: Proc. of the 15th European Conference on Artificial Intelligence (ECAI'2002). (2002) 277–281
12. Halpin, T.: Information Modelling and Relational Database Design. (2001)
13. Spaccapietra, S., Parent, C., Zimanyi, E.: Spatio-Temporal Conceptual Models: Data Structures + Space + Time. In: 7th ACM Symposium on Advances in Geographic Information Systems (ACM GIS'99). (1999) 26–33
14. Spaccapietra, S., Parent, C., Vangenot, C.: From Multiscale to Multirepresentation. In Choueiry, B., Walsh, T., eds.: Proceedings 4th International Symposium, SARA-2000, Horseshoes Bay, Texas, USA, Springer-Verlag, LNAI 1864 (2000)
15. OKBC: OKBC - Generic Knowledge Base Editor (1998)
<http://www.ai.sri.com/gkb/>.
16. Fensel, D., Hendler, J., Liebermann, H., Wahlster, W.: Spinning the semantic web, The MIT Press, Cambridge, Massachusetts (2003)
17. Klein, M., Broekstra, J., Fensel, D., van Harmelen, F., Horrocks, I.: Ontologies and schema languages on the Web. In D. Fensel, J. Hendler, H.L., Wahlster, W., eds.: Spinning the Semantic Web, The MIT Press, Cambridge, Massachusetts (2003)
18. Motik, B., Maedche, A., Volz, R.: A Conceptual Modeling Approach for Semantic-Driven Enterprise Applications. In Meersman, R., Tari, Z., et al., eds.: CooPIS/DOA/ODBASE, Springer-Verlag, LNCS 2519 (2002) 1082–1099
19. Borgida, A.: On the relative expressive power of Description Logics and Predicate Calculus. In: Artificial Intelligence 82. (1996) 353–367
20. Baader, F., Nutt, W.: Basic Description Logics. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: Description Logic Handbook, Cambridge University Press (2002) 43–95
21. Haarslev, V., Lutz, C., Möller, R.: Foundations of Spatioterminological Reasoning with Description Logics. In A.G. Cohn, L.K. Schubert, S., ed.: Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR'98). (1998) 112–123
22. Artale, A., Franconi, E.: A Survey of Temporal Extensions of Description Logics. In: Annals of Mathematics and Artificial Intelligence (AMAI), Vol. 30 No. 1-4, Kluwer Academic (2001)
23. MurMur: MurMur Consortium - MurMur Project: Multi-representations and Multiple resolution in geographic databases (2002) Final Report. <http://lbdwww.epfl.ch/e/MurMur>.

Efficient RDF Storage and Retrieval in Jena2

Kevin Wilkinson¹, Craig Sayers¹, Harumi Kuno¹, Dave Reynolds²

HP Laboratories

¹ 1501 Page Mill Road

Palo Alto, CA, 94304 USA

² Filton Road, Stoke Gifford

Bristol BS34 8QZ United Kingdom

firstName.lastName@hp.com

Abstract. RDF and related Semantic Web technologies have been the recent focus of much research activity. This work has led to new specifications for RDF and OWL. However, efficient implementations of these standards are needed to realize the vision of a world-wide semantic Web. In particular, implementations that scale to large, enterprise-class data sets are required. Jena2 is the second generation of Jena, a leading semantic web programmers' toolkit. This paper describes the persistence subsystem of Jena2 which is intended to support large datasets. This paper describes its features, the changes from Jena1, relevant details of the implementation and performance tuning issues. Query optimization for RDF is identified as a promising area for future research.

1.0 Introduction

Jena is a leading Semantic Web programmers' toolkit[1]. It is an open-source project, implemented in Java, and available for download from SourceForge. Jena offers a simple abstraction of the RDF graph as its central internal interface. This is used uniformly for graph implementations, including in-memory, database-backed, and inferred graphs.

Jena2 is the second generation of the Jena toolkit. It conforms to the revised RDF specification, has new capabilities and a new internal architecture. A design principle for Jena2 was to minimize changes to the API. This simplifies migration from Jena1 to Jena2. This paper describes Jena2 persistent storage; details on other aspects of Jena2 are available in [2].

The Jena database subsystem implements persistence for RDF graphs using an SQL database through a JDBC connection. The Jena1 implementation supported a number of database engines (e.g., Postgresql, MySQL, Oracle) and had a flexible architecture that facilitated porting to new SQL database engines and experimentation with different database layouts. Some options included the use of value-based identifiers (e.g., SHA-1) for inter-table references, use of database procedures, etc. Jena1 also worked with Berkeley DB.

Among the lessons learned from Jena1 was that database portability was valu-

able to the open source community and this was retained as a goal for Jena2. However, Jena1 users did little experimentation with schema flexibility. In general, the default layouts were used. The design focus for Jena2 was performance and scaling. Although Jena1 performance was quite good, there is room for improvement. It was felt that performance issues will become more important with the renewed interest in applying semantic web technology to real-world applications [6]. Jena2 addresses the following performance issues.

Too many joins. The use of a normalized schema requires a three-way join for find operations.

Single statement table. A single statement table doesn't scale for large data sets and cannot take advantage of locality among subjects and predicates.

Reification storage bloat. A naive implementation of the RDF specification stores four statements for each reification. A more efficient technique is required, especially since some applications reify every statement. Jena1 provided optimized storage for reified statements but a statement could only be reified once. Revisions of the RDF specification removed this restriction. The goal for Jena2 was to implement the revised specification with similar or better optimization.

Query optimization. In Jena1, joins for RDQL queries were not pushed-down to the database engine and were instead performed within the Java layer of Jena.

This paper describes how these performance issues were addressed. The next section provides an overview of Jena and RDF for readers unfamiliar with those technologies. More details on RDF are available in [3]. Section 3.0 describes the Jena database schema. Section 4.0 is a high-level overview of the database subsystem of Jena2. Section 5.0 addresses query processing. The final sections cover miscellaneous topics, the status of the implementation and related work.

2.0 Overview of Jena and RDF

2.1 Jena Overview

Jena offers a simple abstraction of the RDF graph as its central internal interface. This is used uniformly for graph implementations, including in-memory, database-backed, and inferred graphs. The main contribution of Jena is a rich API for manipulating RDF graphs. Around this, Jena provides various tools, e.g., an RDF/XML parser, a query language, I/O modules for N3, N-triple and RDF/XML output. Underneath the API the user can choose to store RDF graphs in memory or in databases. Jena provides additional functionality to support RDFS and OWL.

The two key architectural goals of Jena2 are:

- Multiple, flexible presentations of RDF graphs to the application programmer. This allows easy access to, and manipulation of, data in graphs enabling the application programmer to navigate the triple structure.

- A simple minimalist view of the RDF graph to the system programmer wishing to expose data as triples.

The first is layered on top of the second, so that essentially any triple source can back any presentation API. Triple sources may be materialized, for example database or in-memory triple stores, or virtual, for example resulting from inference processes applied to other triple sources.

An simplified overview of the Jena 2 architecture is shown in Figure 1. Applications typically interact with an abstract Model which translates higher-level operations into low-level operations on triples stored in an RDF Graph. There are several different graph implementations, but in this paper we focus on those which provide persistent storage.

At an abstract level, the Jena2 storage subsystem need only implement three operations: add statement, to store an RDF statement in a database; delete statement, to remove an RDF statement from the database; and the find operation. The find operation retrieves all statements that match a pattern of the form $\langle S,P,O \rangle$ where each S, P, O is either a constant or a don't-care. Jena's query language,

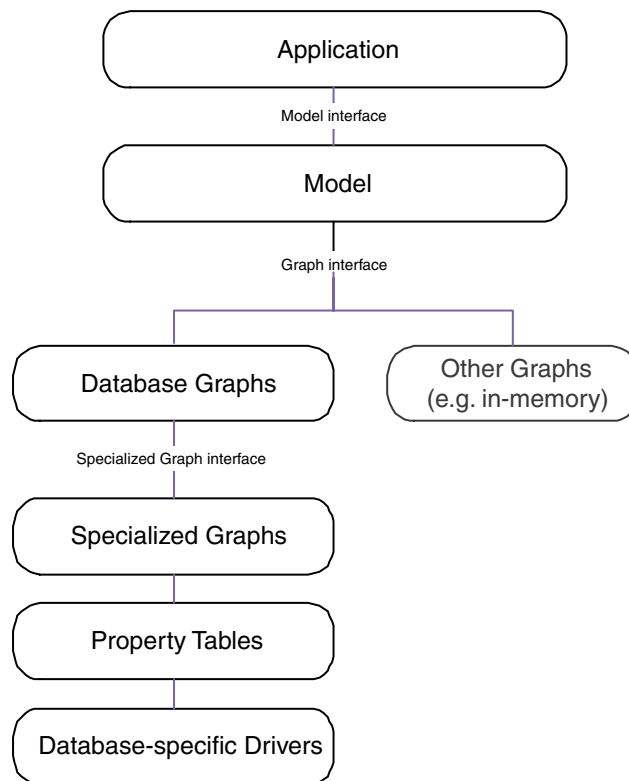


FIGURE 1. Jena2 Architectural Overview

RDQL, is converted to a set of find operations in which variables are permitted in the find patterns. The variables enable joins across the patterns.

A widely used implementation technique [4,5] is to store RDF statements in a relational database using a single statement table, often called a “triple-store.” This is a table that stores each RDF statement as a row and has columns for the subject, predicate and object. Jena1 used this approach but normalized the statement table so that literals and resources are only stored once. This is described below.

2.2 RDF Overview

The Resource Description Framework (RDF) has rapidly gained popularity a means of expressing and exchanging semantic metadata, i.e., data that specifies semantic information about data. RDF was originally designed for the representation and processing of metadata about remote information sources (referred to as resources or Web resources), and defines a model for describing relationships among resources in terms of uniquely identified properties (attributes) and values. RDF provides a simple tuple model, $\langle S,P,O \rangle$, to express all knowledge. The interpretation of this *statement* is that subject S has property P with value O, where S and P are resource URIs and O is either a URI or a literal value. RDF and its related specifications, RDF Schema and OWL, provide some predefined basic properties such as type, class, subclass, etc.

RDF is characterized by a property-centric, extremely flexible and dynamic data model. Resources can acquire properties and types at any time, regardless of the type of the resource or property. This flexibility makes RDF an attractive technology for the specification and exchange of arbitrary metadata because resource descriptions are “grounded” without necessarily being bound by fixed schemas.

In object-oriented (OO) terms, we might consider RDF resources to be analogous to objects, RDF properties to represent attributes, and RDF statements to express the attribute values of objects. A key difference between the two communities is that unlike OO systems which use the concept of a type hierarchy to constrain the properties that an object may possess, RDF permits resources to be associated with arbitrary properties; statements associating a resource with new properties and values may be added to an RDF fact base at any time.

The challenge is thus how to provide persistent storage for the new RDF data model in an efficient and flexible manner. A naïve approach might be to map the RDF data to XML and simply leverage prior work on the efficient storage of XML. However, the standard RDF/XML mapping is unsuitable for this since multiple XML serializations are possible for the same RDF graph, making retrieval complex. Non-standard RDF-to-XML mappings are possible, and have been used in some implementations. However the simpler mappings are unable to support advanced features of RDF, such as the ability of RDF to treat both properties and statements as resources, which allows metadata describing these elements to be incorporated seamlessly into the data model and queried in an integrated fashion.

Statement Table			
Subject	Predicate	ObjectURI	ObjectLiteral
201	202	null	101
201	203	204	null
201	205	101	null

Literals Table		Resources Table	
Id	Value	Id	URI
101	Jena2	201	mylib:doc
101	The description - a very long literal that might be stored as a blob.	202	dc:title
		203	dc:creator
		204	hp:JenaTeam
		205	dc:description

FIGURE 2. Jena1 Schema (Normalized)

Many RDF systems have used relational or object databases for persistent storage and retrieval. However, this is not always a good fit and the mapping can be challenging because the semantics of the underlying database model clash with the openness and flexibility of RDF. For example, SQL requires fixed, known column data types; object systems often have restrictions on class inheritance and type membership (changes).

3.0 Storage Schema

In this section we compare the storage of arbitrary RDF statements between Jena1 and Jena2. We then look at optimizations for common patterns of statements.

3.1 Storing Arbitrary RDF Statements

Jena1. The first version of Jena used two different database schemas. One for relational databases, and a special one for Berkeley DB. For relational databases, the schema consisted of a statement table, a literals table and a resources table (Figure 2). The statement table contained all asserted and reified statements and referenced the resources and literals tables for subjects, predicates and objects. To distinguish literal objects from resource URIs, two columns were used. The literals table contained all literal values and the resources table contained all resource URIs in the graph. There was no reference counting to reduce overhead. This schema was very efficient in space as multiple occurrences of the same resource URI or literal value were only stored once. However, every find operation required multiple joins between the statement table and the literals table or the resources table.

The Jena1 schema for BerkeleyDB was quite different. It stored all parts of a

statement in a single row and each statement was stored three times: once indexed by subject, once by predicate and once by object.

Comparing the two approaches, it was observed that Jena graphs stored using Berkeley DB were often accessed significantly faster than graphs stored in relational databases [8]. While part of this could be attributed to the absence of transactional overheads in BerkeleyDB, our intuition was that most of the speed difference was because the Berkeley DB schema used a single access method to store statements and that use of a denormalized relational schema might reduce response times.

Jena2. The Jena2 schema trades-off space for time. Drawing on experience with Jena1, it uses a denormalized schema in which resource URIs and simple literal values are stored directly in the statement table (Figure 3).

In order to distinguish database references from literals and URIs, column values are encoded with a prefix that indicates which the kind of value (codes are not shown). A separate literals table is only used to store literal values whose length exceeds a threshold, such as blobs. Similarly, a separate resources table is used to store long URIs. By storing values directly in the statement table it is possible to perform many find operations without a join. However, a denormalized schema uses more database space because the same value (literal or URI) is stored repeatedly.

The increase in database space consumption is addressed in several ways. First, common prefixes in URIs, such as namespaces, are compressed. They are stored in a separate table (not shown) and the prefix in the URI is replaced by a database reference. It is expected that the number of common prefixes will be small

Statement Table		
Subject	Predicate	Object
mylib:doc1	dc:title	Jena2
mylib:doc1	dc:creator	HP Labs - Bristol
mylib:doc1	dc:creator	Hewlett-Packard
mylib:doc1	dc:description	101
201	dc:title	Jena2 Persistence
201	dc:publisher	com.hp/HPLaboratories

Literals Table		Resources Table	
Id	Value	Id	URI
101	The description - a very long literal that might be stored as a blob.	201	hp:aResource-WithAnExtremelyLongURI

FIGURE 3. Jena2 Schema (Denormalized)

and cacheable in memory. Expanding the prefixes will be done in memory and will not require a database join.

Second, as mentioned earlier, long values are stored only once. The length threshold for determining when to store a value in the literals or resources table is configurable. Applications may trade-off time for space by lowering the threshold. Third, Jena2 supports property tables as described below. Property tables offer a modest reduction in space consumption in that the property URI is not stored.

Both Jena1 and Jena2 permit multiple graphs to be stored in a single database instance. In Jena1, all graphs were stored in a single statement table¹. However, Jena2 supports the use of multiple statement tables in a single database so that applications can flexibly map graphs to different tables. In this way, graphs that are often accessed together may be stored together while graphs that are never accessed together may be stored separately (Figure 6). For example, as described below, the system metadata is stored as RDF statements in its own statement table separate from user tables. The use of multiple statement tables may improve performance through better locality and caching. It may also simplify database administration since the separate tables can be separately managed and tuned.

3.2 Optimizing for Common Statement Patterns

An RDF graph will typically have a number of common statement patterns. One source of those patterns is the RDF specification itself which defines some types and properties for modeling higher-level constructs such as bags, sequences and reification. For example, if object x is a sequence, it will have a type property with value *rdf:Seq* and one or more element properties, *_1*, *_2*, *_3*, etc. that specify elements of the sequence. A reified statement (i.e., a statement about some other statement) has a type property with value *rdf:Statement* and three properties, *rdf:subject*, *rdf:predicate*, *rdf:object* for values of the statement's subject, predicate and object.

The other source of common patterns is the user data. Applications typically have access patterns in which certain subjects and/or properties are accessed together. For example, a graph of data about persons might have many occurrences of objects with properties name, address, phone, gender that are referenced together. Using knowledge of these access patterns to influence the underlying database storage structures can provide a performance benefit. Techniques for detecting patterns in user data and in RDF query logs are reported in [16].

Jena1. In Jena1, the commonly-occurring case of reified statements was handled as a special-case. Rather than storing four separate triples for each reified statement, it stored the reified subject, predicate and object in the regular statements table, with two additional columns to indicate its reified state and to store a statement identifier. Since a statement had only one identifier, it could only be reified once. For

1. In Jena1 and Jena2, tables include a column for the graph identifier; this is not shown.

Jena2, changes were required to conform with the revised RDF specification that allows multiple reified instances of any statement.

Jena2 Property Tables. Jena2 will provide a general facility for clustering properties that are commonly accessed together. A Jena2 property table is a separate table that stores the subject-value pairs related by a particular property. A property table stores *all* instances of the property in the graph, i.e., that property does not appear in any other table used for the graph. For properties that have a maximum cardinality of one, it is possible to efficiently cluster multiple properties together in a single table. A single row of the table would store those property values for a common subject. For example, in Dublin Core, it may be beneficial to create a property table for the three properties dc:title, dc:publisher, dc:description if these properties are frequently accessed together. The use of such a property table for the data in Figure 3 is presented in Figure 4.

Multi-valued properties may be clustered or may be stored in a separate table. For example, dc:creator might be stored in a multi-valued property table containing two columns, one for the subject and one for dc:creator. Alternatively, it might be stored with the same property table as title, publisher and description although this may be less efficient if it results in many null-valued columns for a row. At first glance, it may seem that multi-valued property tables offer little benefit. However, there may be benefits to clustering values if they are frequently accessed together, e.g. a set of values that is searched as a lookup table. Note that property tables offer a small storage savings because the property URI is not stored in the table, and for clustered property tables, the subject is only stored once.

For some properties, the datatype of the object value will be fixed and known. It may be specified as a property range constraint. Property tables can leverage this knowledge by, when possible, making the underlying database column for the property value match the property type¹. This may enable the database to better optimize the storage and searching of the column.

Jena 2 Property-Class Tables. A property-class table is a special kind of property

Statement Table

Subject	Predicate	Object
mylib:doc1	dc:creator	HP Labs - Bristol
mylib:doc1	dc:creator	Hewlett-Packard

DC Properties Table

Subject	Title	Publisher	Description
mylib:doc1	Jena2	-	101
201	Jena2 Persistence	com.hp/HPLaboratories	-

FIGURE 4. Dublin Core Property Table

table that serves two purposes. It records all instances of a specified class, i.e., resources that have that class. It also stores properties of that class, i.e., each property in the table must have the class as its domain. Thus, a property-class table has two or more columns: one for the subject resource, a second boolean column indicating if the subject has been explicitly asserted as a class member (as opposed to inferred as a member), and zero or more columns for property values.

It is worth noting that Jena2 implements reification as a property-class table (Figure 5). The properties are `rdf:subject`, `rdf:predicate`, `rdf:object` and the class is constrained to be `rdf:Statement`. The subject of the property-class table is the URI that reifies the statement. Storing a reification this way saves space compared to the alternative of explicitly storing the four statements of a reification. Note that partially reified statements are easily supported.

Applications specify the schema for a graph, i.e., the property, property-class and statement tables at graph creation time through the configuration meta-graph. To simplify the implementation, once defined, the table configuration cannot be altered. However indexes may be added or removed. In the future, some limited changes to the table configuration may be enabled.

4.0 Jena2 Persistence Architecture

An overview of the Jena2 persistence architecture was presented in Figure 1. In this section, we describe the implementation of that architecture, including the specialized graph interface that implements RDF sub-graphs and the database drivers that access the database on their behalf.

4.1 Specialized Graph interface

The Jena2 persistence layer presents a Graph interface to the higher levels of Jena, supporting the usual Graph operations of add, delete and find (Figure 6). Each logical graph is implemented using an ordered list of specialized graphs; each of which is optimized for storing a particular style of statement. For example, in the figure the first logical graph is implemented using three specialized graphs: one optimized for reified statements, another optimized for ontology triples and a third which han-

Reified Statement Table

StmntURI	Subject	Predicate	Object	Type
mydir:alice	mylib:doc1	dc:title	Jena2	rdf:Statement
mydir:bob	mylib:doc2	-	Jena2	-

FIGURE 5. Reification as a Property-Class Table

1. Not all XSD types correspond to an SQL datatype.

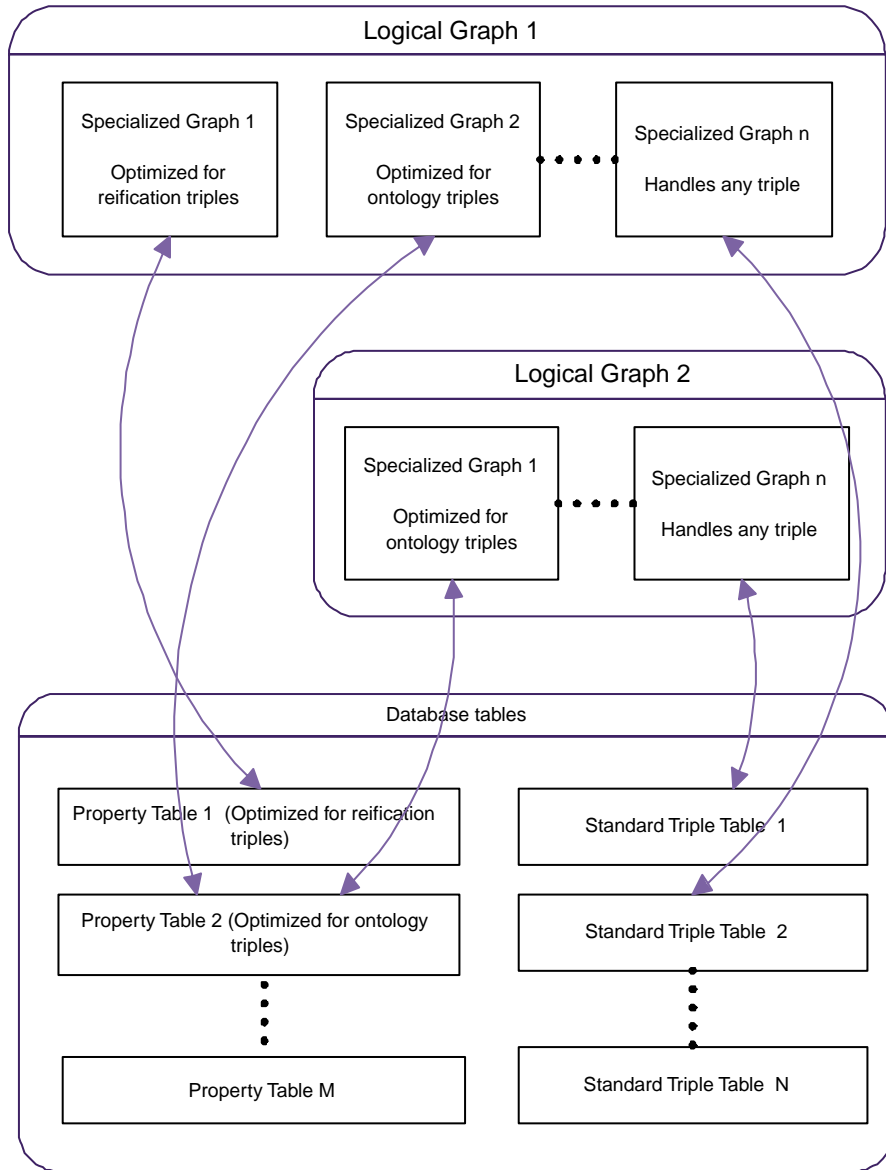


FIGURE 6. Graphs Comprise Specialized Graphs Over Tables

dles any remaining triples.

An operation on the entire logical graph, such as add statement, delete statement or find, is processed by invoking add, delete, find on each specialized graph, in turn. The results of the individual operations are combined and returned as the

result for the entire graph.

Note that this process can be optimized because, in certain cases, an operation can be completely processed for the entire graph by one specialized graph. Thus, the operation need not be invoked on the remaining specialized graphs. For example, a specialized graph that stores every statement with a property of `dc:title` can process all add and delete operations that reference `dc:title` and can fully satisfy any request to find all such properties. To support this optimization, each specialized graph operation returns a completion flag to indicate if the operation has been completely processed and the ordering of the specialized graphs is kept constant.

In the case of a find operation, an additional optimization, which the specialized graphs permit, is to evaluate the find on each graph in a lazy fashion; using resources from later specialized graphs only if the application is still hungry after consuming results from earlier graphs.

Each specialized graph maps the graph operations onto appropriate tables in the database. In the present implementation, there is a many-to-one mapping between specialized graphs and database tables. In some cases, this allows the overhead of each database table to be amortized across several graphs.

4.2 Database Driver

The database driver provides an abstract storage interface that insulates the specialized graphs from differences in how database engines support blobs, nulls, expressions, table and index creation, etc. There is a generic driver implementation for SQL databases and engine-specific drivers for Postgresql, MySQL, Oracle, etc. The engine-specific drivers override the generic methods as necessary, e.g., for different quoting conventions or treatment of blobs.

The driver is responsible for data definition operations such as database initialization, table creation and deletion, allocating database identifiers. It is also responsible for mapping graph objects between their Java representation and their database encoding. For data manipulation, the drivers use a combination of static and dynamically generated SQL. The static SQL is used for fixed, predefined operations such as inserting a triple in a graph or the various forms of the find operation. For access to property-class tables and for RDQL queries, the drivers dynamically generate SQL select statements. To reduce the overhead of query compilation, the driver layer maintains a cache of prepared SQL statements.

The driver uses a storage abstraction that is designed to be mapped to other persistent stores. Non-SQL drivers are also possible. In the future, we plan to support a Berkeley DB driver and a native-Java persistent store.

4.3 Configuration and Meta-graphs

In Jena1, database configuration parameters and options were specified in a configuration file of property-value pairs that was read when initially connecting to the database. In Jena2, the files are not used. Instead, configuration parameters are

specified as RDF statements. This is analogous to storing metadata for relational databases in tables. A graph containing configuration parameters may be passed as an argument when creating a new persistent graph. Jena2 provides default graphs containing the default configuration parameters for all supported databases.

Associated with each Jena2 persistent store is a meta-graph, a separate, auxiliary RDF graph containing metadata about each logical graph. This auxiliary graph includes the configuration parameters and options mentioned above as well as other metadata such as the date the database was formatted, the version of the driver, a list of graphs stored in the database, the mapping of graphs to tables, etc. The meta-graph may be queried just as any other Jena graph but, unlike other graphs, it may not be modified and it does not support reification.

The default schema for a graph is a statement table and a reified statement table, implemented as a property-class table. The user-provided meta-graph may specify that graphs share tables. The meta-graph may also specify additional property, property-class tables and indexes. Parameters such as the threshold size for long literals and resources are also specified as statements within the meta-graph.

5.0 Jena2 Query Processing

There are two forms of Jena querying. The find operation returns all statements satisfying a pattern. A find pattern has the form (S,P,O) where each element is either a constant or a don't-care. An RDQL query [17] is compiled into a conjunction of find patterns that may include variables to specify joins. It returns all possible valid bindings of the variables over statements in the graph¹.

The addition of property tables significantly complicates query processing. Consider iterators. Unlike a statement table where each row corresponds to a single RDF statement, an iterator over a property table may need to expand a row into multiple statements and add URIs for properties that are not explicitly stored. In addition, columns in a property table can be null.

However, the major complexity occurs when a query references an unknown property, i.e., where the property is a don't-care or a variable that will be bound when the query is processed. These cases are discussed below.

5.1 Find Processing

In Jena1, a find pattern was evaluated with a single SQL select query over the statement table. In Jena2, this has to be generalized because there can be multiple statement tables for a graph. To evaluate a pattern in Jena2, the pattern is passed, in turn, to each specialized graph handler for evaluation, stopping when the completion flag is set. The results are concatenated and returned to the application. This handles the

1. Querying over inferred graphs is not addressed here.

case when the pattern contains an unspecified property, i.e., a don't-care (note that find operations do not have variables).

Currently, each specialized graph issues a separate database query for the pattern. We plan to investigate if a single database query over all specialized graphs would be more efficient. For example, suppose the pattern is (*,*,*), which retrieves all statements, and suppose the graph has two tables, a statement table and a reified statement table. Rather than two separate queries, the following single SQL query could be used to process the pattern.

```
Select t.subject, t.predicate, t.object from stmt_table t
Union
Select r.URI, "rdf:subject", r.subject from reif_table r
       where r.subject is not null
Union
Select r.URI, "rdf:predicate", r.predicate from reif_table
       r where r.predicate is not null
Union
Select r.URI, "rdf:object", r.object from reif_table r
       where r.object is not null
Union
Select r.URI, "rdf:type", r.type from reif_table r where
       r.type is not null
```

Such queries can quickly become unwieldy for complicated patterns and several statement tables and may cause challenges for query optimizers. In addition, it is not clear that a single, large union query is more efficient than the alternative of issuing two separate queries. With the single, union query, rows in the reification table are read four times.

5.2 RDQL Processing

In Jena1, an RDQL query is converted into a pipeline of find patterns connected by join variables. The query is then evaluated in a nested-loops fashion in Jena by using the result of a find operation over one pattern to bind values to variables and then generating patterns for new find operations. It would be more efficient if the join could be pushed into the database engine for evaluation.

This is a goal of Jena2 query processing, i.e. converting multiple triple patterns into a single query to be evaluated by the database engine. A full discussion of query processing is beyond the scope of the paper. In this section, we discuss two simple cases and mention the difficulties for the general case.

For the first simple case, assume that the find patterns for a query reference only the statement table, i.e., it can be determined a priori that statements in the property tables match none of the patterns. As mentioned above, a single pattern can be completely evaluated by a single SQL query over the statement table. To evaluate multiple patterns, it is sufficient to associate a table alias with each pattern and perform a join across the aliases for linking variables in the find patterns. For

example, consider an RDQL query to get the authors of a paper. It requires two patterns, each of which has an associated SQL evaluation expression.

```
Pattern 1: (Var1, dc:title, "Jena2")
Pattern 2: (Var1, dc:creator, Var2)
Select p1.subject, p2.object
from stmt_table p1, stmt_table p2
where p1.predicate = "dc:title" and p1.object="Jena2" and
      p2.predicate="dc.creator" and p1.subject = p2.subject
```

The second simple case occurs when all patterns can be completely evaluated by a single property table. The interesting thing about this case is that it is possible to eliminate joins if the patterns reference single-valued properties clustered together. For example, suppose there is a clustered property table for dc:title and dc:creator (assume creator is single-valued here). Then, the two patterns in the previous example require only a single table alias and can be evaluated without a join.

```
Select p1.subject, p1.creator from dcPropertyTable p1
where p1.title="Jena"
```

When the find patterns for a query apply to multiple tables, it is more difficult to construct a single SQL query to satisfy all patterns. This presents the same issues as generating a single SQL query for a find operation. The current approach in Jena2 is to partition the patterns into groups, where each group contains patterns that can be completely evaluated by a single table, plus one additional group containing patterns that span tables. A SQL query is then generated for the former groups and the latter group is processed using the nested loops approach as in Jena1.

In Jena2, there are three cases in which a pattern may span tables. First, the property may be a don't-care in which case all tables must be searched. Second, the property may refer to an unspecified class, i.e., the property is *rdf:type* but the object value (the class) is not specified. In this case, it is impossible to know which property table may contain values for the pattern. Third, the property may be a variable. This is the most interesting case as it corresponds to table variables in relational database querying processing, i.e., the table name is unknown until the query is processed. This is a difficult query processing problem.

Finally, a feature of Jena2 is that queries may span graphs. This is done by specifying the graph to which a pattern applies. If the graphs reside in the same database instance, it is possible to optimize those query patterns as if they were all part of the same graph. If the graphs reside in different instances or different database engines, no attempt is made to optimize the query and the basic nested-loop approach is applied.

6.0 Miscellaneous Topics

Jena2 Performance Toolkit. To explore various layout options and understand

performance trade-offs, a set of Jena utility programs are under development. The first is an RDF synthetic data generator that generates statements for a specified number of classes and instances. Uniform and skewed data distributions can be generated as well as predefined reference patterns for properties, such as trees (for taxonomies, ancestor relations, etc.).

The second tool is a benchmark suite to measure the effectiveness of Jena2 enhancements and to compare different database layouts. It is designed as a general framework that can be used to make comparative runs of an arbitrary set of Jena programs. The third tool is an RDF data analysis tool that, when applied to a set of RDF statements, suggests potentially beneficial property and property-class tables to store the statements [16].

Jena Transaction Management. In Jena1, the underlying database may or may not support transactions. Consequently, all Jena API methods were atomic to ensure database consistency. In addition, transaction *begin*, *commit* and *abort* methods were available to declare explicit transactions when desired. Jena2 provides the same capabilities. However, there is an interesting case in which Jena cannot ensure database consistency. The Jena2 query handler supports queries across graphs. If the graphs are stored in separate databases, then a consistent read-set for the query cannot be guaranteed because a Jena2 transaction applies to a single database connection.

In principle, it should be possible to open an XOpen/XA distributed transaction connection to the other data source to ensure consistency. However, in the open world of the semantic web, the common case is that data sources do not support transactions, let alone the XA protocol. This suggests that a richer transaction interface for Jena2 is needed and it remains future work.

Bulk Load. A goal of Jena2 was to significantly reduce the time to load persistent graphs compared to Jena1. This is a critical issue if RDF is to be applied to very large datasets. The use of a denormalized schema helps address this problem since a typical Jena2 add operation updates fewer tables than Jena1. Jena2 also includes support for JDBC2 batch operations which enable multiple JDBC statements to be passed in one call to the database engine. The value of batching depends on the level of optimization within the database engine but in any event it reduces the number of database calls significantly.

7.0 Status and Future Work

Performance Notes. Preliminary performance measurements indicate that the denormalized schema of Jena2 is faster than the normalized schema of Jena1, twice as fast for many operations. The results of one simple retrieval experiment are presented in Table 1. The test retrieved a single, 200-byte property value for 1000 randomly selected objects. The test was run under two configurations. The denormalized configuration stored the property value directly in the statement

table. The normalized configuration reduced the long literal threshold (see Section 3.1) to 100 bytes which caused the property value to be stored in the literals table.

Thus, retrieving the property value in the denormalized configuration requires two retrievals, one for the statement table and a second for the literals table while the denormalized case requires only one. Each configuration was run multiple times with different random seeds and the result of the first and final run are presented. The times are in milliseconds and the tests were run using MySQL under WinXP on a recent generation PC workstation with 1.5GB RAM.

The large reduction in run time for the initial run compared to the final run we attribute to hardware cache effects. For the warm run, as expected, the denormalized retrieval is twice as fast as the normalized retrieval. If the schema were completely normalized so that the subject and predicate were also stored in separate tables as was done in Jena1, we would see an even greater speed-up for the denormalized schema. A more systematic study will be done upon completion of a Jena

TABLE 1. Retrieval Times for Normalized vs. Denormalized Literal

Number of Retrievals	Normalized	Denormalized	Speed-up
1000 (initial run)	3270	2850	1.2
1000 (final run)	840	420	2.0

performance toolkit. Similarly, the database size increase due to the denormalized schema has not been studied pending impletion of URI prefix compression.

Next we provide some preliminary results that show the value of property-class tables for reification. A synthetic database of 10,000 reified RDF statements was generated and stored in two different formats. In the first case, the reified statement was stored in an optimized form as a property-class table. In the second case, the reified statement was stored unoptimized as RDF triples, i.e., each reified statement was stored as four RDF statements. Consequently, the first table contained 10,000 rows while the second table contained 40,000 rows.

Then a simple test program randomly selected a reified statement and retrieved the four reification triples for that statement (recall that on retrieval, the property-class table converts each table row to a set of triples). Each test was run four times with different random number seeds and three different test sizes were run of 200, 1000, 5000 retrievals. The results are presented in Table 2. As before, the times are in milliseconds and the tests were run using MySQL under WinXP on a recent generation PC workstation with 1.5GB RAM.

Our expectation was that the optimized format would perform anywhere between one and four times faster than the unoptimized form since it only needs to invoke the database engine once to get all four triples whereas the unoptimized format makes four calls. For a small number of retrievals, the optimized format shows a large improvement between the first and fourth run. We attribute this to caching effects that decrease with larger numbers of retrievals. It is interesting to see that the speed-up for large numbers of retrievals exceeds our expectations. This may be

TABLE 2. Retrieval Times for Four Triples of a Reified Statement

Number of Retrievals	Optimized	Unoptimized	Speed-up
200 (initial run)	1000	1860	1.8
200 (final run)	270	1470	5.4
1000 (initial run)	1330	7380	5.5
1000 (final run)	700	6970	10.0
5000 (initial run)	4220	34380	8.1
5000 (final run)	3470	34270	9.9

due to database caching effects. Since the optimized table is smaller, it is possible to cache a larger percentage of the entire table which reduces the number of relatively slow disk seek operations.

A comprehensive study of RDQL query processing has not been done. Some preliminary analysis indicates that the Jena2 algorithm is a modest improvement over the Jena1 nested-loops approach. The Jena1 algorithm works quite well for queries with high selectivity since such queries require few nested find operations. For such queries, Jena1 and Jena2 perform about the same. Jena2 performs better than Jena1 on queries which join a large number of tuples.

Future Work. Currently, Jena2 stores all literals as strings. An important enhancement for typed literals will be to store them as native SQL types. This will enable inequality comparisons and range queries to be processed within the database engine. This is future work.

A major goal of Jena2 is support for OWL and reasoning. Now that this is available, it will be interesting to explore how the persistence layer can better support these capabilities, e.g. performing transitive closure within the database.

We are presently investigating caching strategies to improve performance. One approach is to implement the caching inside the persistence layer using the specialized graph interface. An alternative is to implement caching for arbitrary logical graphs. The latter provides a convenient general-purpose solution, while the former may make use of intimate knowledge of the database to improve performance. Our initial caching algorithm is to implement a write-through cache which holds statements with commonly-used subjects. If the cache holds one statement with subject=X then it has every statement with subject=X. Currently, the cache assumes exclusive access to that subject to avoid cache consistency issues due to conflicting updates from other Jena applications. However, such exclusive access appears to be a common case. This style of cache has previously been suggested by others with experience in using RDF with Jena1 [11] and we hope will prove to be a good match for common application usage patterns. Testing and analysis is underway.

8.0 Related Work

A good introduction to RDF storage subsystems and a comparative review of

implementation is available in [4,5]. We do not attempt to duplicate such a survey here. However, if we compare the Jena2 persistent store to some of these systems along the dimensions of database schemas, architecture, and system functionality, then we can better characterize the strengths and limitations of our approach.

The Jena2 schema design is unique in that it supports two basic schema types: both a denormalized schema used for storing generic triple statements as well as property tables to store subject-value pairs related by arbitrarily specified properties. To the best of our knowledge, no other system supports the generation of property tables based on arbitrary properties; other systems are strictly schema-specific. Jena2 uses the arbitrary property tables to implement a novel architecture where the statements associated with a given graph are stored in multiple specialized subgraphs. This architecture enables the Jena2 query processor to effectively treat the subgraphs as data partitions and provides an efficient implementation for reification.

Most systems (including KAON [9,], Parka Database[13], and rdfDB[14]), support only a fixed set of underlying tables that implement a (non-schema-specific) generic store. This means that the storage mechanism cannot adapt to the data characteristics, impacting scalability.

ICS-FORTH's RDF Suite [10] supports both generic stores as well as automatically-generated schema-specific Object-Relational (SQL3) schema definitions. However, unlike Jena2, RDF Suite relies on schema specifications to create the specialized tables; it doesn't support arbitrary property tables. Similarly, the Sesame [15] system creates one specialized table per class. Tightly coupling the table layout to schema structure can facilitate inferencing by allowing the systems to exploit the explicit schema relationships, but it also means that the tables must be rebuilt whenever the schema structure changes. This forces the storage system to forfeit RDF's unique support for flexible dynamic schema restructuring; Jena2 is not subject to this limitation.

Insofar as the schema-specific tables partition the stored data, such schema-specific storage resembles the Jena2 notion of specialized subgraphs. However, because these systems tightly couple the subgraphs with the schemas, they can only partition data according to its syntactic structure; they cannot create subgraphs based on other factors. The Storage and Inference Layer (SAIL) [15] provides layered interfaces to Sesame modules that stack and allow actions to be passed between them until handled. However, because it based upon Sesame, the SAIL database schema is class-specific, and thus subject to the limitations listed above.

To the best of our knowledge, no other RDF system optimizes storage for reification in the style of Jena2. The notion of property-class tables appears to be new in RDF stores although it is commonly used in object and functional database systems.

9.0 Conclusions

The Jena2 persistence layer supports application-specific schema while retaining the flexibility to store arbitrary graphs. The notion of property-class tables appears to be new and should be beneficial for query languages that expose higher-level abstractions to applications. However, the mixing of property tables and statement tables in a graph database complicates query processing and optimization. More work is needed on efficient algorithms for this case.

Acknowledgements

The first three authors are new to the Jena effort and wish to thank the rest of the Jena team, which includes the fourth author, for their help and for allowing us to participate in its development. The rest of the Jena team includes Jeremy Carroll, Ian Dickinson, Chris Dollin, Brian McBride and Andy Seaborne. For the work in this paper, we particularly thank Chris Dollin and Andy Seaborne for being so generous with their time.

References

1. B. McBride Jena IEEE Internet Computing, July/August, 2002.
2. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, K. Wilkinson, The Jena Semantic Web Platform: Architecture and Design, HP Laboratories Technical Report HPL-2003-146.
3. T. Berners-Lee et al. Primer: Getting into RDF & Semantic Web using N3, <http://www.w3.org/2000/10/swap/Primer.html>
4. D. Beckett, SWAD-Europe: Scalability and Storage: Survey of Free Software / Open Source RDF storage systems, http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report/
5. D. Beckett, J. Grant, SWAD-Europe: Mapping Semantic Web Data with RDBMSes, http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/
6. T. Berners-Lee, Web Services and Semantic Web, keynote speech at World Wide Web Conference, 2003, <http://www.w3.org/2003/Talks/0521-www-keynote-tbl/>
7. S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle, The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, 2nd Intl Workshop on the Semantic Web (SemWeb'01, with WWW10), pp. 1-13, Hongkong, May 1, 2001.
8. D. Reynolds, Jena Relational Database Interface – Performance Notes, in Jena 1.6.1 download: <http://www.hpl.hp.com/semweb/download.htm>
9. KAON - The Karlsruhe Ontology and Semantic Web Tool Suite, <http://kaon.semanticweb.org/>

10. J. Broekstra, A. Kampman, F. van Harmelen, Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema PDF, First International Semantic Web Conference (ISWC'02), Sardinia, Italy, June 9-12, 2002.
11. D. Banks, personal communication.
12. The ICS-FORTH RDFSuite: High-level Scalable Tools for the Semantic Web. <http://139.91.183.30:9090/RDF/>
13. PARKA-DB - A Scalable Knowledge Representation System.
14. <http://www.guha.com/rdfdb/internals.html>
15. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. <http://sesame.aidadministrator.nl/publications/del10.pdf>
16. L. Ding, K. Wilkinson, C. Sayers, H. Kuno, Application-Specific Schema Design for Large RDF Datasets, HP Laboratories Technical Report HPL-2003-170.
17. A. Seaborne, An RDF NetAPI, , HP Laboratories Technical Report HPL-2002-109.

An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays

Akiyoshi MATONO¹, Toshiyuki AMAGASA¹, Masatoshi YOSHIKAWA², and
Shunsuke UEMURA¹

¹ Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara 630-0192, Japan
{akiyo-ma, amagasa, uemura}@is.aist-nara.ac.jp

² Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya-shi 464-8601, Japan
yosikawa@itc.nagoya-u.ac.jp

Abstract. The Semantic Web is a candidate for the next generation of the World Wide Web. It is anticipated that the number of metadata written in RDF (Resource Description Framework) and RDF Schema will increase as the Semantic Web becomes popular. In such a situation, demand for querying metadata described with RDF and RDF Schema will also increase, and therefore effective query retrieval of RDF data is important. To this end, we propose an indexing scheme for RDF and RDF Schema. In our (proposed) scheme, we first extract four kinds of DAGs (Directed Acyclic Graphs) from an RDF data, and extract all path expressions from the DAGs. Then, we generate four kinds of suffix arrays based on the path expressions. Using the indices, we can achieve efficient processing of query retrievals on RDF data including schematic information defined by RDF Schema (for example, classes and/or properties).

1 Introduction

The Semantic Web [1, 2] has emerged as the next generation of the World Wide Web. In the Semantic Web, human-to-machine and machine-to-machine interactions are expected to become more intelligent from the wealth of metadata associations between resources on the Internet. The key difference between the current Web and the Semantic Web is the quality and quantity of metadata. Currently available metadata are insufficient, in terms of quality and quantity, for the purposes of advanced processings. The Semantic Web, on the other hand, makes it possible to perform high-level processes, such as reasoning, deduction, and semantic searches, to make the best use of metadata associated with web resources.

In the Semantic Web, RDF (Resource Description Framework) [3] and RDF Schema [4] are commonly used to describe metadata. RDF is a framework to describe data and their semantics, and is composed of the RDF model and RDF syntax. In the RDF model, *statements* are used to describe relationships between pairs of terms. A *statement* is called a triple, because a statement is comprised of three elements: a resource, a property and a value. The value can be either literal or resource, and thus complex information can be represented as a set of statements, such as a form of directed graphs. RDF

syntax is a specification to serialize RDF statements as XML (Extensible Markup Language) data. RDF Schema is the schema language for RDF used to specify schematic information, such as definitions of resources, properties and classes.

In the near future, the quantity of metadata represented by RDF is expected to increase significantly as the Semantic Web comes into wide use. We expect that RDF databases will become important as an efficient means of access to massive metadata bases written in RDF and RDF Schema. One naive approach to constructing RDF databases is to use XML databases to store and retrieve RDF data simply because any RDF data can be represented in XML. However, this approach is not practical because the structure of RDF data is different from the structure of XML data, and there are many ways to serialize RDF data in XML form. Thus, queries to retrieve RDF data cannot be implemented as queries of their XML representations.

Another way to implement RDF databases is to utilize relational databases. In this approach, a piece of RDF data is decomposed and stored into relational tables. Several methods have been proposed already [5]. RDFSuite [6] is an implementation of RQL (RDF Query Language) [7], a query language for RDF. To store RDF data, RDFSuite uses tailor-made relational schema specially designed for the RDF Schema that we would like to explore. Jena [8] is an RDF database that implements RDQL (RDF Data Query Language) [9] using MySQL. However, a few of the previously mentioned works has investigated the performance of RDF databases.

We propose an indexing scheme for RDF and RDF Schema to achieve efficient query retrieval. Specifically, we focus on path expressions extracted from RDF and RDF Schema. Our first step is to extract four types of partial graphs from RDF and RDF Schema, because RDF and RDF Schema data have four distinct relationships. The graphs represent relationships among instances, classes and properties. Then, we extract all possible path expressions from the graphs, and construct suffix arrays on the path expressions. As a result, for a given query as partial path expression, we can efficiently detect the result.

The basic concept underlying our proposal is similar to that of Yamamoto et al. [10]. The main difference is that this approach is used for XML data, whereas we propose applying it to RDF and RDF Schema. Since XML data is a tree structure, enumeration of all possible path expressions in XML is an easy task. However, path expression cannot be as straightforward with RDF and RDF Schema, because they may contain multiple paths and/or cycles. For this reason, we will limit our first targets to cases where RDF and RDF Schema do not contain cycles.

However, even if we limit our target to DAGs, we should claim that our scheme can be applicable to many applications due to the fact that a large majority of RDF data in real applications is expressible as DAGs. For instance, WordNet [11], a famous on-line lexical database written in RDF, does not contain cycles based on our investigation. Following this step we will introduce a method to cope with cycles.

We have implemented our approach and evaluated its performance in a series of experiments. We used four kinds of RDF documents with different sizes using Wordnet [11], and stored each of the four RDF documents in RDFSuite. Eight queries were executed against the RDF database to compare the processing time using our index and

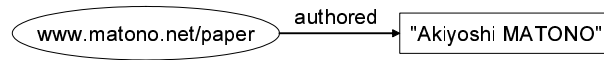


Fig. 1. An RDF data model statement.

those of non-index (or original indices of RDFSuite). Our index was more efficient than the non-index, and our approach has shown scalability.

The rest of this paper starts with an outline of RDF and RDF Schema using examples in Section 2. In Section 3, we describe our approach for efficient RDF retrieval. In particular, we defined a suffix array for DAG, explained about extracting the four DAGs from the RDF data, and described path expressions for each DAG. In addition, we describe our experimental setup and evaluate the performance using our index in Section 4. In Section 5, we describe an idea to cope with cycles. We discuss related work in Section 6, and conclude the paper in Section 7.

2 An Overview of RDF

RDF (Resource Description Framework) [3] is a foundation for representing and manipulating metadata on Web resources. RDF enables us to implement various applications, such as resource discovery, interoperation of metadata and description of machine-understandable information.

In RDF specification, the data model and its syntax are defined. In addition, RDF Schema [4] is used to describe schematic information of RDF data.

RDF can be used to describe the metadata of any resource in the Net as long as its location is identifiable using a URI (Uniform Resource Identifier) [12]. In RDF, “statements” are used to represent binary relationships between two distinct (or maybe identical) resources. Complex information can be represented by a set of statements. Thus, an RDF document is modeled as a directed graph (DG), where a resource corresponds to a vertex and a relation corresponds to an arc. For example, let us take a look at the statement “this paper is authored by Akiyoshi MATONO.” The statement consists of three parts, namely, a subject (“This paper”), a predicate (“is authored by”) and an object (“Akiyoshi MATONO”). For this reason, the statement is also called a triple. We call the relation represented by a statement the “predicate relation” (Figure 1).

For the purpose of exchanging metadata written in RDF, RDF syntax, by which we can serialize RDF data into XML data, is defined. Figure 2 shows an RDF document corresponding to the above example.

RDF Schema is used to give semantic information to RDF data. Specifically, RDF Schema makes it possible to specify the properties of a resource, data type of a property, class memberships of properties, and class hierarchies.

Using RDF and RDF Schema, we can represent complex information (Figure 3). Classes and properties defined by RDF Schema are shown in the upper part. For example, the property “creates” takes an “Artist” and an “Artifact” as its domain and range, respectively. “Sculptor” is a subclass of “Artist”, and so on. Resource descriptions can

```

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://www-db.aist-nara.ac.jp/~akiyo-ma/test.rdfs#">
  <rdf:Description about="www.matono.net/paper">
    <s:authored>Akiyoshi MATONO</s:authored>
  </rdf:Description>
</rdf:RDF>

```

Fig. 2. An RDF document.

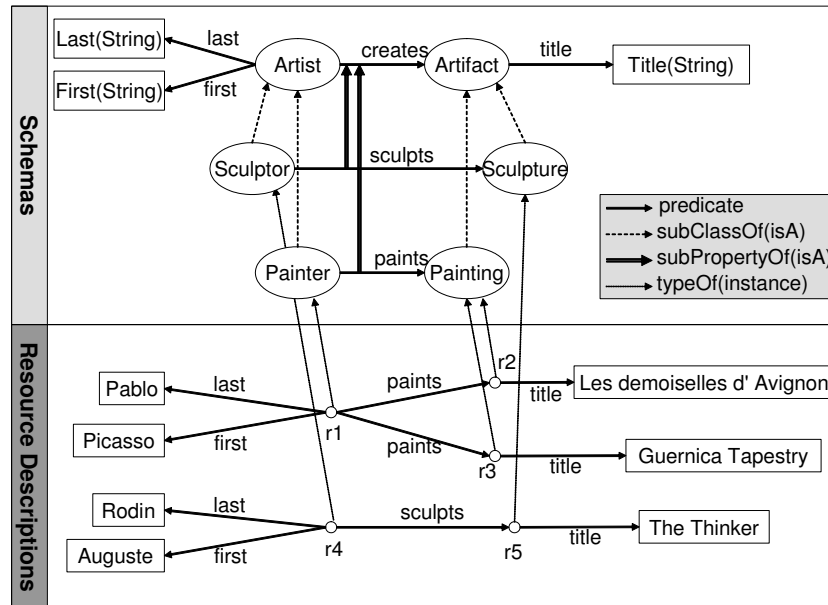


Fig. 3. A complex example using RDF and RDF Schema.

be found in the lower part. Resources, such as “r1” and “r2”, are defined as instances of classes. Consequently, resource “r1”, for example, has three properties, “last”, “first” and “paints”, which are inherited from the “Artist” class. Resources as character strings, such as “Pablo” and “Picasso”, are instances of the Literal class in RDF Schema.

3 Efficient RDF Data Retrieval using Suffix Array for DAGs

3.1 Problem description

Basically, queries on RDF data can be expressed as combinations of some path expressions based on graph structures of RDF data. For example, the query “find all resources that are created by artists” can be constructed as follows: 1) find all artists, and 2) for each artist, find all resources that are reachable by following “create” property. As we can see, both steps, 1) and 2), can be processed on the basis of path expressions of RDF

graphs. We can therefore say that efficient processing of path expressions is crucial to achieve efficient RDF data retrieval.

In fact, this is similar to XML data retrieval, and many researchers are devoted to efficient XML query processing based on path expressions [13, 10, 14]. Both XRel [13] and XParent [14] propose a relational schema based on path expressions for efficient storage and retrieval of XML data into a relational database. In Yamamoto et al. [10], an indexing scheme based on path expressions is proposed. In this approach, all path expressions are extracted from XML data first. Then, a suffix array is constructed on the extracted path expressions where the occurrences of element (or attribute) names are alphabets. As a consequence, we can efficiently find any (partial) path expression using the full-text search functionality provided by the suffix array.

However, we cannot apply the above technique to RDF data, because of the differences between RDF and XML data. The differences can be summarized as follows; 1) RDF data may contain cycles, whereas XML data does not. This comes from the topology of RDF graphs, that is, an RDF data forms a directed graph and an XML data forms a tree. Extracting all possible path expressions from an RDF data is not trivial, consequently. 2) In RDF data, not only vertexes but also arcs have labels, whereas arcs are not labeled in XML data. Thus, we need to take care in path expressions. 3) We need to take schematic information provided by the RDF Schema, because we think that the query which involves schematic information is general on the Semantic Web.

3.2 Proposed method

In this paper, we propose a novel indexing technique based on suffix arrays for efficient retrieval of RDF data. The basic idea behind our approach is similar to that of Yamamoto et al. [10]. In order to cope with the above problems, we made the following modifications.

1. To cope with problem 1), we first limit out target to RDF data of DAGs (Directed Acyclic Graphs), that is, we assume that RDF graphs do not contain cycles. Thus, we can extract all possible path expressions from a DAG. Then, we construct suffix arrays on the path expressions. To this end, we newly introduce a suffix array for DAGs, which is an extension of suffix arrays for character strings. In fact, the proposed scheme can be adapted to the cases of general directed graphs. The algorithm will be shown later in Section 5.
2. To cope with problem 2), we define a path expression as an alternation of labels of vertexes and labels of arcs. In addition, we introduce special symbols to make a distinction among classes, properties and literals.
3. To cope with problem 3), we extract four kinds of subgraphs, namely, predicates in schema, predicates in resource descriptions, class inheritance and property inheritance graphs, from an original RDF graph. Then, we construct four kinds of suffix arrays for each subgraph. As a consequence, queries including schematic information can be processed by a collaboration of these suffix arrays. To answer such queries that include both schema and instance (e.g. find the titles of Paintings painted by the instances of Painter class), we first get the instances of the "Painter" class using class inheritance graph. We then get the titles of "Paintings" painted

by the instances of “Painters” using predicates in resource descriptions. Finally, we merge the answers and obtain the final result. In this way, complicated queries can be processed using our proposed indexing scheme.

3.3 Extracting DAGs from RDF data

Given an RDF data with RDF Schema, we extract four kinds of DAGs by taking vertex types, arc types and their semantics into account.

Predicates in schema This graph is obtained by extracting classes and their properties from the schema part of an RDF graph. This graph may contain cycles.

Predicates in resource descriptions This graph is obtained by extracting resources and their properties from the resource description part of an RDF graph. This graph may contain cycles.

Class inheritance This graph is obtained by extracting classes and “subClassOf” arcs connected to the classes. Note that “subClassOf” arcs do not have labels, and this graph does not contain cycles.

Property inheritance This graph is obtained by extracting properties and “subPropertyOf” arcs in the schema part of an RDF graph, and thus we let properties, which are arcs in the original graph, be vertexes in this subgraph. Note that “subPropertyOf” arcs do not have labels, and this graph does not contain cycles.

These subgraphs, except for predicates in the resource descriptions graph, cannot be obtained if RDF Schema is not provided. In those cases, we just use predicates in the resource descriptions graph. Otherwise, we can make full use of schematic information to query RDF data.

3.4 Path expressions

Figure 4 shows the syntax, represented in EBNF (Extended Backus-Naur Form), for path expressions. In the figure, `schemaPath`, `instancePath`, `classPath` and `propertyPath` correspond to path expressions extracted from predicates in schema, predicates in resource descriptions, class inheritance and property inheritance subgraphs, respectively. In the path expressions, ‘>’ is used as a separator. Additionally, some special prefixes, ‘#’, ‘+’ and ‘\$’, are used to distinguish classes, properties and resources. If these special symbols are used in labels, we replace their occurrence with an entity reference of XML for encapsulation. For example, the RDF data in Figure 1 can be represented as

```
#www.matono.net/paper > +authored > "AkiyoshiMATONO"
```

based on the definition.

For a given DAG, we can extract all possible path expressions using the algorithm shown in Figure 5. This algorithm starts with the vertexes whose in-degree are zero (0), and search for traversable paths in a depth first manner.

```

paths      ::= schemaPath* | instancePath* |
            classPath* | propertyPath*
schemaPath ::= (classVertex '>' propertyVertex '>')*
            classVertex
instancePath ::= (resourceVertex '>' propertyVertex '>')*
            literalVertex
classPath  ::= (classVertex '>')* instanceVertex
propertyPath ::= (propertyVertex '>')* propertyVertex
classVertex ::= '#' typeName
propertyVertex ::= '+' propName
instanceVertex ::= resourceVertex | literalVertex
resourceVertex ::= '$' URI-reference
literalVertex ::= '"' literal '"'
typeName      ::= see [3]
propName      ::= see [3]
literal       ::= see [3]
URI-reference ::= see [3]

```

Fig. 4. Path expression syntax (EBNF).

3.5 Suffix array for DAGs

An ordinary suffix array is a data structure for full-text search on documents constructed on one-dimensional character strings. Given a text data, all suffixes are extracted and sorted in lexicographical order. Any substring can then be detected by performing a binary search on the array of suffixes. In addition, because any suffix can be represented by an integer (an indexing point), the array of suffixes can be implemented as an array of integers whose size is equal to the length of the original document.

When applying a suffix array on path expressions, we need an extension that allows a suffix array to accommodate multiple path expressions. For this reason, we use a pair of integers as an indexing point; The first number is for representing an identifier of a path expression, and the other is for representing an indexing point within the path expression. It is defined as follows:

Definition 1 (Suffix array for DAGs) *Let G be a directed acyclic graph (DAG), $V(G)$ be the set of vertexes in G , and $E(G)$ be the set of arcs in G . Arc $e = (u, v)$ in $E(G)$ is represented by a pair of vertexes $u, v \in V(G)$, and u and v are called the “source” and “destination,” respectively. In addition, let $R \subset V(G)$ be a set of vertexes whose in-degree is equal to zero (0), and $L \subset V(G)$ be a set of vertexes whose out-degree is equal to zero (0). We call R and L the “roots” and “leaves,” respectively.*

Given a path on G from a root $s_{t,1} \in R$ to a leaf $s_{t,2k_t-1} \in L$, it can be represented as $p_t = s_{t,1} \cdot s_{t,2} \cdot \dots \cdot s_{t,2k_t-2} \cdot s_{t,2k_t-1}$, where:

- t is the identifier of the path,
- k_t is the length of the path,
- $s_{t,2h-1} \in V(G)$ ($1 \leq h \leq k_t$), and
- $s_{t,2h} = (s_{t,2h-1}, s_{t,2h+1}) \in E(G)$ ($1 \leq h \leq k_t - 1$).

```

var roots := a set of vertexes whose in-degree is 0
var stack : Stack
foreach start ( roots ) begin
  createPath ( start )
end
function createPath ( start : vertex ) : Void
var end : vertex
var arcs : set of arcs
var triple : tuple of (vertex, arc, vertex)
begin
  arcs := a set of arcs connected from start vertex
  foreach arc ( arcs ) begin
    end := a vertex connected from arc
    triple := ( start, arc, end )
    stack.push ( triple )
    createPath ( end )
    stack.pop()
  end
  end
  Creating a path expression based on stack
end

```

Fig. 5. An algorithm for extracting path expressions from DAGs.

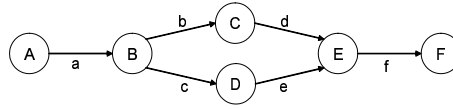


Fig. 6. A simple DAG.

A suffix of $p_j = s_{j,1}.s_{j,2}.\dots.s_{j,2k_j-1}$ is defined as $S_{j,i} = s_{j,i}.s_{j,i+1}.\dots.s_{j,2k_j-1}$ ($i = 1, 2, \dots, 2k_j-1$), whose indexing point is $a_{j,i} = [j, i]$.

The suffix array $S(p_j)$ of the path p_j is then defined as an array of indexing points that is sorted in lexicographical order.

The suffix array of a directed acyclic graph G is an array of indexing points, using all paths from roots $\{u|u \in R\}$ to leaves $\{v|v \in L\}$, that is sorted in lexicographical order, and duplicated occurrences of the suffixes are eliminated. \square

We will demonstrate how a suffix array is constructed on a DAG using a simple example (Figure 6). From the DAG, we can extract two paths, namely, “A.a.B.b.C.d.E.f.F” and “A.a.B.c.D.e.E.f.F.” Then, we assign indexing points to them (Figure 7), sort them in lexicographical order, and eliminate duplicates of identical suffixes (Figure 8). As a result, we obtain the suffix array of [1,1] [2,1] [1,3] [2,3] [1,5] [2,5] [1,7] [1,9] [1,2] [2,2] [1,4] [2,4] [1,6] [2,6] [1,8].

When processing queries, we perform binary searches on the suffix array. For this reason, $O(\log_2(n+1))$ of computational complexity is required.

	1	2	3	4	5	6	7	8	9
1	A	a	B	b	C	d	E	f	F
2	A	a	B	c	D	e	E	f	F

Fig. 7. Suffixes of paths.

A.a.B.b.C.d.E.f.F : (1, 1)	(1, 1) : A.a.B.b.C.d.E.f.F
a.B.b.C.d.E.f.F : (1, 2)	(2, 1) : A.a.B.c.D.e.E.f.F
B.b.C.d.E.f.F : (1, 3)	(1, 3) : B.b.C.d.E.f.F
b.C.d.E.f.F : (1, 4)	(2, 3) : B.c.D.e.E.f.F
C.d.E.f.F : (1, 5)	(1, 5) : C.d.E.f.F
d.E.f.F : (1, 6)	(2, 5) : D.e.E.f.F
E.f.F : (1, 7)	(1, 7) : E.f.F
f.F : (1, 8)	(2, 7) : E.f.F
F : (1, 9)	(1, 9) : F
A.a.B.c.D.e.E.f.F : (2, 1)	(2, 9) : F
a.B.c.D.e.E.f.F : (2, 2)	(1, 2) : a.B.b.C.d.E.f.F
B.c.D.e.E.f.F : (2, 3)	(2, 2) : a.B.c.D.e.E.f.F
c.D.e.E.f.F : (2, 4)	(1, 4) : b.C.d.E.f.F
D.e.E.f.F : (2, 5)	(2, 4) : c.D.e.E.f.F
e.E.f.F : (2, 6)	(1, 6) : d.E.f.F
E.f.F : (2, 7)	(2, 6) : e.E.f.F
f.F : (2, 8)	(1, 8) : f.F
F : (2, 9)	(2, 8) : f.F

Fig. 8. Sorting and deletion of suffixes.

4 Performance Evaluation

This section evaluates the performance of the proposed scheme in a series of experiments.

4.1 Experimental setup

Datasets We used RDF and RDF Schema documents of Wordnet [11] as the experimental data. Wordnet is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept.

As far as we have investigated, the RDF data of Wordnet does not contain any cycles, and thus we can apply our scheme directly to the datasets. We created sub-documents of them with different sizes 500 KB (Type A), 1 MB (Type B), 2 MB (Type C) and 4 MB (Type D). Table 1 shows the details of the datasets.

Table 1. Details of RDF documents of Wordnet

Type	A	B	C	D
Number of RDF Schema documents	1	1	1	1
Number of RDF documents	4	4	4	4
Total size of RDF Schema documents (KB)	4	4	4	4
Total size of RDF documents (KB)	513	999	2,073	3,982
Number of elements and attributes	15,089	29,542	62,565	119,368
Number of classes in RDF Schema documents	6	6	6	6
Number of properties in RDF Schema documents	5	5	5	5
Number of resources in RDF documents	1,555	3,100	6,571	12,380
Number of properties in RDF documents	5,647	10,851	22,773	42,878
Number of literals in RDF documents	4,553	8,645	18,107	33,473

Table 2. Performance evaluation queries

Queries for predicates in schema	
#1 <code>+glossaryEntry>#</code>	Retrieval of classes for a given property
#2 <code>#LexicalConcept>+</code>	Retrieval of properties
#3 <code>#LexicalConcept>+antonymOf>#LexicalConcept>+hyponymOf>#LexicalConcept>+</code>	A long path expression
Queries for predicates in resource descriptions	
#4 <code>+hyponymOf>#</code>	Retrieval of objects for a statement
#5 <code>#&wn;400062583>+wordForm>#</code>	Retrieval of statements
#6 <code>#&wn;100033830>+similarTo>#&wn;100033153>+wordForm>#</code>	A long path expression
Queries for class inheritance	
#7 <code>#Adjective>\$</code>	Retrieval of instances
#8 <code>#Resource#LexicalConcept>#Adjective>#AdjectiveSatellite>\$</code>	A long path expression

Query sets The query expressions used in the experiments are shown in Table 2. In the table, “&wn;” is a character entity reference for representing the namespace of Wordnet. Using these queries, we intend to evaluate the following aspects: queries for predicate relations in schematic information (#1-#3); queries for predicate relations among instances (#4-#6); and queries for inheritance relations among classes (#7 and #8).

Methodology We used an RDF database, RDFSuite [6], as a basis for implementing our (proposed) scheme. RDFSuite is implemented on top of PostgreSQL, an open source relational database management system. Specifically, RDFSuite supports two kinds of relational schemas, *GenRepr* and *SpecRepr*, for storing RDF data. *GenRepr* has two relational tables; *Resources* is for storing resources and their identifiers, and *Triples* is for storing triples extracted from statements. On the other hand, *SpecRepr*’s relational schema is designed according to the RDF Schema of the RDF data being stored. In our experiments, we used *SpecRepr* because it is more efficient than *GenRepr* from the view point of performance.

Table 3. The number of path expressions and arrays of index-points

Description	A	B	C	D
# paths (total)	9,709	19,480	43,008	90,058
# suffixes (total)	25,977	51,060	111,108	217,549
# paths (preds in schema)	10	10	10	10
# suffixes (preds in schema)	21	21	21	21
# paths (preds in resource descs)	8,144	16,370	36,427	77,668
# suffixes (preds in resource descs)	19,409	37,999	83,459	165,512
# paths (class inheritance)	1,555	3,100	6,571	12,380
# suffixes (class inheritance)	6,547	13,040	27,628	52,016
# paths (property inheritance)	5	5	5	5
# suffixes (property inheritance)	10	10	10	10

We compared the query processing time between RDFSuite and RDFSuite powered by our indexing scheme as follows:

1. We store each dataset in RDFSuite based on SpecRepr schema. Then, we construct suffix arrays on the relational tables of RDFSuite. Specifically, a table for storing all path expressions extracted from Wordnet data, and four tables for storing indexing points are created in the relational database.
2. We then measure the query processing time of the queries in Table 2 for the two cases, pure RDFSuite and RDFSuite powered by suffix arrays.

We used a PC with an Athlon 1.1 GHz CPU and 768 MB memory running RedHat Linux 8.0, and used Java 1.4.1 for the implementation.

4.2 Experimental results

Table 3 shows the statistical data of the generated suffix arrays. From the table, we can observe that the number of path expressions and suffixes increase in proportion to the sizes of the datasets for the cases of “predicates in resource descriptions” and “class inheritance.” However, this is not the case for “predicates in schema” and “property inheritance,” because this information solely depends on RDF Schema, and RDF Schema is fixed for the experiments in this paper.

Figure 9 shows ratios (N/I) of the processing time of RDFSuite (N) to our scheme (I). That is, our approach is about four times faster than RDFSuite with respect to query #2 for dataset A. It is clear that our scheme outperforms RDFSuite.

Table 4 shows the details of the processing times. Note that for the case of #1 – #3, because the dataset is small, the absolute processing times are too short, and the results may not be reliable compared to other results.

Our scheme can process #4 – #6 in almost the same time, whereas RDFSuite does not. In particular, #4 is slower than others (#5 and #6). This is because #4 searches objects for a given predicate, while #5 and #6 search objects for a given pair of subject and predicate. For this reason, RDFSuite can make use of built-in indices to process the queries.

Table 4. Processing time

Type Suffix array	A		B		C		D	
	Yes	No	Yes	No	Yes	No	Yes	No
#1	30.7	30.0	25.6	27.0	25.9	28.2	28.4	28.2
#2	27.6	109.0	26.0	97.3	26.0	96.0	28.0	92.0
#3	23.2	164.7	24.3	162.6	24.8	180.4	25.1	158.0
#4	99.5	396.5	130.6	707.3	205.9	1274.9	337.9	2325.8
#5	82.8	166.6	113.6	217.4	182.7	357.5	312.3	541.9
#6	84.7	182.1	108.0	231.6	178.0	385.9	300.7	646.4
#7	71.4	237.6	80.3	334.2	114.1	702.7	142.3	1238.7
#8	77.0	263.0	91.3	447.7	122.0	579.6	160.7	816.9

When processing queries for inheritance between class and instance (#7 and #8), as the data size is large, the ratios of the processing time between our scheme and RDFSuite are larger. In other words, our scheme achieved scalability.

5 Coping with Cycles

In this paper, we limited our targets within directed acyclic graphs. We think that we can find many other RDF data without cycles, because even a large scale data like Wordnet does not contain cycles. Consequently, our scheme can be used for many applications. However, some RDF data with directed graph structures with cycles also exist. Thus, it is important to be able to cope with cycles in order to widen the applications of our scheme.

5.1 Path expressions extraction and index construction

When applying indices based on suffix arrays for querying graphs, we need to extract all possible path expressions beforehand. However, the previous algorithm for extracting path expressions cannot cope with graphs that include cycles, because it may not terminate due to dissatisfaction of terminal conditions. For this reason, we made some improvements to the algorithm so that it can extract all the vertexes and arcs thoroughly.

The algorithm in Figure 12 has two features as follows: 1) if a path expression contains two (or more) identical vertexes, a *loop-stamp(s)* is put on their second (and later) occurrence; and 2) we change our strategy to decide the starting positions of the path expressions. Actually, we make a list of vertexes whose in-degrees are equal to zero (0), followed by vertexes ordered by the differences between the out-degrees and in-degrees in ascending order. Starting from these vertexes, we try to enumerate path expressions until all the vertexes and arcs are included.

Let us take a look at such an example. Figure 10 illustrates a graph including a cycle, and Figure 11 shows two path expressions extracted from the graph using the algorithm in Figure 12. Note that ‘^’ is the *loop-stamp*.

We then create suffixes with respect to those path expressions and sort them in lexicographic order. Finally, we get the following suffix array: [2,1] [1,1] [2,3] [1,3]

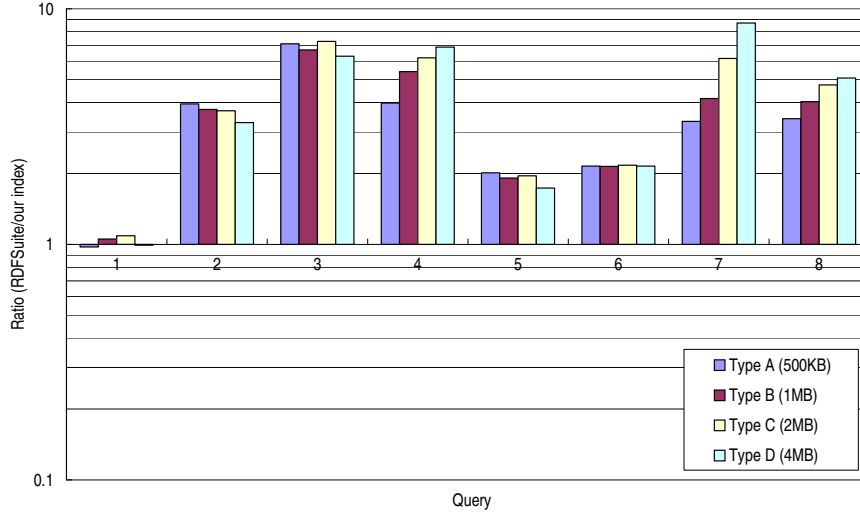


Fig. 9. Processing time (RDFSuite / Suffix array)

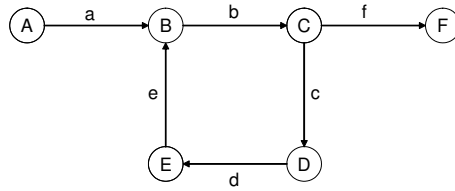


Fig. 10. Directed graph including a cycle

[2,5] [1,5] [2,7] [2,9] [1,7] [2,11] [2,2] [1,2] [2,4] [1,4] [2,6] [2,8] [2,10] [1,6], whose length is 18.

5.2 Query processing

When processing queries based on path expressions against a graph with cycles, handling unintended termination of the path expressions is crucial. That is, path expressions listed in a suffix array are not powerful enough to express cycles because of the limitation of their expressiveness. As a consequence, we may come to the end of such a termination when we are matching a query key and a path expression in the index, and may thus miss correct answers.

If a query #E>+e>#B>+b>#C>+c>#D is given, we cannot find the same occurrence in the suffix array, although it is a correct answer. When processing this query, we

	1	2	3	4	5	6	7	8	9	10	11
1	#A	>a	#B	>b	#C	>f	#F				
2	#A	>a	#B	>b	#C	>c	#D	>d	#E	>e	#B

Fig. 11. Path expressions and indexing points of the suffixes

start from the starting element ($\#E$) and proceed as much as possible as usual. Then, we get the indexing point $[2,9]$ ($\#E>+e>\#B$). This intermediate result partially patches until $\#E>+e>\#B$. Eventually, we come to the *loop-stamp*. Then, we decompose the query path expression here, let the following path expression ($\#B>+b>\#C>+c>\#D$) be a new query, and initiate it. As a result of the brand-new query, we get a suffix $[2,3]$ ($\#B>+b>\#C>+c>\#D>+d>\#E>+e>\#B$). Now, the initial query key is fulfilled, and we get a result of the query.

We expect that we can achieve efficient retrieval for a directed graph with cycles using the indexing scheme. However, we may have to improve the scheme, because the number of path expressions and suffixes are increasing in the case of the target data that contains many cycles.

6 Related Work

Indexing techniques for structured documents are classified into a *position-based index* and *path-based index* according to Sacks-Davis et al. [15].

The indices proposed by Kanemoto et al. [16] and Shin et al. [17] are position-based indices. Kanemoto et al. [16] proposed an approach in which four indices are combined to achieve efficient document retrieval. The indices were a *content index* for maintaining positions of elements and contents, *local structure index* for maintaining the tree structure of document instances, *global structure index* for maintaining the tree structure of document schema, and *structure meta index* for maintaining the meta information of the other indices. Although this approach was efficient, the performance did not scale with respect to data size, because four kinds of indices must be joined. In the study by Shin et al. [17], an indexing scheme called *BUS (Bottom Up Scheme)* was proposed. In this approach, document features were maintained in a bottom-up manner.

Path-based indices were proposed by Yamamoto et al. [10], Kaushik et al. [18] and Cooper et al. [19]. The study by Yamamoto et al. [10] is a basis of our proposal. In this paper, given an XML document, all possible path expressions were extracted, and suffix arrays were constructed on path expressions and reverse path expressions, and hence efficient processing of path expressions (and reverse path expressions) was achieved. In the study by Kaushik et al. [18], they created compact models from document trees by grouping similar vertexes into one vertex. Query processing was performed using path expressions on the compact models. That is, they achieved space efficient indexing by giving up accuracy. An indexing scheme called *Index Fabric*, as an extension of *Patricia trie* [20], was proposed by Cooper et al. [19]. *Patricia trie* was an efficient and compact indexing scheme that could deal with large-size text. *Index Fabric* is an extension of *Patricia trie*, and is a height-balanced indexing structure for semi-structured data,

In addition, combinations of position- and path-based indices were proposed by Sacks-Davis et al. [15] and McHugh et al. [21]. In Sacks-Davis et al.'s study [15], a position-based index was constructed as inverted lists consisting of all words and elements, and a path-based index represented the list of element names and their positions for each path. In McHugh et al.'s study [21], four indices were proposed, namely, the *Value Index*, *Text Index*, *Link Index* and *Path Index*. Value Index has pairs of values and element names. Text Index is implemented as an inverted list of text. Link Index maintains information on a list of children for each element. Finally, Path Index has path information for all elements.

Path-based indices were also used in object databases [22–24]. Similar to our approach, these approaches maintain the relationships of class hierarchies and/or object composition hierarchies. The key difference between our scheme and their approaches is that we treat path expressions as character strings, whereas the others do not.

Christophides et al. [25] have proposed a labeling scheme for efficient retrieval of RDF Schema. The study relevant to our research. They applied previously proposed labeling schemes for tree structures to RDF schema. Concretely, their approach employed the study by Agrawal et al. [26], in which an optimal spanning tree is generated from a DAG based on the number of ancestors per node, so that it can handle DAGs. The labeling schemes investigated in [25] are classified into bit vector, prefix and interval scheme. Bit vector [27] is a labeling scheme in that a node is represented by a n bits vector. Prefix scheme directly encodes the label of a node in an XML tree, in that the prefix is inherited by the parent's label followed by the order of the node in its siblings. Dewey scheme [28] is one of prefix scheme. Interval scheme [29, 26, 30] encodes the interval label (start, end) such that it is contained in its parent's interval label. Christophides et al. [25] limited the target data as RDF Schema for efficient retrieval. Making a comparison between their scheme and ours is an interesting topic. We plan to do it in the near future.

7 Conclusions

In this paper, we proposed an indexing scheme to enable RDF and RDF Schema to achieve efficient query retrievals on path expressions. To this end, we first proposed four types of partial graphs that can be obtained from RDF and RDF Schema. In addition, we proposed suffix arrays on DAGs. By applying this scheme to path expressions extracted from the above graphs, we achieve efficient RDF query processing. Because most of the RDF and RDF Schema in real applications are expected to be modeled as DAGs, we can make use of our proposed scheme. We conducted a performance study and the results showed that our approach outperformed an existing RDF database, RDFSuite.

In the future, we will try to deal with RDF data that include cycles and investigate query optimization techniques for RDF queries. Since our indexing scheme needs to precompute all paths as statical indexing data, we must consider an update of RDF data and schema.

References

1. World Wide Web Consortium: Semantic Web. <http://www.w3c.org/2001/sw/> (2001)

2. Berners-Lee, T.: What the Semantic Web can represent. <http://www.w3.org/DesignIssues/RDFnot.html> (1998)
3. World Wide Web Consortium: Resource Description Framework(RDF) Model and Syntax Specification. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/> (1999) W3C Recommendation 22 February 1999.
4. World Wide Web Consortium: Resource Description Framework(RDF) Schema Specification 1.0. <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/> (2000) W3C Candidate Recommendation 27 March 2000.
5. World Wide Web Consortium: Survey of RDF/Triple Data Stores. <http://www.w3.org/2001/05/rdf-ds/DataStore> (2001)
6. Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., Tolle, K.: The RDFSuite: Managing Voluminous RDF Description Bases (2000)
7. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: RQL: A Declarative Query Language for RDF. In: Proceedings of the eleventh international conference on World Wide Web, ACM Press (2002) 592–603
8. McBride, B.: Jena: Implementing the RDF Model and Syntax Specification. In: Proceedings of the Second International Workshop on the Semantic Web - SemWeb'2001. (2001)
9. Hewlett-Packard Company: RDQL – RDF Data Query Language. (<http://www.hpl.hp.com/semweb/rdql.htm>)
10. Yamamoto, Y., Yoshikawa, M., Umeura, S.: On Indices for XML Documents with Namespaces. In: Conference Proceedings of Markup Technologies '99, GCA, Philadelphia, U.S.A. (1999)
11. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Introduction to WordNet: An On-Line Lexical Database. <http://www.cogsci.princeton.edu/wn/> (1993)
12. Berners-Lee, T., Fielding, R.T., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax. <http://www.isi.edu/in-notes/rfc2396.txt> (1998) RFC2396.
13. Yoshikawa, M., Amagasa, T., Shimura, T., Uemura, S.: XRel: a path-based approach to storage and retrieval of XML documents using relational databases. *ACM Transactions on Internet Technology (TOIT)* **1** (2001) 110–141
14. Jiang, H., Lu, H., Wang, W., Yu, J.X.: Path Materialization Revisited: An Efficient Storage Model for XML Data. In Zhou, X., ed.: Thirteenth Australasian Database Conference (ADC2002), Melbourne, Australia, ACS (2002)
15. Sacks-Davis, R., Dao, T., Thom, J.A., Zobel, J.: Indexing Documents for Queries on Structure, Content and Attributes. In: Proceedings of the International Symposium on Digital Media Information Base, Nara, Japan. (1997) 236–245
16. Kanemoto, H., Kato, H., Kinutani, H., Yoshikawa, M.: An Efficiently Updatable Index Scheme for Structured Documents. In: Proceedings of 9th International Workshop on Database and Expert Systems Applications (DEXA'98), IEEE Computer Society. (1998) 991–996
17. Shin, D., Jang, H., Jin, H.: BUS: An Effective Indexing and Retrieval Scheme in Structured Documents. In: Proceedings of the Third ACM Conference on Digital Libraries. (1998) 235–243
18. Kaushik, R., Shenoy, P., Bohannon, P., Gudes, E.: Exploiting Local Similarity for Efficient Indexing of Paths in Graph Structured Data. In: ICDE. (2002)
19. Cooper, B., Sample, N., Franklin, M.J., Hjaltason, G.R., Shadmon, M.: A Fast Index for Semistructured Data. In: The VLDB Conference. (2001) 341–350
20. Knuth, D.E.: The Art of Computer Programming Volume 3 Sorting and Searching, Second Edition. Addison-Wesley (1998)
21. McHugh, J., Widom, J., Abiteboul, S., Luo, Q., Rajaraman, A.: Indexing Semistructured Data. Technical report, Stanford University, Computer Science Department. (1998)

22. Bertino, E.: Index Configuration in Object-Oriented Databases. *VLDB Journal* **3** (1994) 355–399
23. Lee, W., Lee, D.: Path Dictionary: A New Approach to Query Processing in Object-Oriented Databases (1995)
24. Xie, Z., Han, J.: Join Index Hierarchies for Supporting Efficient Navigations in Object-Oriented Databases. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, September 12-15, 1994, Santiago de Chile, Chile, Morgan Kaufmann (1994) 522–533
25. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the semantic web. In: *Proceedings of the twelfth international conference on World Wide Web*, ACM Press (2003) 544–555
26. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. In Clifford, J., Lindsay, B.G., Maier, D., eds.: *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, Portland, Oregon, May 31 - June 2, 1989, ACM Press (1989) 253–262
27. Wirth, N.: Type Extensions. *ACM Transactions on Programming Languages and Systems* **10** (1988) 204–214
28. Online Computer Library Center: Dewey Decimal Classification. (<http://www.oclc.org/dewey/>)
29. Dietz, P., Sleator, D.: Two algorithms for maintaining order in a list. In: *Proceedings of the nineteenth annual ACM conference on Theory of computing*, ACM Press (1987) 365–372
30. Li, Q., Moon, B.: Indexing and Querying XML Data for Regular Path Expressions. In: *The VLDB Journal*. (2001) 361–370

```

var stack : stack /* for storing triple */
var roots1 : list of vertexes /* whose in-degree = 0 */
var roots2 : list of vertexes /* sorted by values of (in-degree – out-degree)
                               into descending order – roots1 */
var roots := append ( roots1, roots2 )
foreach start ( roots ) begin
    createPath ( start )
end
function searchGraph ( start : vertex ) : Void
var end : vertex
var arcs : set of arcs
var triple : tuple of ( vertex, arc, vertex )
begin
    roots.remove ( start )
    arcs := a set of arcs connected from start
    foreach arc ( arcs ) begin
        end := a vertex connected from arc
        roots.remove ( end )
        triple := ( start, arc, end )
        /* a path expression does not include same statements. */
        if ( stack = nil or triple  $\notin$  stack.items ) then
            stack.push ( triple )
            searchGraph ( end )
            stack.pop ()
        end
    end
    var path : path expression
    var loop_stamp : loop-stamp /* for representing a path expression containing cycles */
    path := concat ( path, stack[0].start )
    for ( var i := 0; i < stack.length; i := i + 1 ) then
        var vertex := stack[i].end
        /* When a path expression has same vertexes */
        if ( vertex  $\in$  path.items ) then
            vertex := vertex + loop_stamp
        end
        path := concat( path, stack[i].arc, vertex )
    end
end

```

Fig. 12. An algorithm of creating path expressions for DG

RDF Core: A component for effective management of RDF Models

FLORIANA ESPOSITO, LUIGI IANNONE, IGNAZIO PALMISANO AND
GIOVANNI SEMERARO

Dipartimento di Informatica
Università degli Studi di Bari

Via Orabona, 4
Bari, 70125, ITALY
+39 080 544 2299

{esposito,iannone,semeraro}@di.uniba.it, ignazio_io@yahoo.it

Abstract.

In order to make Semantic Web effective, the first step was the development of languages that could support data portability, namely XML, metadata descriptions, namely RDF, and ontology management and inference, such as DAML+OIL, OWL etc. Those languages have to be manipulated by applications and many Application Programming Interfaces (APIs) have been developed in order to accomplish this task. Obviously, they differ in implementation details. Moreover, developers often would like to exploit more than an API at a time. Another issue is that a developer would be very advantaged if he could have a uniform support for some services across these frameworks (such as query languages), despite the lack of standards. In this paper, we present a component, called *RDFCore*, developed in order to overcome these problems. We will also illustrate the added value that our framework provides to RDF in order to exploit the full potentiality of the language and to employ it in research as well as in real world applications. Consequently we will provide some test results on the performances of the presented framework.

Introduction

World Wide Web Consortium (W3C), that is the main promoting committee involved in the evolution towards the Semantic Web[1], has been recently working on the development of technologies that could support this process. While some of these technologies are still in early phases, part of them can already be exploited in real world applications. This is the case of Resource Description Framework (RDF). It represents the basic support to write metadata on Web resources and to grant interoperability among heterogeneous applications when exchanging these metadata. RDF describes resources in terms of primitives (classes, properties, resources, etc.) without taking into account the description structure itself. In fact, the description can be encoded in XML (but also in other different formats, see for instance [2]). This ensures its portability across the Web.

Moreover, RDF represents a suitable solution to implement the Semantic Web vision also because it presents three key features:

- **Extensibility.** Each user can add its own description extending pre-existing ones without any limit.
- **Interoperability.** RDF descriptions can rely on XML serialization every time they need to be exchanged among heterogeneous platforms
- **Scalability.** RDF descriptions can be viewed as sets of three field records (triples) (Subject, Predicate and Object). This makes them easy to fetch and manage even when a single description holds many triples in it.

Many Application Programming Interfaces (APIs) have been developed in order to support RDF-based applications. They offer a lot of useful features, ranging from efficient persistence and powerful query languages [8] to simple and well designed object models [4]. That is why we felt the need for a uniform framework (RDFCore) that will be presented in the following sections. The main aim of RDFCore is granting the widest compatibility with existing RDF APIs, exploiting their advantages in a transparent way for users and, where possible, enhancing traditional approaches to RDF-based development.

RDF Core

Overview

In the following section we will describe a framework named RDFCore and, besides its features, we will also point out how the problems related to RDF have been tackled.

RDFCore main components: Managers

The architecture sketched below (Figure 1) shows the main components of the RDFCore Framework.

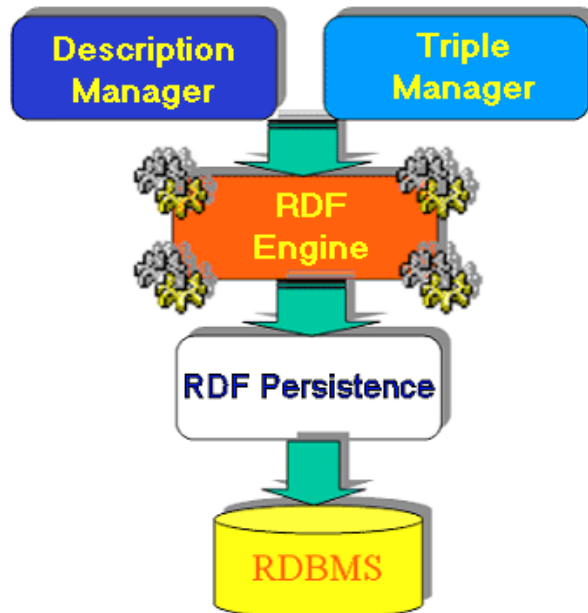


Figure 1 RDFCore Architecture

RDF Descriptions can be seen as sets of statements (typically called *Models*). Each statement is a triple compound by a subject, a predicate and an object. Therefore, users access RDF resources at two different levels of granularity – *Models* and *Statements*. That is why we developed two different entities, called *Description Manager* and *Triple Manager*, that deal with all the possible operations on

Descriptions and on *Triples*, respectively. Therefore, as far as *Descriptions* are concerned, users can:

- Add/Delete, Retrieve a *Description* to/from their own repository
- Update an entire *Description* with a new one
- Query a *Description* or a bunch of them.

while *Triple Manager* offers all the typical operations on single statements or on sets of statements (as subsets of a *Description*) like:

- Add
- Delete
- Update

All these operations would seem quite obvious. Indeed, all the most famous APIs currently available offer similar support to RDF users (see for instance Jena RDF Toolkit [3] or Stanford RDF APIs [4]). However, all these operations within our framework bring with themselves a slight advantage.

First of all, RDFCore has been devised as a multi-user environment. In fact, each user owns its own repository of RDF resources. Furthermore, users can be arranged in groups, can share resources with other members and there is the possibility of establishing policy rights on operations involving shared resources, such as granting/removing read/write access for a particular user or group of users. Other APIs do not offer a well-constructed persistence model like this one. The usefulness of such user management is strictly related to resource authoring. As a matter of fact, if the scenario is the WWW we could easily foresee communities of Web resource authors that generate, along with the actual web-resource, its description in RDF (no matter whether this generation will be automatic or not). Therefore, the need of having such an organisation of the RDF resources would soon arise.

RDF Engine and RDF Persistence

Description Manager and *Triple Manager* make up the sole user interface of RDFCore and they both rely on the RDF Engine module (see Figure 1).

In the RDFCore architecture, RDF Engine represents a specification rather than a concrete piece of software. In fact, it enumerates all the necessary operations for the

upper modules to properly carry out their functionalities. Actually, each call to the business functions of the proper *Manager* is translated into a combination of RDF Engine operations.

In the previous sections, we mentioned that there are many existing APIs to manage RDF and we also pointed out that it is strongly desirable that users can have the possibility to exploit features of any of them without switching architecture. That is why RDF Engine specifies which operations are required and nothing else. The responsibility of actual implementation of the services specified by RDF Engine is delegated to RDF Persistence level components.

In this way, a well-known best practice in Object-Oriented design, that is the *implementation* of abstract interfaces, can be exploited. In practice, RDF Engine is an interface whose implementation can vary depending on the requirements developers want to meet.

Therefore, many RDF Engine implementations can co-exist in a single instance of RDFCore. A typical scenario would be one in which different kinds of users have different implementations of the underlying RDF Engine. The advantage is that some users could need some requirements that are provided (for instance) within some specific persistence. The only effort in order to meet those requirements is to build up an implementation of the RDF Engine that acts as a bridge between that persistence and the upper level components (*Managers*). A more concrete example will be provided below in the description of the applications of our framework.

Actually, two implementations of RDF Engine have been produced, based on two different solutions for RDF Description storage/retrieval:

- An implementation based on RDF/XML serialization
- An implementation based on triple storage, built on Jena Toolkit API [3]

Both of them, as well as the upper components, comply to the well-known Stanford RDF API [4] as a standard for RDF object model, since it is the most widespread basic API for RDF Description management. This is accomplished by means of establishing that the input/output parameters in the modules interface have types taken from the RDF API object model (such as *Model*, *Statement*, *Resource* etc.)

Exploitation of RDFCore: COLLATE

One of the most complete exploitation case studies for our framework takes place in the EU research project COLLATE (IST-1999-20882) [5]. It belongs to the Fifth Framework Programme in scientific European Community research programme, under the Information Society Technology category, Key Action III: “Multimedia Content and Tools”. The focus of this project is the development of a collaborative system for scientists involved in the study of the film production in Austria, Germany and Czech republics in the 30s. Three film archives have to be made electronically available (in order, above all, to preserve very fragile and intangible material) and scientists have to be allowed to index, catalogue and annotate such assets in order to build scientific discourses on their work among the scientific community endorsed with COLLATE [6], [7]).

This could be easily assimilated to the wider scenario foreseen by Semantic Web: a huge quantity of resources (documents, assets) with many relationships among them.

COLLATE requirements are:

- A uniform way of identifying resources (films, film related documents, cataloguing and indexing information, scientist annotations, scientific discourses)
- Distribution of information; in fact, archives still keep their resources in a decentralized architecture in order to avoid the moving of huge amount of data, both physically and electronically (for obvious reasons)
- Intelligent navigation through data and metadata, including navigation across scientific discourses on resources

For all these reasons RDF is a straightforward solution since it holds in itself the features we underlined in the introductory sections.

We go on examining which added value our framework provides to COLLATE. It is quite obvious that a huge collection of documents and metadata such as COLLATE heritage needs a careful devising of a scalable component in order to manage storage and retrieval of both resources and relationships among them. While the solution for the former problem is delegated to efficient RDBMS, as far as the latter we developed a suitable RDF Persistence for granting scalability to RDFCore framework. This module relies on Jena Toolkit storage model for RDF. It consists in exploiting a

relational representation of the RDF triples (subject, predicate, object) stored in a database. This approach takes advantage of the outstanding performance rates of the most famous RDBMS (such as Oracle, MySQL and PostgreSQL). One of the most immediate benefits is the fact that applications need not to load in-memory RDF *Models* (Descriptions) in order to deal with small portions of them (typically small sets of *Statements*), saving lots of memory and time for each operation.

Moreover, Jena Toolkit offers RDF Description Query Language (RDQL [8]) as language for querying RDF Descriptions. This support has been extended for querying multiple *Models*, that together with multi-user environment and scalability, proved to be a suitable solution for COLLATE requirement.

The query language, however, remains a weakness point of all RDF APIs available, including Jena. At the time of writing, still no standard query language specifications are available. This hampers the interoperability between components and, therefore, between different systems; in other words, two systems using different APIs to manage RDF can exchange data, but cannot easily exchange queries on these data.

To address this issue, RDFCore embeds a subcomponent, called Enhanced Query Engine, able to deal with different query languages. The design of this component exploits the Strategy pattern [10] (like other components in RDFCore architecture), enabling the use of a dynamic set of query languages. In order to add the support for a new query language, only the classes implementing the interfaces to wrap the parser of the language and the query engine are needed, allowing for easy update. This update, obviously, can be the standard query language the W3C (together with other organizations) is working on, as soon as it is available¹.

Empirical evaluation of performances

In this section we present some results from a preliminary empirical evaluation we carried out on the RDFCore software components. We mainly tried to investigate one of the key features that a framework devoted to Web (and Semantic Web) development should have: scalability. The notion of scalability is very well known in IT environment and it can be measured with respect to many variables. Being

¹ <http://www.daml.org/dql/>

basically a knowledge storage system, RDFCore needs to be scalable, firstly with respect to the amount of data that it has to manage. Therefore, tests that have been carried out had the purpose of investigating how smoothly RDFCore performances decreased as the data size increased. Particularly, our aim was to have a component showing linear scalability with data size, i.e. time doubles as data size doubles.

In the previous sections, while describing the design of RDF persistence architecture, we pointed out that our framework could provide simultaneously different strategies for the actual data storage thanks to the persistence architectural layer of abstraction. Indeed, as we mentioned in the previous section, we developed two different persistence mechanisms, respectively:

- Based on file system binary storage of RDF/XML resources, relying on a compressed XML storage format (namely PDOM²)
- Based on RDBMS storage of RDF resources, relying on Jena API for RDF.

We prepared two different test sets, both devised in order to progressively scale up in data size but with slightly different strategies. The first one increases data in size but not in content, by simply repeating the basic RDF description n times in the same document. The second one has been created by adding new statements to the starting description without repeating any object, subject or properties. In this way all triples in the descriptions from the second test set are different from each other, while there is a lot of redundancy in the first test set. The reason for doing that is that in both RDF persistence implementations some mechanism to take advantage from redundancy has been devised (e.g.: indexing of URI). Therefore an RDF description with many repetitions should be processed in lesser time than a variegated description.

In all our tests, the descriptions named $NNNx_rdf$ are redundant descriptions, where NNN is the number of times a particular triple is replicated in the description; on the other hand, the descriptions named $OutputNNNNN_rdf$ are descriptions with no redundancy, and $NNNNN$ is the number of triples in the particular model.

² http://www.infonyte.com/en/prod_pdom.html

Obtained results

In Table 1 and Table 2, divided for the sake of readability, we show the results of processing the first test set (highly redundant) with an RDFCore exploiting the file system-based persistence that we mentioned before, and with the JENA-based persistence, relying on the MySQL RDBMS. In Figure 2 and Figure 3 (for PDOM), and subsequently Figure 4 and Figure 5 (for JENA), we show the growth of required time to store descriptions compared with a theoretical linear function on data size (used as baseline). In these figures, as well as in the subsequently ones, the scale on the Y axis is logarithmic. Where not specified, the measuring unit for time is the millisecond. Table 3 reports the results obtained on the redundancy-free test set, while Figure 6 and Figure 7 provide a graphical representation of them. Notice that missing values (- in tables) were omitted because they have been considered irrelevant.

		PDOM Persistence			JENA Persistence		
File	File size (Kbytes)	Elapsed time (milliseconds)	Theoretical elapsed time	PDOM file size	Reading time	Storing time	Theoretical storing time
2x.rdf	173	3886	4000	-	203	9777	10000
3x.rdf	259	4016	6000	-	250	14772	15000
4x.rdf	342	4226	8000	-	313	19779	20000
5x.rdf	432	4446	10000	-	453	24767	25000
6x.rdf	518	4827	12000	-	485	29787	30000
7x.rdf	605	4547	14000	-	563	34787	35000
8x.rdf	691	5178	16000	-	640	39766	40000
9x.rdf	777	4757	18000	-	734	44822	45000
10x.rdf	864	5417	20000	-	875	49822	50000
11x.rdf	950	5117	22000	-	953	54762	55000
12x.rdf	1036	5168	24000	-	1125	59783	60000
13x.rdf	1122	5007	26000	-	1062	64747	65000
14x.rdf	1209	5488	28000	-	1250	69827	70000
15x.rdf	1295	5427	30000	-	1234	74727	75000
16x.rdf	1381	5948	32000	-	1312	79837	80000
17x.rdf	1468	5728	34000	-	1422	84737	85000
18x.rdf	1554	5808	36000	-	1547	89737	90000
19x.rdf	1640	5879	38000	-	1547	94687	95000
20x.rdf	1727	5929	40000	-	1656	99682	100000

Table 1 High redundancy test (a)

		PDOM Persistence			JENA Persistence		
File	File size (Kbytes)	Elapsed time (milliseconds)	Theoretical elapsed time	PDOM file size	Reading time	Storing time	Theoretical storing time
20x.rdf	1727	5929	40000	-	1656	99682	100000
30x.rdf	2590	7411	60000	261	2390	149573	150000
40x.rdf	3453	9043	80000	314	3250	199394	200000
50x.rdf	4316	10104	100000	370	4015	249230	250000
60x.rdf	5179	10905	120000	427	4781	299171	300000
70x.rdf	6042	11636	140000	482	5578	348987	350000
80x.rdf	6905	13930	160000	532	6453	398823	400000
90x.rdf	7768	13450	180000	584	7203	448658	450000
100x.rdf	8631	14130	200000	638	9437	498494	500000
110x.rdf	9494	15292	220000	691	9406	548403	550000
120x.rdf	10357	15793	240000	744	10343	598239	600000
130x.rdf	11220	18807	260000	797	11032	648011	650000
140x.rdf	12083	13450	280000	848	11688	697917	700000
150x.rdf	12946	22272	300000	900	12594	747847	750000
160x.rdf	13809	21802	320000	952	13469	797643	800000
170x.rdf	14672	23384	340000	1003	14469	847636	850000
180x.rdf	15535	24105	360000	1055	15469	897452	900000
190x.rdf	16398	25317	380000	1106	17438	947425	950000
200x.rdf	17261	26201	400000	1157	16891	997201	1000000

Table 2 High redundancy test (b)

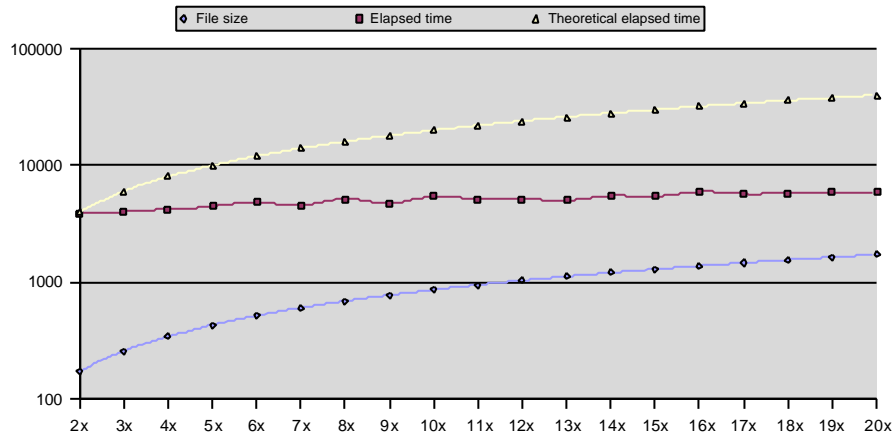


Figure 2 High redundancy test (PDOM) (a)

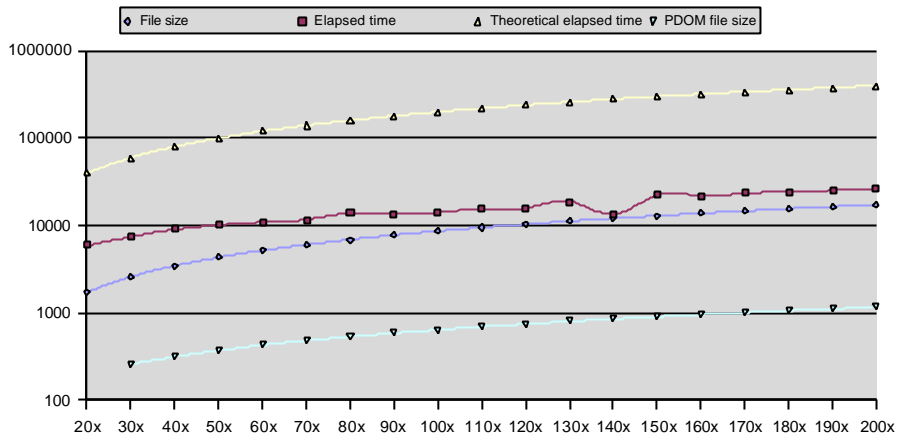


Figure 3 High redundancy test (PDOM) (b)

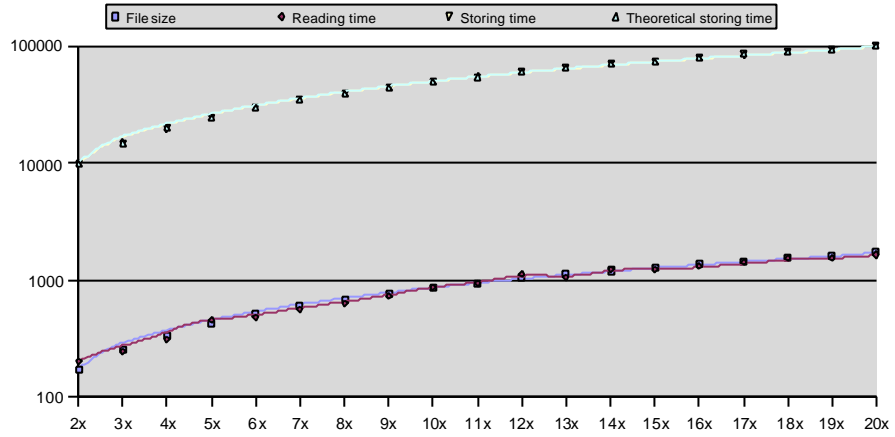


Figure 4 High redundancy test (JENA) (a)

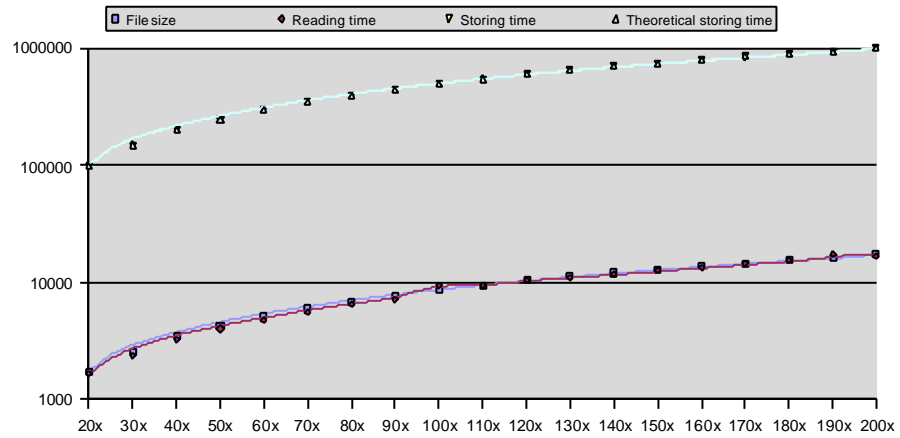


Figure 5 High redundancy test (JENA) (b)

File	PDOM Persistence					JENA Persistence		
	File size	PDOM file size	Reading time	Storing time	Theoretical storing time	Reading time	Storing time	Theoretical storing time
Output10000	1480	1210	6990	15382	15000	2219	83612	80000
Output20000	2990	2470	10404	26689	30000	3140	167201	160000
Output30000	4490	3700	15682	36823	45000	4797	250750	240000
Output40000	6000	4970	19999	48139	60000	6125	334200	320000
Output50000	7510	6210	26178	59776	75000	7828	418035	400000
Output60000	9000	7450	29893	76700	90000	9422	501715	480000
Output70000	10500	8700	34089	99152	105000	13281	585204	560000
Output80000	12000	9920	38305	145219	120000	15328	667835	640000
Output90000	13500	11100	45174	208650	135000	16531	752483	720000
Output100000	15000	12400	49812	308293	150000	18438	836212	800000

Table 3 No redundancy test

The X axis in Figure 6 and Figure 7 reports the number of triples in the files used for the test.

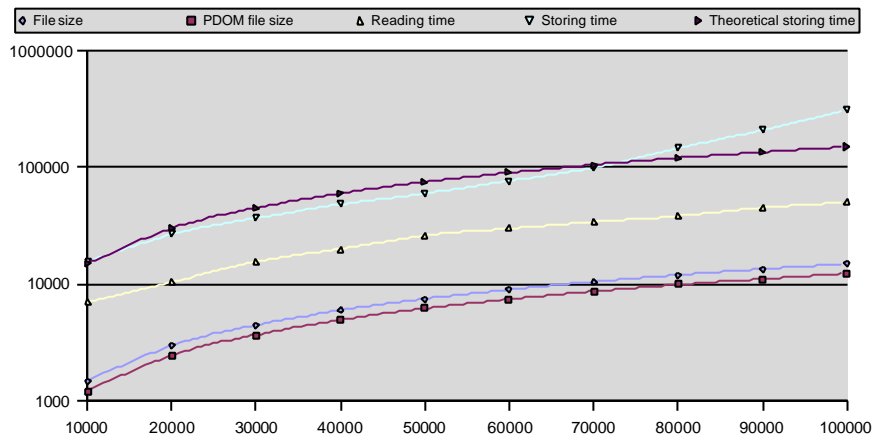


Figure 6 No redundancy test (PDOM)

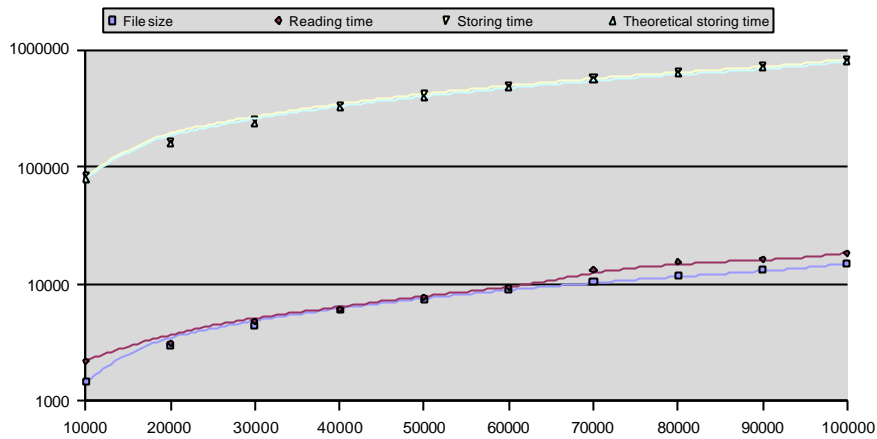


Figure 7 No redundancy test (JENA)

Table 4 reports RDFCore performances in adding a statement to very huge descriptions that have been already stored in the repository. Figure 8 and Figure 9 show the graphic trend of required time.

PDOM Persistence			JENA Persistence		
File	Elapsed time	Theoretical elapsed time	File	File size	Elapsed time
160x.rdf	9333	9333	Output10000	1480	358
170x.rdf	9564	9916	Output20000	2990	12
180x.rdf	10826	10500	Output30000	4490	25
190x.rdf	10756	11082	Output40000	6000	70
-	-	-	Output50000	7510	36
-	-	-	Output60000	9000	10
-	-	-	Output70000	10500	17
-	-	-	Output80000	12000	21
-	-	-	Output90000	13500	20
-	-	-	Output100000	15000	20

Table 4 Add triple test

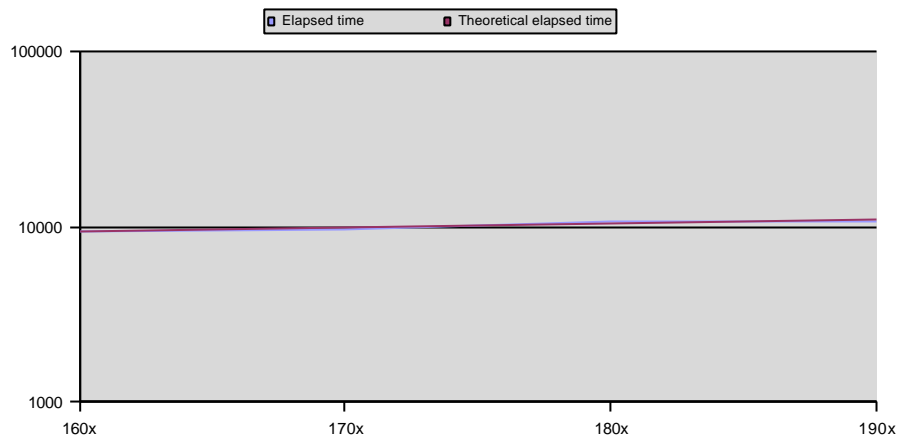


Figure 8 Add triple test (PDOM)

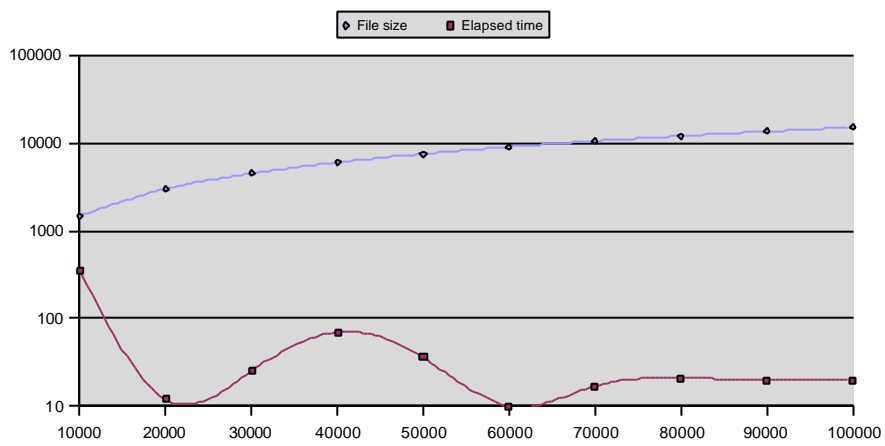


Figure 9 Add triple test (JENA)

Furthermore, we measured the time spent by RDFCore to retrieve a description from the repository and make it ready for manipulation by user (Table 5 and Figure 10 and Figure 11) and in querying a model for every triple it contains (Table 6 and Figure 12 and Figure 13).

Resource	PDOM Persistence		JENA Persistence
	Elapsed time	Theoretical elapsed time	Elapsed time
Output10000	13570	13000	484
Output20000	23804	26000	5
Output30000	34420	39000	15
Output40000	43573	52000	63
Output50000	59285	65000	31
Output60000	-	-	5
Output70000	-	-	7
Output80000	-	-	15
Output90000	-	-	16
Output100000	-	-	16

Table 5 Retrieve description test

As for Figure 6 and Figure 7, in Figure 8 and Figure 9 the X axis reports the number of triples in the files used for the test.

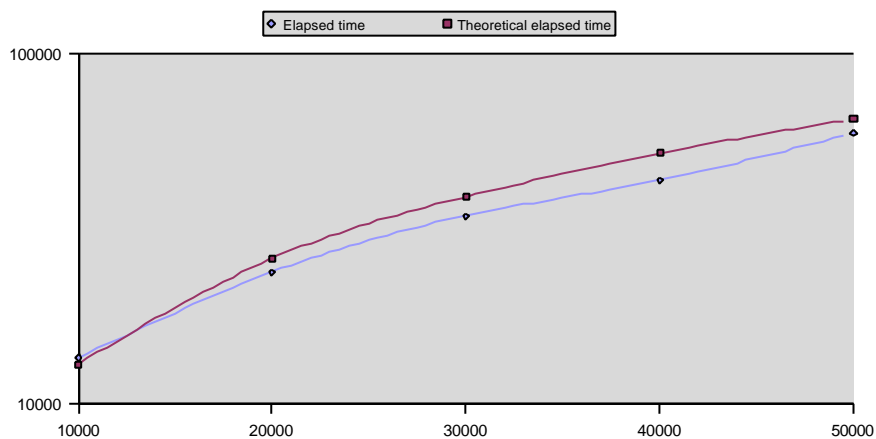


Figure 10 Retrieve description test (PDOM)

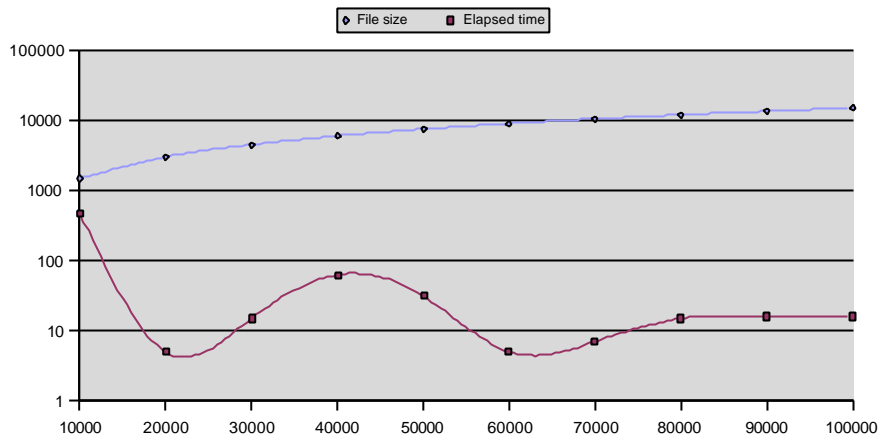


Figure 11 Retrieve Description(JENA)

		PDOM Persistence	JENA Persistence
Resource	Triple number	Elapsed time	Elapsed time
Output10000_rdf	10000	10505	453
Output20000_rdf	20000	15502	31
Output30000_rdf	30000	24075	16
Output40000_rdf	40000	32497	15
Output50000_rdf	50000	49361	16

Table 6 Querying persistence

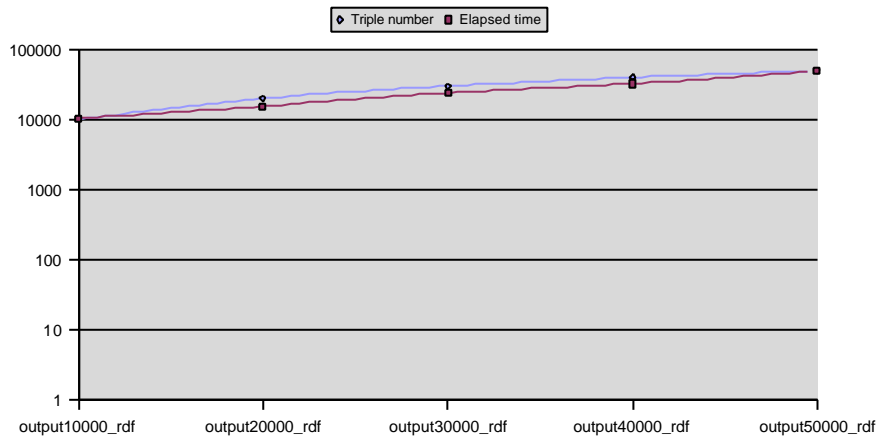


Figure 12 Querying persistence (PDOM)

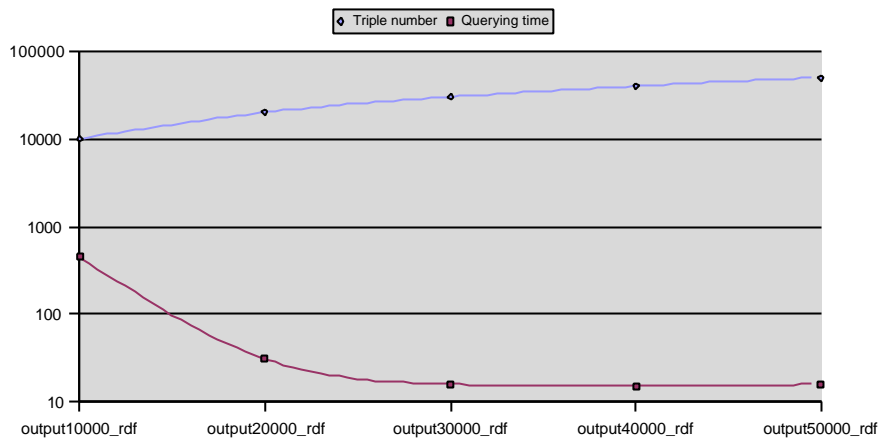


Figure 13 Query test (JENA)

Queries

The query test involves the use of the Enhanced Query Engine component of our architecture; specifically, the query used to stress the system (taking into account the size of the dataset and the size of results) was a very simple one: we asked the system to return every statement, describing a matching statement as a statement with a variable value for subject, predicate and object. This is done, in our system, creating a *Pattern* (a list of conditions on statements) and translating it into a query expressed in one of the query languages that are supported by the Enhanced Query Engine. In our test, we used RDQL as a query language; the translated query is

```
SELECT ?s, ?p, ?o WHERE (?s, ?p, ?o)
```

that returns every statement in the given model.

Result analysis

The obtained results show that the whole system does scale in a linear way with both persistence layers. It is noteworthy that JENA persistence absolute times, when adding a new model, are higher than those of the PDOM implementation. This depends on a JENA weakness due to the complexity of the internal database structure. The next version of JENA (JENA 2.0) promises substantial performance improvements, and this should tackle the resulting weakness of our system. On the

other hand, when doing retrieving and querying tests, where PDOM is still linear, JENA is very close to constant complexity, independently from the size of managed data. This result was expected because of the different approaches used by the two distinct layers: PDOM loads its data into in-memory representations, while Jena relies on its RDBMS persistence, obviously faster in these operations.

Conclusions

In this paper, we briefly described motivations and requirements for the brand new vision emerging on the Web: the Semantic Web. We pointed out, among others, the need of exploiting suitable technology for dealing with metadata, such as RDF. This technology has many benefits and, as we stated in the first sections of this paper, has to be integrated in frameworks that offer both scalability and standard support. Then, we presented our solution to tackle RDF related issues and we mentioned one specific application of RDFCore in a current ongoing EU research project (COLLATE). Finally, we presented an empirical evaluation from which we noticed that our designed architecture resulted in a scalable system (as shown by early tests on the prototype presented in this paper). Forthcoming research will have three main directions:

- Integration with RDF Schema Technology
- Moving to a standard RDF Query Language (when issued by responsible committee)
- Embedding Semantic Web upper level languages, such as DAML+OIL[9], in order to deal with ontologies and reasoning.

References

- [1] T. Berners-Lee, J. Henders and O. Lassilla, The Semantic Web Scientific American, May 2001 <http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2>
- [2] D. Beckett N-Triples EBNF Grammar definition <http://mail.lirt.bris.ac.uk/~cmdjb/2001/06/ntriples/>

- [3] B. McBride, Jena: A Semantic Web Toolkit, IEEE Internet Computing, Vol. 6, N. 6, 55-59, Nov/Dec 2002.
- [4] S. Melnik: "RDF API Draft", working document, Stanford University, 1999
- [5] COLLATE – COLLATE - Collaboratory for Annotation, Indexing and Retrieval of Digitized Historical Archive Material <http://www.collate.de/>
- [6] S. Ferilli, Management of Cultural Heritage Material: The COLLATE project. In: L. Bordoni, G. Semeraro (Eds.), Proceedings of the Workshop on Artificial Intelligence for Cultural Heritage and Digital Libraries, 7th Congress of the Italian Association for Artificial Intelligence (AI*IA '01), Bari, 25 September 2001, pp. 29-33.
- [7] H. Brocks, U. Thiel, A. Stein & A. Dirsch-Weigand, Customizable Retrieval Functions Based on User Tasks in the Cultural Heritage Domain. In: Constantopoulos, P. & Sølvsberg, I.T. (Eds.). Research and Advanced Technology for Digital Libraries. Proceedings of the 5th European Conference, ECDL 2001. Berlin: Springer, 2001, pp. 37-48.
- [8] Jena RDF Query Language <http://www.hpl.hp.com/semweb/rdql-grammar.html>
- [9] Horrocks, DAML+OIL: a Reason-able Web Ontology Language, in Jensen, C. S.; Jeffery, K. G.; Pokorny, J.; Saltenis, S.; Bertino, E.; Böhm, K.; Jarke, M. (Eds.), (2002) Advances in Database Technology - EDBT 2002, Lecture Notes in Computer Science 2287, 2-13, Springer:Berlin, 2002.
- [10] E.Gamma, R.Helm, R.Johnson, J.Vlissides, Design Patterns Addison-Wesley Pub Co; 1st edition (1995) ISBN 0201633612, pp. 315-324

Sharing Ontology by Web Services:

Implementation of a Semantic Network Service (SNS) in the context of the German Environmental Information Network (gein®)

Thomas Bandholtz

Consultant, Karl-F.-Schinkelstr. 2, 53127 Bonn, Germany
(formerly: Solutions Manager Knowledge Technologies, SchlumbergerSema)
thomas@bandholtz.info

Abstract. A thesaurus, a gazetteer and a chronology have been integrated in a consolidated ontology on the basis of the Topic Map pattern. The result has been made accessible to a working information community of 89 environmental authorities in Germany by Web Services technology. A semantically shared ontology can be shared physically in the Web.

1 Introduction

Way back in 1998, the Federal Environmental Agency in Germany launched the German Environmental Information Network [1] (*gein*®, www.gein.de), an R&D project which resulted in the implementation of a first version of an *Internet Information Broker* in 2000. In most aspects, this was what today is called an *agent* in the Semantic Web. *gein*® was a loose coupling of – initially - 50 information providers with about 50,000 Web pages and nine Web-interfaced databases, integrated by the agent (broker) with the help of a - hopefully - shared ontology, common Internet technology, and XML. Thus *gein*® is part of the "database and information system research as they relate to the Semantic Web and more broadly, to gain insight into the Semantic Web technology as it relates to databases and information systems" (<http://swdb.semanticweb.org>), as it is focused by the current workshop.

gein® successfully applied a common content classification system as a first step to any further content-related integration (or even "harmonization") of the different Internet information sources in its domain. The semantics had been formalized by a Thesaurus, a Gazetteer, and a Chronology. Bases on these, *gein*® was practicing automatic indexing of unstructured documents as well as a distributed query using XML metadata in HTTP requests. With this rather "avantgardistic" approach in 2000, *gein*® proved as the public information portal ("The Portal to German Environmental Information") of the German environmental authorities on the federal and states level anyway.

Following this encouraging experience, a follow-up project named “Semantic Network Service (SNS)” [2] has been launched in 2001 to overcome some restrictions of the initial version of ontology management and automatic indexing by improvements such as:

- Semantic integration of thesaurus, gazetteer, and chronology;
- Resolving of homonym ambiguities by context analysis
- Elaborated criteria for keyword ranking according to their significance in one document.
- Sharing ontology by Web Services
- Accessing semantic methods by Web Services

In this paper, I will concentrate on issues of *Semantics* and *Application*, as these have proved to be the more crucial aspects. The *Infrastructure* (*gein*[®] and SNS are built on J2EE, with open source as far as possible) sometimes has raised problems in reliability, interoperability, or performance, but these never have been critical for the project. In the following, I will discuss:

- Topic Maps, in their ability to integrate the *gein*[®] legacy and expose it to the Semantic Web,
- Web Services as an interfacing method that allows to share an ontology not only semantically, but as well physically.

2 Semantic Integration of a Thesaurus, a Gazetteer, and a Chronology in a Topic Map

The SNS project has been started in 2001, and there has been an early decision to use Topic Maps to model the ontology. While there is a – sometimes controversial - discussion about Topic Maps and the Semantic Web [3,45], I recommend considering Topic Maps as a pattern to be applied to Web Ontology. This may include using the Web Ontology Language (OWL) [6] to serialize Topic Maps.

There had been an early RDF discussion [7] in the design phase of *gein*[®] in 1999 which resulted in the decision not to use RDF as the productive XML format in the network. We implemented a community metadata profile in XML instead, with the option to be converted into RDF later.

In early 2001, we experienced a kind of *déjà vu* discussing the XML Topic Maps (XTM) [8] interchange format. Again, there was a format which was designed on an extremely abstract level, while we were looking for something which was optimized for fast and simple processing. That is why we developed a different XML structure for Topic Maps [9] first, defined in an XML Schema. After XTM became an Annex to ISO13250 as a recommended *interchange* format, we also implemented an XTM interface. From today’s perspective one would consider to implement an OWL interface as well, but this had been out of scope in 2002.

Anyway - none of these formats can embarrass the architecture of SNS, they just add another interchange format. The physical storage structure is encapsulated, following the requirements of a smooth performance. What had attracted us to apply the Topic Map model was not an interchange format, but the semantic pattern of Topic Maps itself, as described in the core ISO 13250 document [10].

Having worked with a thesaurus, a gazetteer, and a chronology, each of them in an individual (XML-) structure, we understood the need for an integrated model. Topic Maps promised a generic pattern to integrate the given diversity without loss.

2.1 Building on Linguistic Inheritance

The *gein*[®] vocabulary has been developed since 1999 integrating and extending the major semantic sources of the environmental domain in Germany.

The starting point was in the initial requirement to implement a thesaurus-based search with dimensions of *subject*, *location*, and *time*. Following this, *gein*[®] combined three semantic structures:

1. a thesaurus of currently 39,143 environmental terms (UmThes[®]),
2. a gazetteer including the intersections between 48,213 geographical objects of all kinds,
3. a chronology – the synopsis of historical and contemporary events that affected the environment.

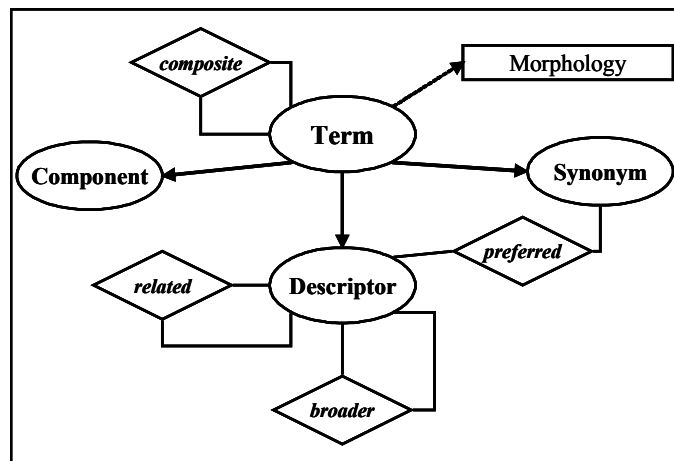


Fig. 1. Thesaurus Model of UmThes[®]

UmThes[®] [11] is a full-blown thesaurus supporting all the relations required by ISO 2788/5964 (Broader/Narrower; Synonym; Related; Component), and it contains most of the word morphology, as shown in Fig.1. It is also used by several German-speaking authorities such as the German and Austrian Environmental Data Catalogue, and it is the German source of the GEneral Multilingual Environmental Thesaurus (GEMET) [12,13].

The gein® Gazetteer is based on the GN250 (by Federal Agency for Cartography and Geodesy), but it adds several layers relevant for the environment, and it contains all the spatial intersections as explicit relations in the data, ready-to-use in a rapid query.

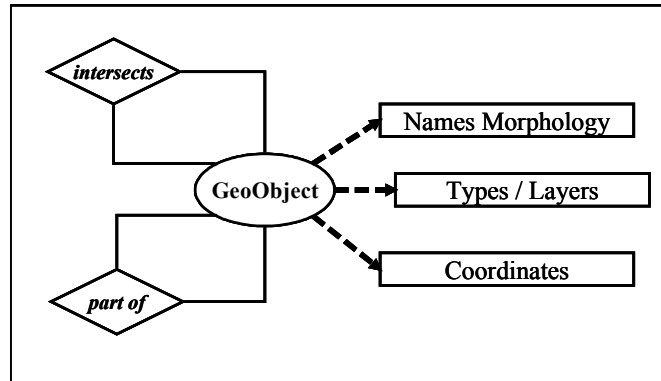


Fig. 2. Gazetteer Model of the "Geo-Thesaurus"

Today there is no established standard about gazetteers as it is for thesauri. There was an early approach of the Alexandria Digital Library in 1999 [14], and now we have the Open GIS Consortium's proposal of a "Gazetteer Service Specification", and the ISO Draft 19112 "Geographic information - Spatial referencing by geographic identifiers" [15]. Fig. 2 shows a generic model which is more or less implemented (or extended) by most of the existing gazetteers.

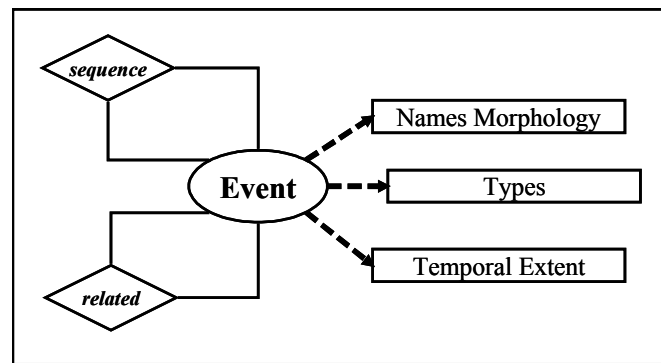


Fig. 3. Chronology Model of gein®

After having harvested a rich ontological legacy for the dimensions *subject* and *location*, we were inspired to find something comparable for the *temporal* aspect. We discovered that there are several symbolic names for events that do not contain any temporal notation, but an implicit reference to a date, such as "before (or after) Christ". While most people in the Christian culture can associate this with year "0", this cannot be postulated globally. Each domain knows its specific major events "by name", and most people cannot tell the exact date that they are talking about when the

use phrases like "since the Chernobyl disaster" (1986-04-26). This raised the idea to set up a mapping of symbolic names for events to their dates. The *gein*® Chronology has been started from scratch. Fig. 3 shows the structure.

2.2 Topic Maps

Topic Maps have originated in the neighborhood of SGML, more closely: in the ISO/IEC JTC 1/ SubCommittee (SC) 34 "Information Technology -- Document Description and Processing Languages" [16] which had worked with SGML, DSSSL, HyTime before. Unsurprisingly, the first interchange format has been written in HyTime, two years before an additional XML format (XTM) has been released by TopicMap.org.

But the standardization has not been based on interchange formats ("transfer serializations", which has been stressed by Jim Mason, Chairman of ISO/IEC JTC1/SC34:

"We need to keep clear that the transfer serializations are not the definition of Topic Maps: The standard is the definition. SC34 intends that the supplementary standards will clarify the meaning of Topic Maps without changing their essential nature. (We also recognize that other transfer serializations are possible, outside the standard.)" [17]

Topic Maps have often been described as the "GPS of the Information Space". They can be represented by graphs ("nodes and arcs"), but they are restricted to a more specific pattern of Topics, Associations, and Occurrences. *Topics* have *Occurrences* (in information objects), and there are certain *Associations* between these Topics.

This exactly corresponded to the view of the *gein*® information broker: a Topic may be a thesaurus descriptor or synonym, a geographic object in a gazetteer, an event, (or a person, an organization), whatsoever. Distinct kinds of Topics are defined as Topic Types in a Topic Map instance.

Associations may interconnect Topics in some kind of semantic relation. Distinct kinds of Associations, bound to certain Topic Types as their members, are defined as Association Templates in a Topic Map instance (though this is not sufficiently standardized yet).

An Occurrence may be seen as any kind of existing information about a Topic, but, as Occurrences are "groupings of addressable information objects around topics" [10], this should not be misunderstood to be the general *index* of a "corpora" like *gein*®. In SNS, the document index is separated from the Topic Map. Topics are used as classification properties in document metadata, which rather means: "groupings of topics around addressable information objects" [9].

The current work of SC34 [16] is dedicated to the creation of two related standards:

- ISO 18048: Topic Maps Query Language (TMQL)
- ISO 19756: Topic Maps Constraint Language (TMCL)

It is planned to create a *Standard Application Model* (SAM), a "formal data model for topic maps", flanked by a *Reference Model*, and a *Canonicalization*.

Not only to my opinion, these activities closely relate to the Semantic Web. In particular, couldn't the Web Ontology Language (OWL) [6], which had just advanced to a W3C Candidate Recommendation, function as a "Topic Maps Constraint Language"? I think, definitely yes, although OWL may not satisfy *every* TMCL requirement [18] currently in discussion. This has been explored by Lars Marius Garshol, SC34 member and editor, with the result that "semantic annotations in OWL can be translated directly into a topic map representation of the same information" [5]. While he states anyway that "merging the two technologies does not appear desirable or possible" (ibid.), I see relevant benefits in applying the Topic Map pattern to the modeling of Web ontologies, and in using OWL to serialize Topic Maps and their constraints.

Besides SC34, there is a vivid Topic Map community at OASIS with three technical committees [19] working on "Published Subjects". This work wants to extend the concept of *subjects* as given in the original ISO13250:

"In the most generic sense, a subject is anything whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever." [10]

In this concept, each Topic "reifies" a subject by referencing a "subject indicator".

"Any information resource can be considered a subject indicator simply by being referred to as such by an application, *whether or not that resource was intended by its publisher to be a subject indicator, and whether or not the publisher is aware of (or even cares about) its use as a subject indicator.*" [20]

The OASIS TCs are proposing the use of more explicit *published* subjects, *published* subject indicators (PSIs) and *published* subject identifiers (PSIDs). To me this sounds reasonable (and I am personally contributing), but this idea is not necessarily dedicated to solely Topic Maps.

While SC34 still behaves quite reserved about OWL, there is a first draft of expressing Topic Maps in OWL by Bernard Vatant, chair of the TC Published Subjects, providing

"... a reasonable platform for interoperability at a pragmatic level, covering quite a range of moderately complex use cases and applications, without need of any extension of current specifications beyond declaration of a minimal OWL vocabulary" [21].

2.3 Modeling the *gein*® Ontology in a Topic Map

SNS has defined its own Topic types and Association templates to model the three components of the *gein*® ontology. The *Thesaurus* type and its sub-types reproduce the classical thesaurus structure as defined in ISO 2788/5964. The *Location* type is the abstract parent of all the spatial types such as cities, catchment areas, or national parks. Likewise, the *Event* type is parent of conferences, disasters, and so on.

The given relations (such as broader/narrower terms, or intersection of locations) can be easily typed as Associations. So far, the three different structures can be formally integrated into a single Topic Map without any significant semantic loss or modification.

Beyond this, the three components have been interlinked by two new association types labeled *where*, and *what*. Both of them are using *Event* as the integration point. The *Where*-association links between *Event* and *Location*, pointing out where an event has happened. The *What*-association links between *Event* and *Descriptor* to describe which subjects have been affected by the event.

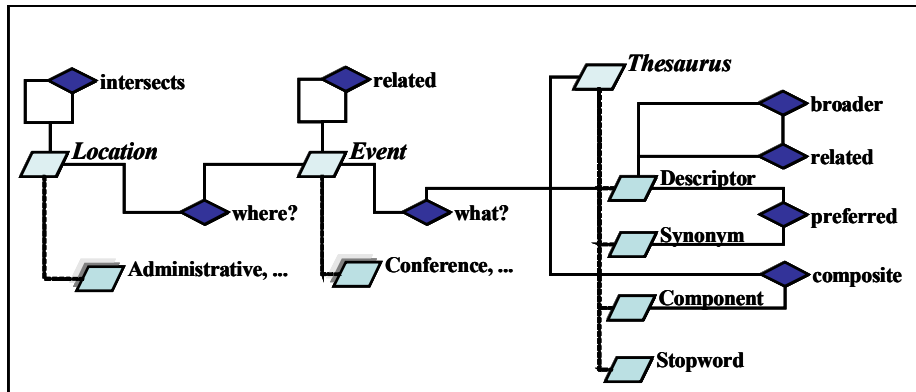


Fig. 4. The SNS Topic Map Typology

This modeling remains implicit, as the Topic Map community still owns no modeling or "constraint" language. There is kind of a "good practice" of describing the types in form of Topics themselves. But there the semantic expressivity of this style is only rudimentary, and there is no well-defined validation as it exists with XML Schema. In 2001, we experimented with using XML Schema to describe our Topic Map model and have the XML serialization validated against it, but this resulted in a rather proprietary solution which finally cannot be recommended. These issues have been discussed more closely in [9].

What I experienced as the most restricting issue is the missing support of extending Topic characteristics in an object-oriented manner. E.g., we need a temporal extent attribute for the Event types, and a bounding box attribute for the Location types. XML Topic Maps allow to (miss?-) use Occurrences to add properties, but you cannot use data types and explicit modeling to do so. This has been solved by OWL.

3 Sharing Ontology by Web Services

The *gein*® Broker has been hosting all the domain ontology since 1999. It has been used for the classification of currently 200,000 static Web pages published by 89 information providers, and in the distributed query to include nine cooperating databases in a distributed query.

There have been several requests by the information providers to be enabled to apply the same ontology and auto-classification methods for their own purposes. Thinking about the effort to prepare a compact module to be distributed for implementation in 89 possibly different technical environments, we preferred to consider a centralized service that can be accessed online by any of them.

gein® looks back to very positive experiences with distributed queries using XML embedded in HTTP requests. We had implemented this communication in the distributed query in 1999, even before the Simple Object Access Protocol (SOAP) had been submitted to the W3C (2000), which initiated the XML Protocol Working Group, and later expanded to the Web Service Activity.

In the recent months, Web Services have been discussed in the context of the Semantic Web quite frequently. In most cases the discussion is about using Web Services to process the Semantic Web, as by Tim Berners-Lee ("A story of program and data as old as computing" [22,23]), or using the Semantic Web for an approved Web Service description, as in Semantic Web Enabled Web Services (SWWS) [24]. Also the W3C Web Service Architecture [25] and the W3C Web Service Choreography [26,27] are recognizing the importance of explicit semantics and ontology to clarify the semantics of services. Similar for UDDI [28], or ebXML [29]. There is an elaborated approach of an "ontology of services" by DAML-S [30].

What we had in mind, was *sharing ontology by Web Services physically*.

3.1 Semantics of SNS Web Services

Based on the application experience in *gein*®, we designed three services [31]:

- Single Topic access by a unique ID (*getPSI*)
- Search for Topics by a single character string (*findTopics*)
- Auto-classification of a natural language document (*autoClassify*)

For "Single Topic access by a unique ID", the notation *getPSI* was taken from the Published Subject paradigm already introduced above and is short for: "get Published Subject Indicator (PSI)". We wanted to support Published Subject Identifiers (PSID) for each Topic.

Like in most Web Service applications, we bind this service to the SOAP protocol. However, SOAP does not satisfy the requirement that a PSID must have the form of a single URL, while SOAP needs a more complex protocol (HTTP Post).

Single URLs can have the form of a HTTP Get request, and indeed Web Services can be bound to the HTTP Get protocol. Doing so (additionally to the SOAP binding), a URL like:

http://www.semantic-network.de/.../getPSI?id=uba_thes_24027

will result in a representation of the referenced Topic, in this case the "Technical Instructions on Air Quality Control".

The idea of this service simply is to provide the Topic's characteristics (names, description, etc.) once a client (*agent*) has taken the ID from a reference. A typical use case may be finding this reference in some metadata, and trying to resolve it.

There has been lots of discussion in the committee about the kind of representation of a PSI. Should it be readable for humans or machines? In the Semantic Web, there *must* be a machine readable presentation, so that it may be processed by an agent. Likewise, Web Services are not directly invoked by humans, and an XML format is

expected in the response. So this PSI response is definitely machine-readable – (leaving out the argument that XML may be human-readable as well).

A human readable version is also provided by the URL:

http://www.semantic-network.de/displayTopic.html?lang=en&tid=uba_thes_24027,

but, while this may be called a kind of display service in the Web, it is not a Web Service, as it responds with only semi-structured, display-oriented HTML code.

"*Search for Topics by a single character string*" (*findTopics*) is provided as a classical free text query against the textual properties of Topics. There are several parameters controlling the search tolerance, such as restricting the search to names only or including textual parts of occurrences as well. The basic idea of this service is that the client is looking for Topics that possibly match a given keyword (character string). This is used by *gein*® to assist a human user who wants to proceed from a colloquial term to a Topic. In most cases, more than one Topic is returned, and the list may become quite long when the parameters are set to gain the most search tolerance.

"*Auto-classification of a natural language document*" (*autoClassify*) invokes a linguistic analysis of the passed text. It is the same analysis that *gein*® is using to generate the document index of the corpora automatically, but it may be applied in different cases as well, e.g. using a paragraph of a known document as an initial search condition. In this case, *autoClassify* returns a list of Topics which are significant for the given text paragraph and should be used as search terms.

3.2 Responses are Topic Map Fragments

In the design phase of the service responses we came across the problem that a single Topic with its full characteristics cannot be isolated from the Topic Map it appears in. The reason is Associations. ISO 13250 clearly sees Associations as part of the characteristics of a Topic, but each Association is referring to at least a second Topic. Surely an Association cannot be understood without an understanding of the associated Topic – which has more Associations ...

Practice has to find a solution. We have chosen to omit Associations in the results of *findTopics* and *autoClassify* which return lists of Topics, and to leave it to the requester of *getPSI* if he wants Associations to be included in the representation of a single Topic – together with the associated Topics, even recursively. *getPSI* has a parameter named *distance* to control the appearance of associated Topics.

But still, a fragment remains a fragment. Each thinkable subset of a Topic Map is losing semantics by being isolated from the original context. That is why we decided to let the fragment be explicit, which means adding a notation that expresses the origin, method of filtering (i.e. the request and its parameters), and date of filtering.

Given the not too mature state of current implementations of WSDL processors, this structure had its odds and ends to be settled before everything worked on today's major platforms of WSDL processing (Apache Axis and Microsoft dotNet).

4 State of Realization

The SNS R&D project has been finalized end of 2002, with some additional minor enhancements in 2003. The 2003 version of *gein*® replaces the previous semantic methods completely by interfacing SNS Web Services, which will enter the production phase in September.

But SNS has not been intended to be a *gein*®-only service. Its semantic model and functional services are provided for the integration in any kind of information system dealing with environmental issues in Germany, and, as SNS is bi-lingual, internationally.

In the near future there are several integration options, targeted to different users in different application areas, such as

1. UDK (German Catalog of Environmental Data Sources): An administrative agreement [33] of the Federal and *Länder* authorities in Germany has become effective, in which SNS is intended as the common basis of both systems in the next year.
2. *gein*® Information Providers: the (currently 89) contributing organizations [34] are invited to integrate SNS by Web Services for any kind of information activities. Some of them intend to implement a local version of SNS themselves. Finally, there may be a network of cascading Topic Maps depending on the spatial or thematic focus of an application.
3. GeoMIS.Bund: the "Metainformation-System for geodata of the Federation" of (IMAGI) [35], part of the German "national Geo data infrastructure is incorporating SNS to support thesaurus-based search and geographic names.
4. Europe: The eEIONET community discusses "environmental web services e.g. Reportnet, country networks, and metadata, as well as terminology/ontology issues" on a European level [36]. As the relation between GEMET and UmThes® is very close, and as SNS already is working bi-lingual (German/English), it is a candidate to be extended to a European Scope (gazetteer) and to the full multilingual context of currently 19 GEMET languages. This has been proposed in an Expression of Interest [37] within the 6th Framework Program of the European Commission.

5 Conclusions

SNS has successfully integrated the *gein*® thesaurus, gazetteer and chronology legacy into a service-oriented, integrated ontology system that serves a large information community.

Topic Maps have proved as a generic modeling pattern, but there are deficits in a formal modeling language.

Web Services have proved as a working communication protocol in order to access a domain ontology physically.

There are several issues to be solved, among which I regard the most crucial:

- Apply the Web Ontology Language with the Topic Maps pattern.
- Advance the interoperability of the Web Service Description features and XML Schema details to improve rapid implementations on different platforms.

References

1. Umweltinformationsnetz Deutschland. German Environmental Information Network - GEIN Research project UFOPLAN-Ref. No. 298 116 03/0. <http://www.gein.de/docs.html>.
2. "Implementation of a Semantic Network Service (SNS) in the context of the German Environmental Information Network (gein®)". Research project UFOPLAN-Ref. No. 20111612, promoted by BMU/UBA, Germany. <http://www.semantic-network.de>.
3. Martin S. Lacher and Stefan Decker: On the Integration of Topic Maps and RDF Data. International Semantic Web Working Symposium, Stanford 2001. <http://www.semanticweb.org/SWWS/program/full/paper53.pdf>
4. Marc de Graauw: Business Maps: Topic Maps Go B2B. August 21, 2002. <http://www.xml.com/lpt/a/2002/08/21/topicmapb2b.html>
5. Lars Marius Garshol: Living with topic maps and RDF. <http://www.ontopia.net/topicmaps/materials/tmrdf.html>
6. Web Ontology Language (OWL). Overview. W3C Candidate Recommendation 18 August 2003. <http://www.w3.org/TR/2003/CR-owl-features-20030818/>
7. Thomas Bandholtz: GEIN 2000 and beyond: Environmental Information in the "Semantic Web". 1. Workshop "Environmental Markup Language (EML)". Berlin, Humboldt University, 1999. <http://www.bandholtz.info/publications/1999/gein-eml99-en.pdf>
8. XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification 1.16 2001/08/06. <http://www.topicmaps.org/xtm/index.html>
9. Thomas Bandholtz: A Taxi in Knowledge Land. XMLEurope 2002, Barcelona. <http://62.231.133.220/idea-eks-nav/papers/03-05-03/03-05-03.html>
10. ISO/IEC 13250 Topic Maps. Second Edition. 19 May 2002 http://www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2nd-ed-v2.pdf
11. A short description (in German) of UmThes can be found at <http://www.umweltbundesamt.de/uba-datenbanken/thes1.htm> See also: the Thesaurus Editorial Board, http://www.cedar.at/wgr_home/
12. The GEneral Multilingual Environmental Thesaurus (GEMET) was developed by the European Environment Agency (EEA) and the ETC/CDS. http://www.mu.niedersachsen.de/cds/etc-cds_neu/library/Gemet.pdf HTML version at: http://www.mu.niedersachsen.de/cds/etc-cds_neu/library/select.html
13. UNEP.Net and the GEMET thesaurus. <http://www.unep.net/help/about-gemet.cfm>
14. Alexandria Digital Library: ADL Gazetteer Content Standard (version of 4/21/99). http://adl.billzworld.com/projects/gazetteer/content_standard/
15. Jens Fitzke: Standard-based Gazetteer Services. Presentation at the NKOS Workshop, JCDL 2002, on Digital gazetteers: integration into distributed digital library services, July 18, 2002. jens.fitzke@uni-bonn.de

16. M. Biezunski, S. Newcomb, and M. Bryan (ISO SC34): Guide to the topic map standards. 2002-06-23. <http://www.y12.doe.gov/sgml/sc34/document/0323.htm>
17. Jim Mason, Chairman of ISO/IEC JTC1/SC34 in his posting from 2001-12-14. <http://lists.oasis-open.org/archives/topicmaps-comment/200112/msg00012.html>
18. ISO/IEC JTC 1/SC34. Topic Map Constraint Language. <http://www.isotopicmaps.org/tmcl/>.
19. OASIS Topic Maps Published Subjects TC
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tm-pubsubj
 OASIS Topic Maps Published Subjects for Geography and Languages TC
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=geolang
 OASIS Topic Maps Vocabulary for XML Standards and Technologies TC
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xmlvoc
20. Published Subjects: Introduction and Basic Requirements. OASIS Published Subjects Technical Committee Recommendation, 2003-06-24
<http://www.oasis-open.org/committees/download.php/3050/pubsubj-pt1-1.02-cs.pdf>
21. Bernard Vatant: Cooking for the Semantic Web. OWL and Topic Map Pudding
<http://www.mondeca.com/owl/lang.rdf>
22. Tim Berners-Lee: (W3C Design Issues) Roadmap for Web Services. 2003/07/24.
<http://www.w3.org/DesignIssues/WebServices.html>
23. Tim Berners-Lee: Web Services - Semantic Web.
<http://www.w3.org/2003/Talks/0521-www-keynote-tbl/>
24. Christoph Bussler, Dieter Fensel, Alexander Maedche: A Conceptual Architecture for Semantic Web Enabled Web Services. ACM Special Interest Group on Management of Data: Volume 31, Number 4, Dec 2002. See also <http://swws.semanticweb.org/swws>
25. Web Services Architecture. W3C Working Draft 8 August 2003.
<http://www.w3.org/TR/2003/WD-ws-arch-20030808/>
26. Web Services Choreography Working Group Charter, W3C v 1.25 2003/01/14.
<http://www.w3.org/2003/01/wscwg-charter>
27. Web Services Choreography Requirements 1.0. W3C Working Draft 12 August 2003
<http://www.w3.org/TR/2003/WD-ws-chor-reqs-20030812>
28. UDDI tModels. Classification Schemes, Taxonomies, Identifier Systems, and Relationships, Version 2.04 11 December 2002.
http://uddi.org/taxonomies/UDDI_Taxonomy_tModels.htm
29. ebXML Case Study: Exploiting Web Service Semantics through ebXML Registries and Software Agents. February 19, 2003.
http://www.ebxml.org/case_studies/documents/metuebxmljmtcasestudy_060503.pdf
30. David Martin (The DAML Services Coalition): DAML-S: Semantic Markup for Web Services. White Paper 2003-05-05. <http://www.daml.org/services/daml-s/0.9/daml-s.html>
31. For closer information see the Semantic Network Services online documentation.
http://www.semantic-network.de/doc_intro.html?lang=en
32. W3C Web Services Description Working Group. <http://www.w3.org/2002/ws/desc/>
33. Administrative Agreement and Coordination UDK/GEIN®. <http://www.udk-gein.de/>
34. gein® Information Providers. http://www.gein.de/provi_en.html
35. Inter-ministerial commission for geo-information (IMAGI), Germany,
<http://www.imagi.de/>
36. EIONET: European Environment Information and Observation Network. eEIONET Work Conference from 26-28 September 2002 in Vienna. Released: 2002/07/03.
http://eea.eionet.eu.int/Best_Practice/eEIONET2002
37. EoI: European Environmental Topic Map Engine with Multilingual Auto-Classification (EETM). Expression of Interest to the 6th Framework Programme of the European Commission. June 2002. http://www.jiscmail.ac.uk/files/DC-ENVIRONMENT/EoI_Bandholtz.doc

ODE-SWS: A Semantic Web Service Development Environment

Óscar Corcho¹, Asunción Gómez-Pérez¹, and Mariano Fernández-López¹
Manuel Lama²

¹ Departamento de Inteligencia Artificial. Facultad de Informática.
Campus de Montegancedo, s/n. Universidad Politécnica de Madrid.
28660 Boadilla del Monte, Madrid, Spain.
{ocorcho,mfernandez,asun}@fi.upm.es

² Departamento de Electrónica y Computación. Facultad de Física.
Campus Sur, s/n. Universidad de Santiago de Compostela.
15782 Santiago de Compostela, A Coruna, Spain.
lama@dec.usc.es, davidal@usc.es

Abstract. Web Services (WS) are software modules that perform operations that are network-accessible through XML messaging. Web Services in the Semantic Web, that is, Semantic Web Services (SWS), should describe semantically their structure and capabilities to enable its automatic discovery, invocation and composition. In this work we present a development environment to design SWS in a language-independent manner. This environment is based on a framework that defines an ontology set to characterize how a SWS should be specified. The core ontology of this framework describes the SWS problem-solving behaviour and enables the SWS design at a conceptual level. Considering this framework, the SWS development environment is composed of (1) a graphical interface, in which the conceptual design of SWSs is performed, and (2) a tool set, which instantiates the framework ontologies according to the graphical model created by the user, verifies the completeness and consistency of the SWS through instance evaluation, and translates the SWS conceptual model description into SWS (and WS) languages, such as DAML-S, WSDL or UDDI. This tool set is integrated in the WebODE ontology engineering workbench in order to take advantage of its reasoning and ontology translation capabilities.

1 Introduction

Web Services (WSs) are software modules that describe a collection of operations that can be network-accessible through standardized XML messaging [1]. WSs are distributed all over the Internet, and in order to enable this accessibility and interactions between WSs, it becomes necessary an infrastructure offering mechanisms to support the WS discovery and direct invocation from other services or agents. Nowadays, there are a number of proposals (usually ecommerce-oriented) that claim to enable partial or totally this required infrastructure, such

as ebXML [2], E-Speak [3], or BPEL4WS [4]. However, the approach that has emerged as a de facto standard, due to its extended use and relative simplicity, is the Web Service Conceptual Architecture [1]. This framework is composed of a set of layers that, basically, enable: (1) *WS publication*, where the UDDI specification [5] is used to define the WS capabilities and characterize its service provider; (2) *WS description*, which use the WSDL language [6] to specify how the service can be invoked (input-output messages), and SOAP [7] as the communication protocol for accessing web services; and (3) *WS composition*, which specifies how a complex service can be created combining simple ones. The language used to describe this composition is WSFL [8].

In this context, the Semantic Web [9] has risen as a Web evolution where the information is semantically expressed in a markup language (such as DAML+OIL [10]) and, thus, both agents and services could access directly to it. This approach considers that the Web Services in the Semantic Web, so called Semantic Web Services (SWSs), should specify their capabilities and properties in a semantic markup language [11], [10]. This markup would enable other services to reason about the SWS, and, as a result, decide whether it matches their requirements. Taking this into account, two frameworks, SWSA [13] and WSFM [14], have been proposed to describe a semantic Web infrastructure for enabling the automatic SWS discovery, invocation and composition. Both frameworks use the DAML-S specification [15], which is a DAML+OIL ontology for SWS specification, and emphasize the SWS integration with de facto standard WS, in order to take advantage of its current infrastructure.

On the other hand, Problem-Solving Methods (PSMs) describe explicitly how a task can be performed [16]. PSMs are intended to be *reusable* components applicable to similar tasks but in different domains. A PSM description specifies the tasks in which the PSM is decomposed (methods-tasks tree); the input-output interactions between the tasks; the flow control that describes the task execution; the conditions in which a PSM can be applied to a domain or task; and, finally, the ontology used by the PSM (method ontology). The UPML specification [17] provides containers in which these PSM views can be described, and, also, it incorporates elements that enable the PSM reuse. UPML has been developed in the context of the IBROW project [18] with the aim of enabling the semi-automatic reuse of PSMs. This objective could be interpreted as a composition of PSMs.

In this work we provide a SWS development environment, called *ODE-SWS*, which would allow the user to design SWSs on the basis of PSM modelling, enabling its description and composition at a conceptual level. This environment also performs verification about the consistency and completeness of the design created by the user. Once the design is verified, the user will select the specific languages in which the SWS will be specified. Thus, the SWS development process supported by this environment does not depend on a specific SWS specification language. On the other hand, ODE-SWS is integrated in WebODE [19], an ontology development workbench that offers an infrastructure in which ontology services (such as merging, evaluating and reasoning with ontologies) can

be reused by other services or applications. In this way, ODE-SWS development has been facilitated with its integration in WebODE.

The structure of the paper is as follows. In section 2, a PSM-based framework that enables the SWSs (and WSs) development is presented. In section 3, the software architecture of the environment that supports this framework and how it has been integrated in WebODE is described. In section 4, the current capabilities of its graphical interface are explained. Finally, in section 5, the main contributions of the work are summarized and other proposals to develop SWS are discussed.

2 Framework for SWS Development

Relationships between SWSs and PSMs have been emphasized by several authors [20], [14]. When both SWSs and PSMs are applied, they execute an operation (or equivalently a method) to perform a task in a domain. As a result of this execution, either new domain information is obtained or an effect is provoked in the real world. Taking this similarity into account, it seems to be reasonable to use the PSM paradigm to define the SWS features related to their internal structure (SWS description and composition). Thus, we propose a framework in which the SWS development is *based on* PSM descriptions, which could be extended with knowledge about ecommerce features (to facilitate SWS discovery) and communication protocols (to provide network-accessibility).

On the other hand, the design of the framework has been guided by a set of requirements that establish the conditions to define an open and extensible framework to develop SWSs. These requirements are as follows:

1. *SWS conceptual modeling.* SWS development must be carried out at a conceptual level and, therefore, characterization and description of the SWS capabilities and internal structure (for composition and description) cannot depend on specific languages that could limit the expressiveness of the SWS model. Our aim is to allow the users to develop SWSs in a language-independent manner; the environment that supports the framework will be responsible to translate the SWS design into the required SWS languages.
2. *Integration of SWS with Web Service standards.* SWS specifications should be integrated with Web service de facto standards (both frameworks and languages) to be able to use the current infrastructure that supports these standards [13], [20]. This requirement is compatible with the need of enabling a SWS conceptual design, because this integration is carried out once the SWS conceptual model has been created.
3. *Modular design.* The framework must be composed of a set of independent, but related, modules, which contain knowledge about different views of the SWS development process. This criterion guarantees the extensibility of the framework, because we could introduce new modules without have to modify the others.

2.1 Layered-Based Framework

To cover these requirements we propose a framework with a layered design, whose layers are identified following a generality criterion, from the data types (lower layer) to the specific languages in which SWSs will be expressed (higher layer). Each layer is defined by an *ontology* that describes its elements on the basis of well-known standards. These ontologies (or layers) are the following (see figure 1):

- *Data Types (DT) Ontology*. It contains the data types associated to the concept attributes of the domain ontology. The data types included in the DT ontology are the same as the ones defined in the XML Schema Data Types specification [21].
- *Knowledge Representation (KR) Ontology*. It describes the representation primitives used to specify the domain ontology managed by SWSs in its operations. That is, the components of the domain ontology will be KR instances. KR ontology is needed because higher framework ontologies (PSM and SWS) could need to reason about the domain ontology. For example, preconditions of a method could impose that the input-output data should be attributes. Usually, the KR ontology is associated to the knowledge model of the tool used to develop the domain ontology.
- *PSM Description Ontology*. This ontology describes the elements that compose a PSM, which, as we have previously discussed, can be used to generate SWS descriptions. The PSM ontology is constructed following the UPML specification [17], that has been extended with (1) a *programming structure*

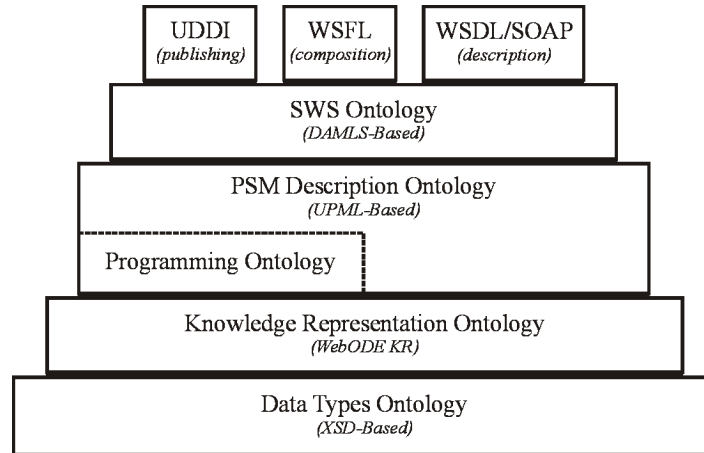


Fig. 1. Framework for SWS development. This framework is composed of a set of design layers, each one defined by an ontology that is based on well-known specifications of the components that it describes

ontology, which describes the primitives used to specify the PSM flow control (such as conditional and parallel loops, conditional statements, etc.); (2) *inferences*, which are new PSM elements defined as in the CommonKADS knowledge model [22], that is, as building blocks for reasoning processes; and (3) relations between PSM elements to explicitly declare whether an element may be executed *independently* of the others or not and whether they can be invoked by an external agent (or service). In figure 2 an excerpt of the PSM ontology is showed. On the other hand, the PSM ontology contains a number of axioms that constrain how PSM element instances are created. This guarantees the consistency of the PSM model. For example, there exists an axiom establishing that the input method must be covered by the inputs associated to the tasks that compose the method.

- *SWS Ontology*. This ontology is constructed on the basis of the PSM description ontology, which is extended with both knowledge related to ecommerce interactions, which enable the publication and advertisement of services, and communication protocols. These extensions are performed using the DAML-S specification as reference [15], because it describes containers to include these types of knowledge.
- *Standard language ontologies for Web Services*. They describe the elements associated to the de facto Web standard languages for service publication (UDDI), description (WSDL/SOAP), and composition (WSFL). These on-

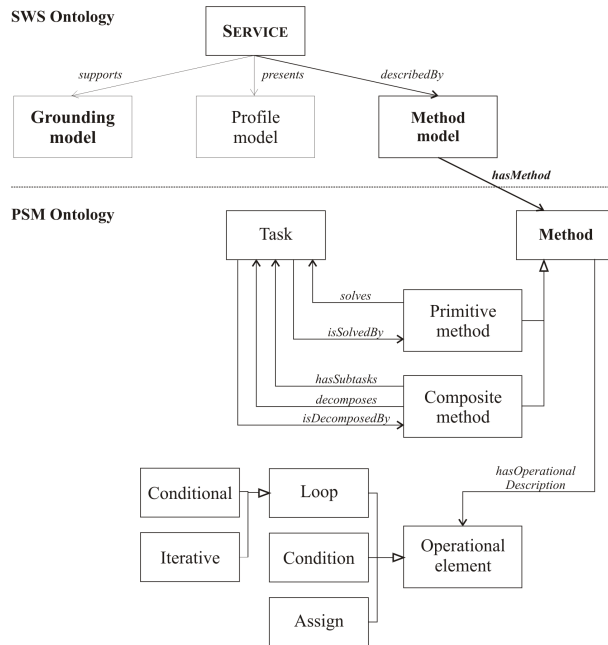


Fig. 2. Excerpt of the PSM ontology and how it is related with the SWS ontology

tologies complete the SWS specification, because they facilitate its integration in the current infrastructure of the Web.

This framework verifies the design requirements: conceptual modeling of SWSs is performed in the PSM layer, which is not constructed following a specific language, but modelled at knowledge level [23]; integration with Web service standards is explicitly enabled in the higher framework layer, which, if required, could be easily extended to include new standards; and, finally, modular design is achieved through the layered approach itself.

3 SWS Development Environment

To provide support for the framework, we have designed a SWS development environment, in which users can design the conceptual model of SWS through a graphical interface. Once finished, the model must be checked to guarantee its consistency and correctness. Then the SWS model can be converted into a DAML+OIL specification (such as DAML+OIL), which will be complemented with Web service standard languages. The software architecture of this environment, which is called *ODE-SWS*, has been designed following the framework requirements, that is, to develop an open and extensible environment, which, if required, could be easily modified to support new SWS (and WS) specification languages or frameworks.

3.1 Software Architecture

According to the proposed framework, the SWS development could be viewed as the process of instantiating an ontology set that contains the knowledge needed to generate the SWS specifications. ODE-SWS software architecture is based on this consideration and it is composed of: a *graphical interface*, which allows the users to develop SWSs at a conceptual level (section 4); and a set of services (or tools), called *ODE-SWS services*, which process the SWS graphical descriptions (previously created by the users) to generate the instances of the framework ontology at which each service is *connected*. That is, each framework layer is associated to a ODE-SWS service which operates with the knowledge contained into the ontology that describes that layer.

Figure 3 shows the general structure of a ODE-SWS service. Usually, a service is activated by the ODE-SWS graphical interface to (1) verify the consistency and completeness of the SWS conceptual model; or (2) translate this model from its graphical description into a specific language. In both cases, however, it is necessary to generate an instance set of the ontology connected to the service. In the first case, the SWS conceptual model is verified applying the ontology axioms to the instance set; the ODE-SWS service contains a module that will activate the reasoning with the ontology axioms. In the second case, it is also necessary to check the consistency and completeness of the SWS model to avoid errors in the specification of the SWS. Once this verification has been carried out, an

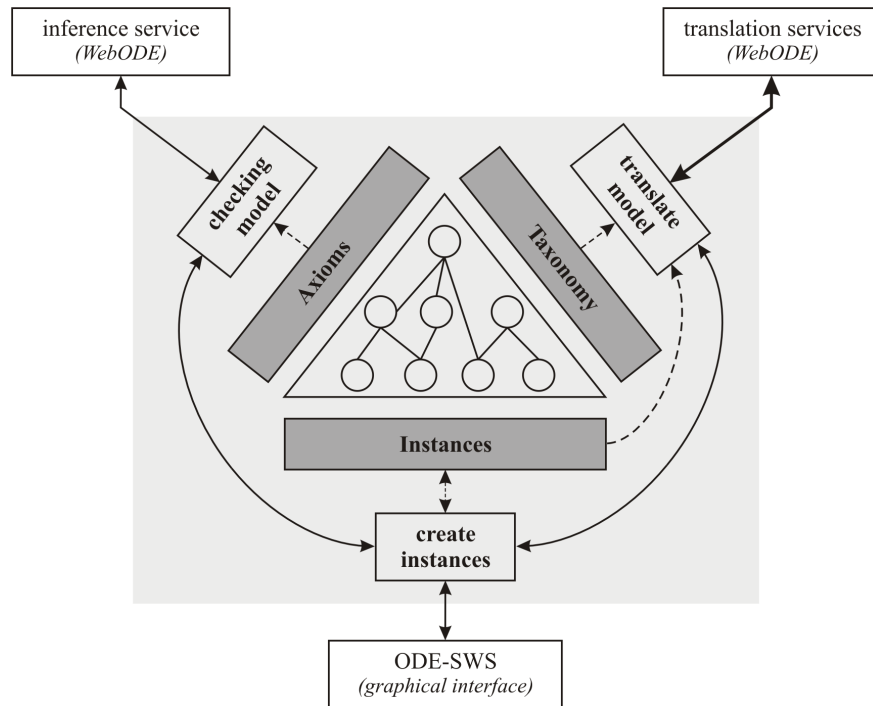


Fig. 3. General structure of a ODE-SWS service, where the ontology with which the service operates must be *one* of the ontologies identified in the SWS development framework

ODE-SWS service module will export the ontology to the language selected by the user.

On the other hand, ODE-SWS is completely integrated in WebODE [19], which is a workbench for ontology development that provides additional services for exporting ontologies to different languages (such as DAML+OIL, RDF, etc.), merging and evaluating ontologies, and reasoning with ontologies using their axioms. The WebODE software architecture is scalable and easily extensible, and it is divided in three layers (figure 4). In the first layer, the *ontology development services* are included. They verify the ontology consistency, enable the access to the ontologies stored in a relational database, reason with ontology axioms, and export/import the ontologies to/from different languages.

In the second layer the *middleware services* are located. They use the ontology development services in their operations and provide additional capabilities to WebODE, such as merging or evaluation. The ODE-SWS services are integrated in this layer. Thus, they directly use: (1) the *WebODE inference service* to evaluate the ontologies by means of their axioms; (2) the *WebODE ontology access service* to manage the framework ontologies (which are stored in We-

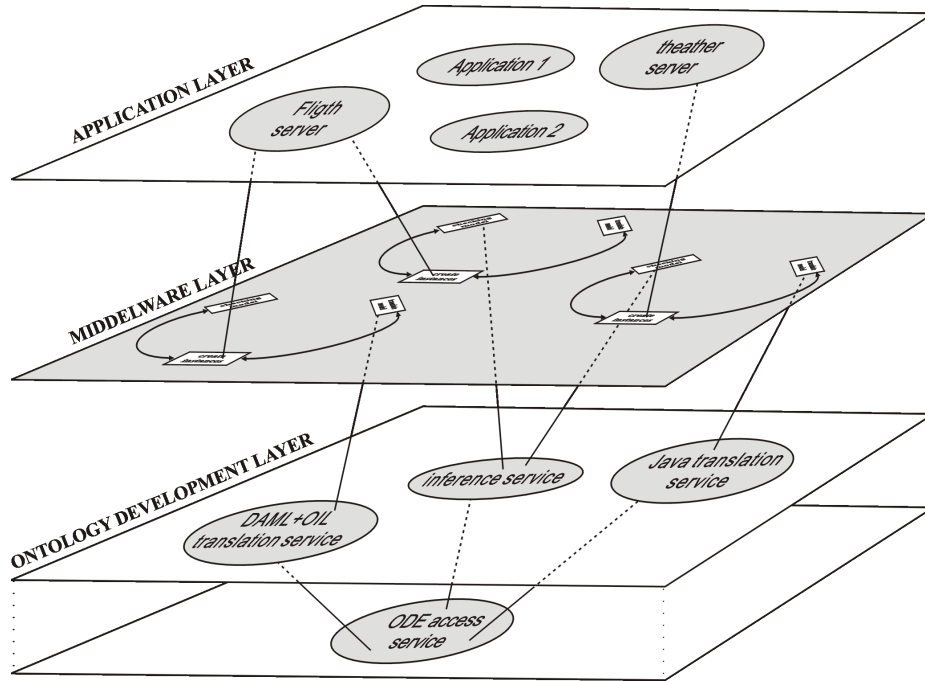


Fig. 4. Integration of ODE-SWS services in the WebODE architecture

bODE); and (3) the *export* services to translate the SWS model into a specific SWS language. In this layer the ODE-SWS graphical interface is also included and uses the ODE-SWS services and the WebODE ontology access service.

Finally, in the third layer the *applications* that mainly use the middleware services in their operations are constructed. For example, a *theatre server* application that offers SWSs to allow the users to book tickets for a particular film projected in the theatre, will probably use ODE-SWS because it provides capabilities needed in the application definition. Therefore, WebODE platform could be considered as an application development environment, in which new services can be easily integrated and reused by other applications by means of the *infrastructure* provided by the platform.

3.2 ODE-SWS services

ODE-SWS services are directly invoked from the ODE-SWS graphical interface when the users, once they create the SWS conceptual model in a graphical manner, require to export that model to well-known SWS languages or when the graphical interface itself needs to verify whether an operation carried out by the user has generated a SWS inconsistent model or not. Taking this into account, we identify the following ODE-SWS services (figure 5):

- *KR service*. This service gets as input the ontology used in SWS operation (usually the domain ontology) and establishes the instances associated to the KR and Data Types ontologies. The domain ontology can be available in WebODE or could be imported from an ontology language into the WebODE specification. In both cases, this service will invoke the ODE service to access the domain ontology components stored in a database.
- *PSM service*. It uses the graphical descriptions of the SWS model to generate an instance set that describes completely the PSM model (internal structure and flow control). Once the instance set is created, this service must invoke the WebODE inference service [24] to verify the consistency and completeness of the PSM model. In this verification, the axioms that constrain how the PSM elements can be combined with each other are used. For example, if we would define a general service that is decomposed in two sub-services,

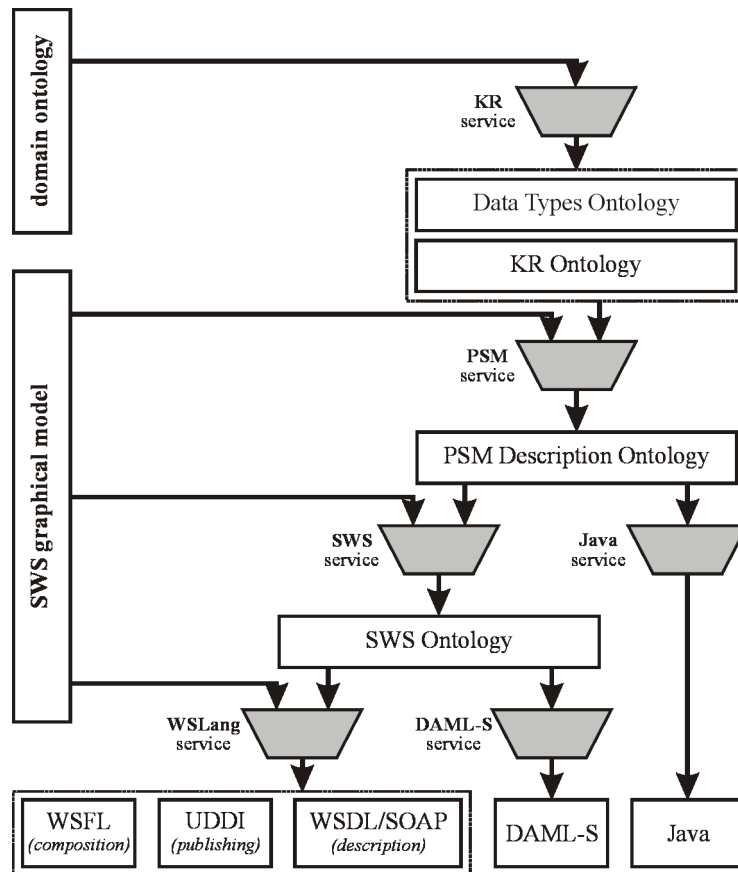


Fig. 5. Input-output relations between ODE-SWS services in order to generate the SWS model and its specification in a SWS language

it would be necessary to verify that the inputs of these sub-services would be of the same (or subsumed) type as the general service inputs. In order to perform this verification, the PSM service must operate with an explicit description of the representation primitives in which the domain ontology will be instanced.

- *SWS service*. Instances created by this service will enhance the knowledge included in the PSM model by adding the information related to ecommerce interactions. This information will be directly obtained from the ODE-SWS graphical interface.

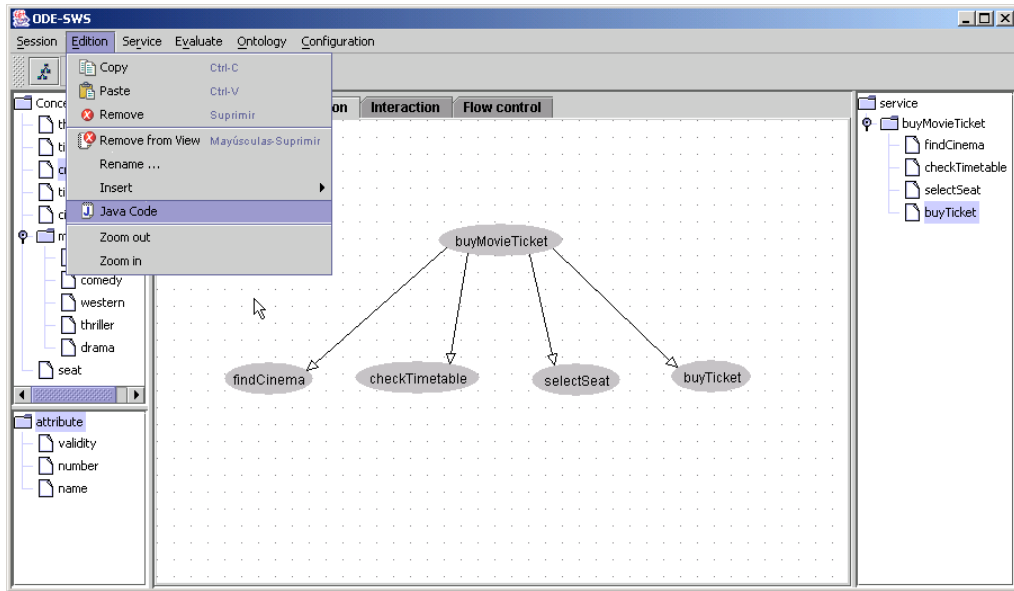
These three services constitute the *ODE-SWS core*, because they support the generation of the SWS conceptual model (from the SWS graphical descriptions) and their operation does not depend on the specific languages in which the SWS will be described. Therefore, these services will be modified only if their associated framework layers are also changed.

- *WSLang service*. It gets as inputs the SWS ontology instances and generates an instance set from which the SWS model is specified in UDDI, WSDL/SOAP and WSFL de facto standard languages.
- *DAML-S service*. It provides the DAML-S specification of the SWS having as inputs the instances of the SWS ontology. Nevertheless, this operation is not straightforward because in the DAML-S ontology a service is modeled as a *process*, whereas in our framework a service is considered to be a *specialization of a PSM* (or method). Once this operation is performed, this service must invoke the WebODE service, which exports an ontology into the DAML+OIL language.
- *Java service*. Using the PSM ontology instances as inputs, this service will generate the skeleton of the programming code (Java beans) needed to execute the SWS and to perform its operations. Once this code has been generated by the service, the user must fill in the methods responsible of carrying out the operations modelled in the PSM.

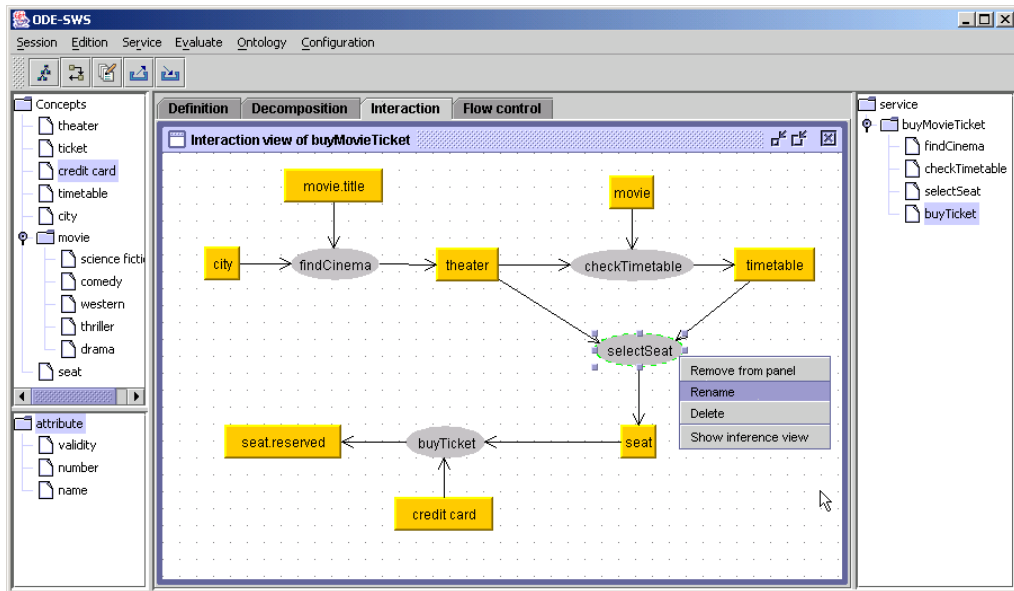
These three services represent ODE-SWS *additional services*, because they have been specifically included in the environment to support the translation from the SWS model into the languages in which the SWS will be expressed. This means that these services would be changed (or substituted) if it was required to use other languages or if the core services were also modified.

4 Graphical Interface

ODE-SWS graphical interface is based on the assumption that the design and development of a service should be performed from different, but complementary, points of view (such as in PSM modelling). These different views help the user to understand the internal structure of a service and the interactions between its components (sub-services); that is, these views facilitate the SWS description and composition. Taking this into account, the graphical interface contains the following views (see figure 6):



(a)



(b)

Fig. 6. ODE-SWS graphical interface

- *Definition view.* In this view the user defines a service by specifying its name (mandatory) and, optionally, by introducing the information needed to enable service discovery and advertisement, such as a description of the provider that offers the service, the types of business for which the service is oriented (industry classifications), etc.
- *Decomposition view.* This view allows the user to define (and also create) the services (sub-services) that would be executed when a (composite) service is activated. That is, a service hierarchy can be specified. This view, therefore, enables *service composition* by creating a hierarchy in which the sub-services of a composite service are activated if it verifies their execution conditions. Figure 6.(a) shows how the service *BuyMovieTicket* is decomposed in its sub-services. On the other hand, this view can be used to detect possible inconsistencies between different views. For example, in the flow control of a service cannot appear services that do not belong to its hierarchy.
- *Interaction view.* In this view the input-output interactions between the sub-services of a composite service are specified. This operation requires that the domain ontology would be previously loaded from WebODE database to the graphical interface. Figure 6 shows the main window of the ODE-SWS, where the specification of the interactions between the sub-services of *buyMovieTicket* composite service can be seen. All these services have been created in the decomposition view (or in the definition view), which will generate the service tree shown in the right side of figure 6.(b).
- *Flow control view.* In this view the user specifies the flow control of a service, where its sub-services are combined with programming structures to obtain a description of the service execution. This view, which is not implemented yet, will be used to model the service composition by means of several diagrams that describe the different compositions of services. On the other hand, this view and the decomposition view could be used to export to languages (as WSFL) that specify the service composition.

The graphical interface guarantees the consistency and completeness of the models that have been created in each one of its views. For example, if the user specifies that a service is composed of three sub-services (decomposition view), the graphical interface will invoke the PSM service to assure that the interaction view contains exactly those three services (as in the example shown in figure 6).

5 Conclusions

ODE-SWS enables the users to develop SWSs following a PSM-oriented design, which is based on a language-independent framework for SWS development. Furthermore, ODE-SWS will assure the consistency and completeness of the SWS designs. Once the SWS design correctness is verified, the user can select the languages in which the SWS will be described. Thus, in ODE-SWS the user does not need to know specific details about the languages used to specify the SWSs.

On the other hand, the ODE-SWS integration in WebODE has simplified its software architecture and implementation, because (1) it uses directly the WebODE services, which offer support for ODE-SWS operations; and (2) it uses the infrastructure itself that WebODE provides for including software modules as services, which could be easily accessed from the graphical interface. Thus, the integration in WebODE favors the ODE SWS modularity, which is a key requirement to adapt the environment to new standard languages or frameworks.

Finally, there exists some development environments which offer capabilities for SWS composition and consistency verification [26], [25]. Both environments are based on the DAML-S ontology and they use the reasoning capabilities associated to the DAML+OIL language to verify the SWS model consistency. These environments are language-dependent and the SWS conceptual modelling depends on the DAML+OIL mark-up, which, therefore, highly difficult its translation to others languages or frameworks. On the other hand, none of these two environments are supported by an infrastructure that could offer other useful capabilities such as evaluation or reasoning about ontologies.

References

1. H. Kreger: *Web Services Conceptual Architecture (WSCA 1.0)*. <http://www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf>, May 2001.
2. D. Webber and A. Dutton: Understanding ebXML, UDDI and XML/edi. http://www.xmlglobal.com/downloads/ebXML_understanding.pdf, October 2000.
3. S. Graupner, W. Kim, D. Lenkov, and A. Sahai: *E-Speak - An Enabling Infrastructure for Web-based E-Services*. Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, L'Aquila, Italy, July August 2000.
4. F. Curbera, Y. Golan, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana: *Business Process Execution Language for Web Services. Version 1*. <http://www.ibm.com/developerworks/library/ws-bpel>, July 2002.
5. T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y.L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, and C. von Riegen: *UDDI Version 3.0. Published Specification*. <http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>, July 2002.
6. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana: *Web Services Description Language (WSDL) 1.1*. <http://www.w3c.org/TR/2001/NOTE-wsdl-20010315>, March 2001.
7. D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S. Thatte, and D. Winer: *Simple Object Access Protocol (SOAP) 1.1*. <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>, May 2000.
8. F. Leymann: *Web Service Flow Language (WSFL 1.0)*. <http://www.ibm.com/software/solutions/webservices/pdf/WSDL.pdf>, May 2001.
9. T. Berners-Lee, J. Hendler, and O. Lassila: *The Semantic Web*. Scientific American, 284(5):34-43, 2001.
10. J. Hendler and D. McGuinness: *The DARPA Agent Markup Language*. IEEE Intelligent Systems, 15(6):72-73, 2000.

11. S.A. McIlraith, T.C. Son, and H. Zeng: *Semantic Web Services*. IEEE Intelligent Systems, 16(2):46-53, 2001.
12. J. Hendler: *Agents and the Semantic Web*. IEEE Intelligent Systems, 16(2):30-37, 2001.
13. T. Sollazzo, S. Handshuch, S. Staab, and M. Frank: *Semantic Web Service Architecture – Evolving Web Service Standards toward the Semantic Web*. Proceedings of the Fifteenth International FLAIRS Conference, Pensacola, Florida, May 2002.
14. D. Fensel and C. Bussler: *The Web Service Modeling Framework WSMF*. Proceedings of the NSF-EU Workshop on Database and Information Systems Research for Semantic Web and Enterprises, pages 15-20, Georgia, USA, April 2002.
15. A. Ankolenkar, M. Burstein, J.R. Hobbs, O. Lassila, D.L. Martin, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng: *DAML-S: Semantic Markup for Web Services*. Proceedings of the First Semantic Web Working Symposium, pages 411-430, July August 2001.
16. V.R. Benjamins and D. Fensel: *Special Issue on Problem-Solving Methods*. International Journal of Human-Computer Studies (IJHCS), 49(4):305-313, 1998.
17. D. Fensel, E. Motta, F. van Harmelen, V.R. Benjamins, M. Crubezy, S. Decker, M. Gaspari, R. Groenboom, W. Grosso, M. Musen, E. Plaza, G. Schreiber, R. Studer, and B. Wielinga: *The Unified Problem-Solving Method Development Language UPML*. Knowledge and Information Systems (KAIS): An International Journal, 2003. To appear.
18. V.R. Benjamins, B. Wielinga, J. Wielemaker, and D. Fensel: *Brokering Problem-Solving Knowledge at the Internet*. Proceedings of the European Knowledge Acquisition Workshop (EKAW-99), Lecture Notes in Artificial Intelligence, LNAI 1621, May 1999.
19. J.C. Arpirez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez: *WebODE – A Scalable Ontological Engineering Workbench*. Proceedings of the First International Conference on Knowledge Capture, Victoria, Canada, October 2001.
20. V.R. Benjamins: *Web Service Solve Problems, and Problem-Solving Methods Provide Services*. IEEE Intelligent Systems, 18(1):76-77, January/February 2003.
21. P.V. Biron and A. Malhotra: *XML Schema Part 2: Datatypes*. <http://www.w3c.org/TR/2001/REC-schema-2-20010502>, May 2001.
22. G. Schreiber, H. Akkermans, A. Anjevierden, R. de Hoog, H. Shadbolt, W. van de Welde, and B. Wielinga: *Knowledge engineering and management. The CommonKADS Methodology*. MIT Press, Cambridge, Massachusetts.
23. A. Newell: *The Knowledge Level*. Artificial Intelligence, 18(1):87-127, 1982.
24. O. Corcho, M. Fernández-López, A. Gómez-Pérez, and O. Vicente: *WebODE – An Integrated Workbench for Ontology Representation, Reasoning and Exchange*. Proceedings of the Thirteenth International Conference on Knowledge Engineering and Knowledge Management (EKAW'02), LNAI 2473, pages 138-153, Sigenza, Spain, October 2002.
25. E. Sirin, J. Hendler, and B. Parsia: *Semi-automatic Composition of Web Services using Semantic Descriptions*. Proceedings of the Workshop on Web Services: Modeling, Architecture and Infrastructure in conjunction with ICEIS2003. 2003. Accepted.
26. S. Narayanan and S.A. McIlraith: *Simulation, Verification and Automated Composition of Web Services*. Proceedings of the Eleventh International World Wide Web Conference (WWW-2002), pages 77-88, Hawaii, USA, May 2002.

APPLICATIONS OF PSL TO SEMANTIC WEB SERVICES

MICHAEL GRÜNINGER

ABSTRACT. In this paper we will show how the ontology of the Process Specification Language can be used as an upper-level process ontology that serves as the semantic foundation for the DAML-S ontology for web services.

1. SEMANTICS FOR WEB SERVICES

To achieve the vision of the Semantic Web, software agents will need a computer-interpretable description of the services they offer and the information that they access. Such a description can be provided by an ontology, which explicitly represents the intended meanings of the terms being used. Within the DARPA Agent Markup Language programme, an ontology of services called DAML-S has been proposed to support the discovery, invocation, and composition of the services offered by software agents on the Semantic Web.

The Process Specification Language (PSL) ([2], [4], [5]) has been designed to facilitate correct and complete exchange of process information among manufacturing systems¹. Included in these applications are scheduling, process modeling, process planning, production planning, simulation, project management, workflow, and business process reengineering. In this paper we will show how PSL can be used as an upper-level process ontology that serves as the semantic foundation for an ontology for web services that extends DAML-S.

Any ontology that supports the representation of web services will consist of generic classes to support service specification as well as classes of constraints in service specifications, such as ordering, temporal, occurrence, and duration.

The ontology must also support reasoning problems for web service specifications such as determining the consistency of a service specification and the composability of services, particularly with incomplete service specifications.

The approach taken in this paper will be to specify a first-order semantics for DAML-S concepts through PSL translation definitions and then use the grammars associated with PSL classes as an abstract syntax for service specifications.

2. THE ROLE OF FIRST-ORDER LOGIC

The PSL Ontology is a set of theories in the language of first-order logic. There are several other approaches to semantics for web services, such as BPEL [1], for which Petri nets and π -calculus have been proposed as the basis for their semantics. However, a first-order semantics has several advantages. First, we can specify and implement inference techniques that are sound and complete with respect to models of the theories. Also, a process

¹PSL has been accepted as project ISO 18629 within the International Organisation of Standardisation, and as of October 2002, part of the work is under review as a Draft International Standard.

ontology with a first-order axiomatization can be more easily integrated with other ontologies (which are almost all first-order theories themselves). Finally, a first-order semantics allows a simple characterization of incomplete service specifications.

The semantics of a first-order theory are based on the notion of an interpretation that specifies a meaning for each symbol in a sentence of the language. In practice, interpretations are typically specified by identifying each symbol in the language with an element of some algebraic or combinatorial structure, such as graphs, linear orderings, partial orderings, groups, fields, or vector spaces; the underlying theory of the structure then becomes available as a basis for reasoning about the concepts and their relationships.

First-order logic is sound and complete – a theory is consistent if and only if there exists a model that satisfies the axioms of the theory. This allows us to evaluate the adequacy of the application’s ontology with respect to some class of structures that capture the intended meanings of the ontology’s terms by proving that the ontology obeys the following two fundamental theorems:

- Satisfiability: every structure in the class is a model of the ontology.
- Axiomatizability: every model of the ontology is isomorphic to some structure in the class.

The purpose of the Axiomatizability Theorem is to demonstrate that there do not exist any unintended models of the theory, that is, any models that are not specified in the class of structures. In general, this would require second-order logic, but the design of PSL makes the following assumption (hereafter referred to as the Interoperability Hypothesis): *The ontology supports interoperability among first-order inference engines that exchange first-order sentences.* By this hypothesis, we do not need to restrict ourselves to elementary classes of structures when we are axiomatizing an ontology. Since the applications are equivalent to first-order inference engines, they cannot distinguish between structures that are elementarily equivalent. Thus, the unintended models are only those that are not elementarily equivalent to any model in the class of structures.

Classes of structures for theories within the PSL Ontology are therefore axiomatized up to elementary equivalence – the theories are satisfied by any model in the class, and any model of the core theories is elementarily equivalent to a model in the class. Further, each class of structures is characterized up to isomorphism.

3. PSL ONTOLOGY

Within the PSL Ontology, there is a further distinction between core theories and definitional extensions. Core theories introduce new primitive concepts, while all terms introduced in a definitional extension that are conservatively defined using the terminology of the core theories ².

3.1. Core Theories. All core theories within the ontology are consistent extensions of PSL-Core (T_{psl_core}), although not all extensions need be mutually consistent. Also, the core theories need not be conservative extensions of other core theories. The relationships among the core theories in the PSL Ontology are depicted in Figure 1.

²The complete set of axioms for the PSL Ontology can be found at <http://www.mel.nist.gov/psl/psl-ontology/>. Core theories are indicated by a .th suffix and definitional extensions are indicated by a .def suffix. As of June 2003, the ontology is in version 2.0.

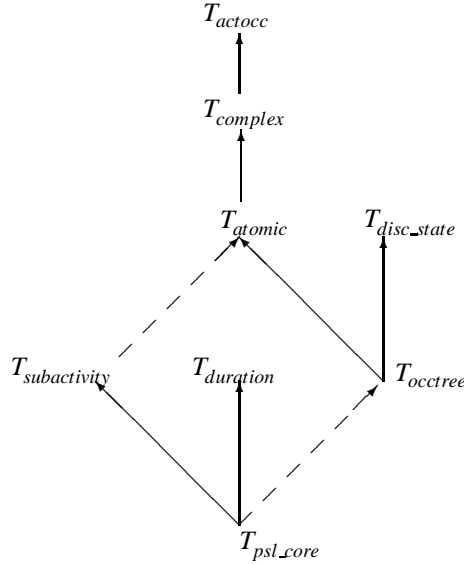


FIGURE 1. The core theories of the PSL Ontology. Solid lines indicate conservative extension, while dashed lines indicate an extension that is not conservative.

3.1.1. *Occurrence Trees.* The occurrence trees that are axiomatized in the core theory $T_{occtree}$ are partially ordered sets of activity occurrences, such that for a given set of activities, all discrete sequences of their occurrences are branches of the tree (see Figure 2). An occurrence tree contains all occurrences of *all* activities; it is not simply the set of occurrences of a particular (possibly complex) activity. Because the tree is discrete, each activity occurrence in the tree has a unique successor occurrence of each activity.

There are constraints on which activities can possibly occur in some domain. This intuition is the cornerstone for characterizing the semantics of classes of activities and process descriptions. Although occurrence trees characterize all sequences of activity occurrences, not all of these sequences will intuitively be physically possible within the domain. We will therefore want to consider the subtree of the occurrence tree that consists only of *possible* sequences of activity occurrences; this subtree is referred to as the legal occurrence tree.

3.1.2. *Discrete States.* The core theory T_{disc_state} introduces the notion of state (fluents). Fluents are changed only by the occurrence of activities, and fluents do not change during the occurrence of primitive activities. In addition, activities have preconditions (fluents that must hold before an occurrence) and effects (fluents that always hold after an occurrence).

3.1.3. *Subactivities.* The PSL Ontology uses the *subactivity* relation to capture the basic intuitions for the composition of activities. This relation is a discrete partial ordering, in which primitive activities are the minimal elements.

3.1.4. *Atomic Activities.* The core theory T_{atomic} axiomatizes intuitions about the concurrent aggregation of primitive activities. This concurrent aggregation is represented by the occurrence of concurrent activities, rather than concurrent activity occurrences.

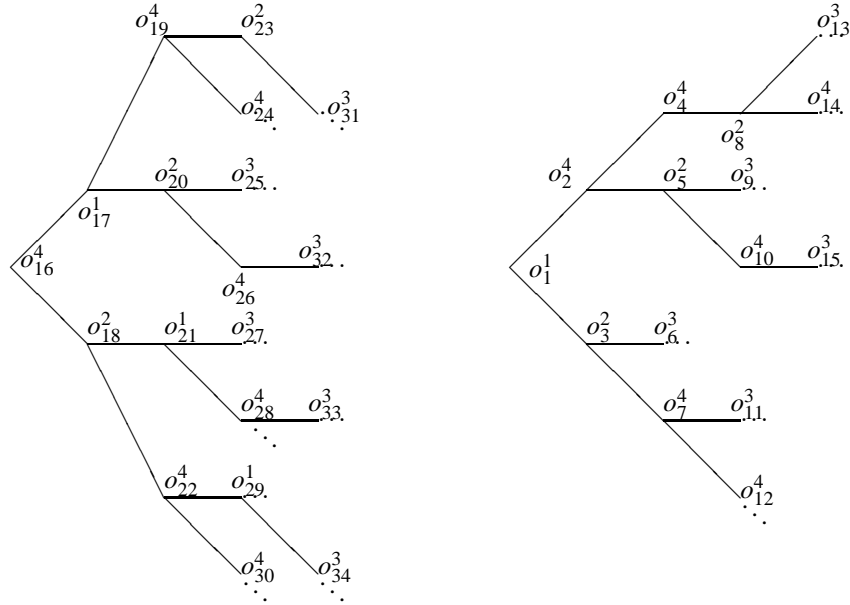


FIGURE 2. Example of legal occurrence trees. The elements o_i^1 denote occurrences of the activity a_1 , o_i^2 denote occurrences of the activity a_2 , o_i^3 denote occurrences of the activity a_3 , and o_i^4 denote occurrences of the activity a_4 . The activity occurrences o_1^1 and o_{16}^4 are the initial occurrences in their respective occurrence trees.

3.1.5. *Complex Activities.* The core theory $T_{complex}$ characterizes the relationship between the occurrence of a complex activity and occurrences of its subactivities. Occurrences of complex activities correspond to sets of occurrences of subactivities; in particular, these sets are subtrees of the occurrence tree. An activity tree consists of all possible sequences of atomic subactivity occurrences beginning from a root subactivity occurrence. In a sense, activity trees are a microcosm of the occurrence tree, in which we consider all of the ways in which the world unfolds *in the context of an occurrence of the complex activity*.

Different subactivities may occur on different branches of the activity tree i.e. different occurrences of an activity may have different subactivity occurrences or different orderings on the same subactivity occurrences. In this sense, branches of the activity tree characterize the nondeterminism that arises from different ordering constraints or iteration.

An activity will in general have multiple activity trees within an occurrence tree, and not all activity trees for an activity need be isomorphic. Different activity trees for the same activity can have different subactivity occurrences. Following this intuition, the core theory $T_{complex}$ does not constrain which subactivities occur. For example, conditional activities are characterized by cases in which the state that holds prior to the activity occurrence determines which subactivities occur. In fact, an activity may have subactivities that do not occur; the only constraint is that any subactivity occurrence must correspond to a subtree of the activity tree that characterizes the occurrence of the activity.

3.2. **Definitional Extensions.** Many ontologies are specified as taxonomies or class hierarchies, yet few ever give any justification for the classification. If we consider ontologies

of mathematical structures, we see that logicians classify models by using properties of models, known as invariants, that are preserved by isomorphism. For some classes of structures, such as vector spaces, invariants can be used to classify the structures up to isomorphism; for example, vector spaces can be classified up to isomorphism by their dimension. For other classes of structures, such as graphs, it is not possible to formulate a complete set of invariants. However, even without a complete set, invariants can still be used to provide a classification of the models of a theory.

Following this methodology, the set of models for the core theories of PSL are partitioned into equivalence classes defined with respect to the set of invariants of the models. Each equivalence class in the classification of PSL models is axiomatized using a definitional extension of PSL. In particular, each definitional extension in the PSL Ontology is associated with a unique invariant; the different classes of activities or objects that are defined in an extension correspond to different properties of the invariant. In this way, the terminology of the PSL Ontology arises from the classification of the models of the core theories with respect to sets of invariants. The terminology within the definitional extensions intuitively corresponds to classes of activities and objects.

4. TRANSLATION DEFINITIONS

Translation definitions specify the mappings between PSL and application ontologies. Such definitions have a special syntactic form – they are biconditionals in which the antecedent is a class in the application ontology and the consequent is a formula that uses only the lexicon of the PSL Ontology.

Translation definitions are generated using the organization of the definitional extensions. Each invariant from the classification of models corresponds to a different definitional extension. Any particular activity, activity occurrence, or fluent will have a unique value for the invariant. Each class of activity, activity occurrence, or fluent corresponds to a different value for the invariant. The consequence of a translation definition is equivalent to the list of invariant values for members of the application ontology class.

4.1. DAML-S Translation Definitions. In this section we will present the translation definitions³ for concepts in the DAML-S Process Ontology. Such translation definitions provide a first-order axiomatization of the intended semantics for the DAML-S constructs. Moreover, this axiomatization inherits the proofs of the Axiomatizability and Satisfiability Theorems from the underlying PSL Ontology.

4.1.1. *Atomic Activities.* The composedOf property in DAML-S is equivalent to the subactivity relation in PSL:

```
(forall (?a1 ?a2)
  (iff (composedOf ?a1 ?a2)
       (subactivity ?a2 ?a1)))
```

Within DAML-S, an AtomicProcess has no subprocesses; consequently, this corresponds to a primitive activity within PSL.

```
(forall (?a)
  (iff (AtomicProcess ?a)
       (and (primitive ?a)
            (markov_precond ?a)
            (or (markov_effects ?a))
```

³The translation definitions in this paper are written in the Knowledge Interchange Format. For more information on this language, see <http://cl.tamu.edu>.

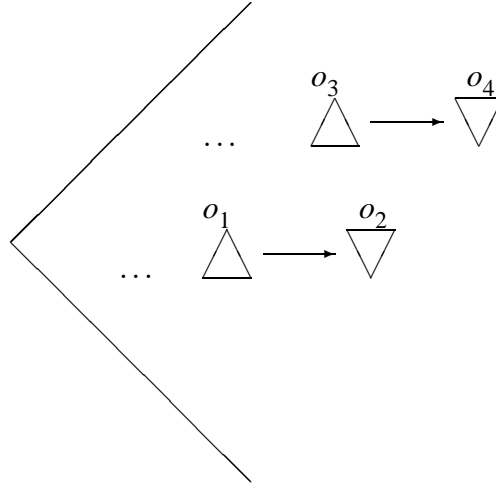


FIGURE 3. Example of activity trees for *transfer*, which is a Sequence DAML-S activity. o_1 and o_3 are occurrences of the subactivity *withdraw*, while o_2 and o_4 are occurrences of the subactivity *deposit*. Note that the diagram depicts two separate activity trees within a stylized legal occurrence tree.

(context_free ?a))))

The most common constraint on the legal occurrences of an activity specify the activity's preconditions. Activities whose preconditions depend only on the state prior to the occurrences The class of activities with markov preconditions is defined in the PSL definitional extension *state_precond.def*.

Effects characterize the ways in which activity occurrences change the state of the world. Such effects may be context-free, so that all occurrences of the activity change the same states, or they may be constrained by other conditions. The most common constraints are state-based effects that depend on the context; the class of activity associated with such constraints are defined as markov effect activities in the PSL extension *state_effects.def*.

A CompositeProcess in DAML-S is decomposable into other processes. Within PSL, the corresponding activity cannot be primitive; it will either be atomic (in which case it is a concurrent activity) or complex:

```
(forall (?a)
  (iff (CompositeProcess ?a)
    (and (activity ?a)
      (not (primitive ?a)))))
```

4.1.2. *Ordered Activities*. The classification of models within the the PSL Ontology leads to classes of activities, activity occurrences, and fluents. Classes of activity occurrences correspond to invariants for activity trees. The translation definitions for remaining DAML-S concepts are all related to invariants for activity trees.

Within DAML-S, a Sequence is a list of processes to be done in order (see Figure 3) ⁴. The translation definition for Sequence has two parts; one says that there exists an activity

⁴All of the examples in this section refer to the activities whose process descriptions are found in the Appendix.

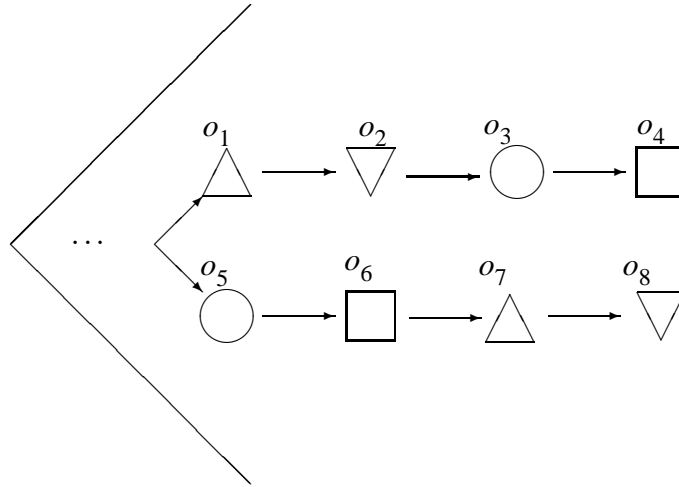


FIGURE 4. Example of an activity tree for *buy_product*, which is a Split DAML-S activity. For this purposes of this example, consider *transfer* to be a complex activity, with *deposit* and *withdraw* as subactivities.

tree for the activity which is ordered and which is simple and rigid (that is, there are no nontrivial permutations of subactivity occurrences). The second part says that the activity is uniform, that is, all activity trees for the activity are isomorphic: ⁵

```
(forall (?a)
  (iff (Sequence ?a)
    (and (uniform ?a)
      (exists (?occ)
        (and (occurrence_of ?occ ?a)
          (simple ?occ)
          (rigid ?occ)
          (ordered ?occ)
          (strong_poset ?occ))))))
```

In a DAML-S Split activity, sets of subactivities are performed in parallel (see Figure 4). Split activities differ from Sequence activities in that there exist nontrivial permutations of subactivity occurrences among the branches of the activity trees, so that the translation definition becomes:

```
(forall (?a)
  (iff (Split ?a)
    (and (uniform ?a)
      (exists (?occ)
        (and (occurrence_of ?occ ?a)
          (not (simple ?occ))
          (ordered ?occ))
```

⁵Two branches of an activity tree are isomorphic if there is a one-to-one mapping of subactivity occurrences that preserves the activities, e.g. occurrences of activity a_i are mapped to occurrences of a_i . Two activity trees are isomorphic if all of their branches are isomorphic. In the visual convention adopted in this paper, occurrences of different activities are depicted by different shapes; thus, a mapping that preserves activities will map a square to a square, a circle to a circle, and so on.

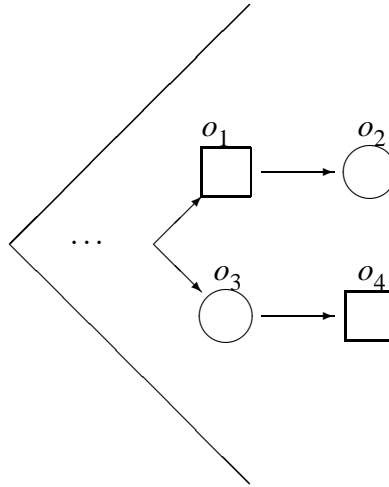


FIGURE 5. Example of an activity tree for *buy_product*, which is an Unordered DAML-S activity. For this purposes of this example, consider *transfer* to be a primitive activity; o_1 and o_4 are occurrences of the subactivity (transfer ?Fee ?Buyer ?Brokerer), o_2 and o_4 are occurrences of the subactivity (transfer ?Cost ?Buyer ?Seller).

(strong_poset ?occ))))))

For example, in Figure 4, the two branches of the activity tree consist of isomorphic subactivity occurrences that occur in different orderings on each branch.

According to [3], the Unordered construct allows process components to be executed in some unspecified ordered, although all components must be executed. This is equivalent to the class of bag activity trees within the PSL Ontology:

```
(forall (?a)
  (iff (Unordered ?a)
    (and (uniform ?a)
      (exists (?occ)
        (and (occurrence_of ?occ ?a)
          (bag ?occ))))))
```

In Figure 5, we see an example of an activity tree that is the unordered activity with two subactivities.

4.1.3. *Nondeterminism.* The simplest form of nondeterminism is captured by the class of activities in which some subactivity occurs (see Figure 6). Given this intended semantics, the translation definition to PSL would be:

```
(forall (?a)
  (iff (Choice ?a)
    (and (uniform ?a)
      (exists (?occ)
        (and (occurrence_of ?occ ?a)
          (simple ?occ)
          (rigid ?occ)
          (unordered ?occ))
```

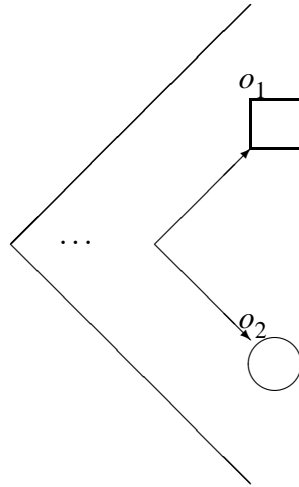


FIGURE 6. Example of an activity tree for a Choice DAML-S activity that is equivalent to a *choice_poset* in PSL. In this example, o_1 is an occurrence of a withdrawal from Account1 and o_2 is an occurrence of a withdrawal from Account3.

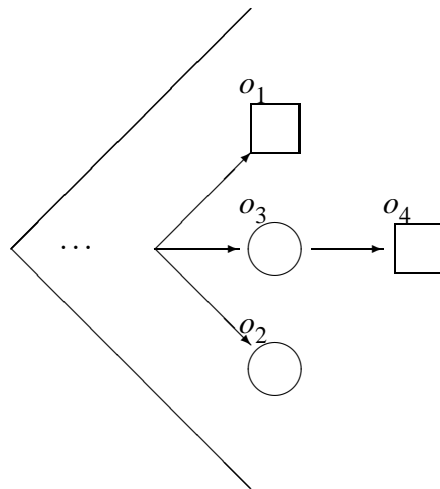


FIGURE 7. Example of an activity tree for a Choice DAML-S activity that is equivalent to a *weak_poset* in PSL.

(choice_poset ?occ))))))

There are some indications in [3] that the intended semantics for Choice activities is more general than this translation definition. For example, some possible applications of this construct may be intended to capture intuitions such as “choose subactivities and perform them in sequence” or “choose subactivities and perform them in parallel”. In such cases, the corresponding PSL class would be based on the notion of weak posets (see Figure 7), so that the translation definition would be:

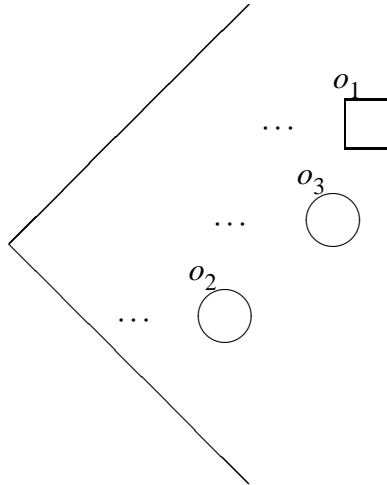


FIGURE 8. Example of activity trees for *withdraw*, which is an IfThenElse DAML-S activity. o_2 and o_2 are occurrences of the subactivity *change_balance*, and o_1 is an occurrence of the subactivity *notify*.

```
(forall (?a)
  (iff (Choice ?a)
    (and (uniform ?a)
      (exists (?occ)
        (and (occurrence_of ?occ ?a)
          (simple ?occ)
          (weak_poset ?occ)))))))
```

In addition, there are suggestions in [3] for extensions that construct new subclasses such as “choose exactly n subactivities from m ”. Such extensions do not correspond to any classes within Version 2.0 of the PSL Ontology.

4.1.4. *Conditional Activities*. The class of IfThenElse activities within DAML-S are equivalent to the conditional activities in PSL:

```
(forall (?a)
  (iff (IfThenElse ?a)
    (conditional ?a)))
```

Conditional activities are not uniform; however, if the same fluents hold prior to two occurrences of a conditional activity, then the activity trees for the activity are isomorphic. Figure 8 depicts three different activity trees, two of which are isomorphic.

4.1.5. *Iterated Activities*. The intended semantics of the Iterate process in DAML-S makes no assumption about how many iterations are made, or when to terminate. Within PSL, this corresponds to an activity in which there exist multiple isomorphic subtrees; for example, the activity tree in Figure 9 contains three subtrees that are isomorphic to the activity tree in Figure 6. Since different activity trees may have different numbers of iterations of the subactivities, the activity is not uniform. These considerations lead to the following translation definition:

```
(forall (?a)
```

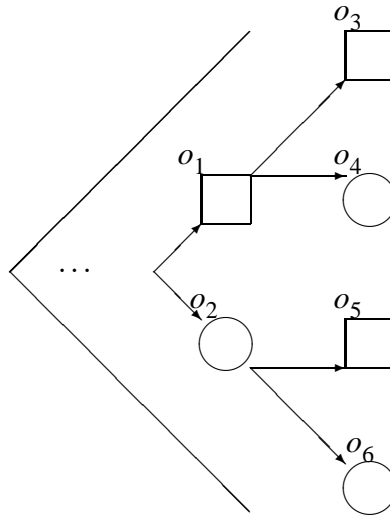



FIGURE 9. Example of an activity tree for an Iterate DAML-S activity.

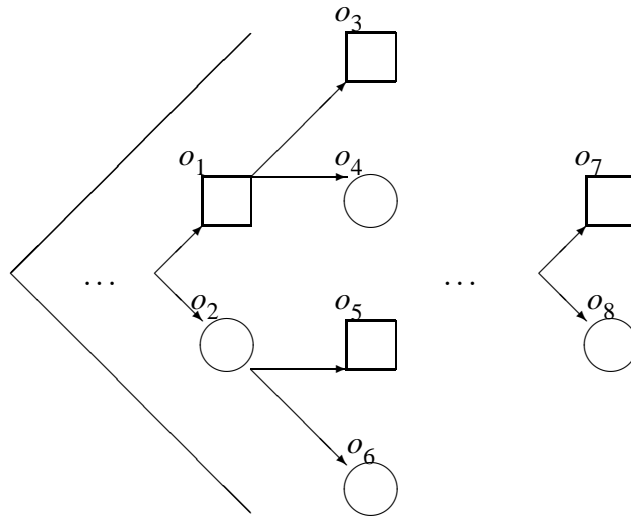


FIGURE 10. Example of activity trees for a RepeatUntil DAML-S activity.

```
(iff (Iterate ?a)
     (forall (?occ)
      (implies (occurrence_of ?occ ?a)
               (and (repetitive ?occ)
                    (multiple_outcome ?occ))))))
```

A RepeatUntil process in DAML-S executes until some state condition becomes true (see Figure 10). Because of this dependence on state, a RepeatUntil process is equivalent to an Iterate process which is conditional:

```
(forall (?a)
  (iff (RepeatUntil ?a)
    (and (conditional ?a)
      (forall (?occ)
        (implies (occurrence_of ?occ ?a)
          (and (repetitive ?occ)
            (multiple_outcome ?occ)))))))
```

Thus, there will exist multiple nonisomorphic activity trees (corresponding to occurrences of the activity with different iterations) and activity trees that agree on state will be isomorphic.

5. GRAMMARS FOR PROCESS DESCRIPTIONS

PSL makes a distinction between the ontology (which is the lexicon together with an axiomatization of their intended meaning) and the process descriptions that are exchanged between software applications. For each class in the ontology, PSL specifies a grammar that is satisfied by process descriptions of the activities or activity occurrences in that class.

For example, if two software applications both used an ontology for algebraic fields, they would not exchange new definitions, but rather they would exchange sentences that expressed properties of elements in their models. For algebraic fields, such sentences are equivalent to polynomials. Similarly, the software applications that use PSL do not exchange arbitrary sentences, such as new axioms or even conservative definitions, in the language of their ontology. Instead, they exchange process descriptions, which are sentences that are satisfied by particular activities, occurrences, states, or other objects.

DAML-S specifications are in fact grammars for service specifications. Using the translation definitions proposed in the previous section, we can use the grammars associated with the classes in the PSL Ontology to characterize the correctness and completeness of the DAML-S specification for the corresponding DAML-S constructs.

There are several classes within the DAML-S Ontology that are classes of sentences rather than classes of activities, activity occurrences, or fluents. In particular, DAML-S has two classes of conditions, *ConditionalEffects* and *UnconditionalEffects*. Within the PSL Ontology, this distinction is captured by the classes of *context_free* and *markov_effects* activities. If one considers the PSL process description grammars for a *context_free* activity, conditions appear as a class of formulae, but they are not a class in the ontology. Similarly comments apply to conditional activities. For example, in the process description for *withdraw* in the Appendix, the condition is the formula

```
(and (prior (balance ?account ?Balance) (root_occ ?occ))
  (greaterEq ?Balance ?amount)))
```

6. SUMMARY

Within the increasingly complex environments of enterprise integration, electronic commerce, and the Semantic Web, where process models are maintained in different software applications, standards for the exchange of this information must address not only the syntax but also the semantics of process concepts.

DAML-S is an attempt to support semantic web services within the framework of the DARPA Agent MARKup Language. However, the intended semantics of the concepts in DAML-S cannot be axiomatized within the Ontology Web Language, and the DAML-S ontology itself combines object level classes of concepts together with metalevel classes of sentences.

The PSL Ontology draws upon well-known mathematical tools and techniques to provide a robust semantic foundation for the representation of process information. This foundation includes first-order theories for concepts together with complete characterizations of the satisfiability and axiomatizability of the models of these theories. The PSL Ontology also provides a justification of the taxonomy of activities by classifying the models with respect to invariants. Finally, process descriptions are formally characterized as syntactic classes of sentences that are satisfied elements of the models.

The translation definitions presented in this paper are the first step towards laying firm logical foundations for semantic web services specified in DAML-S. Through these definitions, DAML-S can be given a sound and complete axiomatization and ontological distinctions can be clarified.

REFERENCES

- [1] Business Process Execution Language for Web Services, Version 1.0
<http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>
- [2] Gruninger, M. (2003) A Guide to the Ontology of the Process Specification Language", in *Handbook on Ontologies in Information Systems*, R. Studer and S. Staab (eds.). Springer-Verlag.
- [3] McIlraith, S., Son, T.C. and Zeng, H. (2001) Semantic Web Services, *IEEE Intelligent Systems*, Special Issue on the Semantic Web. 16:46–53, March/April, 2001.
- [4] Menzel, C. and Gruninger, M. (2001) A formal foundation for process modeling, *Second International Conference on Formal Ontologies in Information Systems*, Welty and Smith (eds), 256-269.
- [5] Schlenoff, C., Gruninger, M., Ciocoiu, M., (1999) The Essence of the Process Specification Language, *Transactions of the Society for Computer Simulation* vol.16 no.4 (December 1999) pages 204-216.

APPENDIX: EXAMPLES OF PROCESS DESCRIPTIONS

To buy a product, pay a fee to the broker and the cost of the product to the seller, performing these steps in parallel.

The PSL process description for *buy_product* is:

```
(forall (?x ?y ?z) (subactivity (transfer ?x ?y ?z) (buy_product ?y)))
(forall (?x ?y ?z) (subactivity (withdraw ?x ?y) (transfer ?x ?y ?z)))
(forall (?x ?y ?z) (subactivity (deposit ?x ?z) (transfer ?x ?y ?z)))

(forall (?occ ?Buyer)
  (implies (occurrence_of ?occ (buy_product ?Buyer))
    (exists (?occ1 ?occ2 ?Fee ?Cost ?broker ?Seller)
      (and (occurrence_of (transfer ?Fee ?Buyer ?Broker))
        (occurrence_of (transfer ?Cost ?Buyer ?Seller))
        (subactivity_occurrence ?occ1 ?occ)
        (subactivity_occurrence ?occ2 ?occ)))))
```

To transfer money from Account1 to Account2, withdraw some amount from Account1 and deposit the amount in Account2.

The PSL process description for *transfer* is:

```
(forall (?occ)
  (implies (occurrence_of ?occ (transfer ?Amount ?Account1 ?Account2))
    (exists (?occ1 ?occ2 ?occ3)
      (and (occurrence_of ?occ1 (withdraw ?Amount ?Account1))
        (occurrence_of ?occ2 (deposit ?Amount ?Account2))
        (subactivity_occurrence ?occ1 ?occ)))))
```

```

(subactivity_occurrence ?occ2 ?occ)
(leaf_occ ?occ3 ?occ1)
(min_precedes ?occ3 (root_occ ?occ2))))))

```

To withdraw money from an account, if the amount is greater than the balance, then change the account balance, otherwise notify the account that there are insufficient funds available.

Suppose

```

(forall (?x ?y ?z) (activity (change_balance ?x ?y ?z)))

(subactivity (change_balance ?Account ?Balance1 ?Balance2)
  (deposit ?Amount ?Account))

(subactivity (change_balance ?Account ?Balance1 ?Balance2)
  (withdraw ?Amount ?Account))

(subactivity (notify ?Account)
  (withdraw ?Amount ?Account))

```

In this case, *deposit* and *withdraw* are conditional activities, with the following PSL process descriptions:

```

(forall (?occ)
  (and (implies (and (occurrence_of ?occ (withdraw ?Amount ?Account))
    (prior (balance ?account ?Balance) (root_occ ?occ))
    (greaterEq ?Balance ?amount))
    (exists (?occ1)
      (and (occurrence_of ?occ1 (change_balance ?account ?Balance
        (plus ?Balance ?Amount)))
        (subactivity_occurrence ?occ1 ?occ))))
    (implies (and (occurrence_of ?occ (withdraw ?Amount ?Account))
      (prior (balance ?account ?Balance) (root_occ ?occ))
      (lesser ?Balance ?amount))
      (exists (?occ2)
        (and (occurrence_of ?occ2 (notify ?Account))
          (subactivity_occurrence ?occ2 ?occ))))))

```

The effects of *change_balance* are:

```

(forall (?occ)
  (implies (and (occurrence_of ?occ (change_balance ?Account ?Amount1 ?Amount2))
    (leaf_occ ?occ1 ?occ))
    (and (holds (balance ?Account ?Amount2))
      (not (holds (balance ?Account ?Amount1))))))

```

H-MATCH: an Algorithm for Dynamically Matching Ontologies in Peer-based Systems ^{*}

S. Castano, A. Ferrara, and S. Montanelli

Università degli Studi di Milano

DICO - Via Comelico, 39, 20135 Milano - Italy

`{castano,ferrara,montanelli}@dico.unimi.it`

Abstract. In this paper, we present H-MATCH, an algorithm for dynamically matching distributed ontologies. By exploiting ontology knowledge descriptions, H-MATCH can be used to dynamically perform ontology matching at different levels of depth, with different degrees of flexibility and accuracy. H-MATCH has been developed in the HELIOS framework, conceived for supporting knowledge sharing and ontology-addressable content retrieval in peer-based systems.

1 Introduction

Ontologies are generally recognized as an essential tool for allowing communication and knowledge sharing among distributed users and applications, by providing a common understanding of a domain of interest. Due to the vision of the Semantic Web, a large body of research is being moving around ontologies, and contributions have been produced regarding methods and tools for covering the entire ontology life cycle, from design to deployment and reuse [8], and ontology languages, such as OIL [9] or OWL [18]. As a matter of fact, when considering distributed contexts, the knowledge of interest is generally provided by many different ontologies. For instance, the vision of the Semantic Web envisages the Web enriched with several domain ontologies, which specify formal semantics of data, for different intelligent services for information sharing, search, retrieval, and transformation [3, 11]. As another example, the problem of distributed knowledge sharing is eminent in the P2P area and is receiving a lot of attention in the research community [10, 15]. Basically, peers need to perform content retrieval by interacting with other peers of the network, and queries have to be routed and resolved based on knowledge descriptions available at the peers. To enable information processing and content retrieval in distributed contexts with a multitude of autonomous ontologies, appropriate *matching techniques* are required to determine semantic mappings between concepts of different ontologies that are semantically related [2, 7, 17]. Some research work on this topic has recently appeared. We review such work in the related work section of the paper.

^{*} This paper has been partially funded by “Wide-scalE, Broadband, MiddleWare for Network Distributed Services (WEB-MINDS)” FIRB Project funded by the Italian Ministry of Education, University, and Research.

An important requirement to be considered in developing ontology matching techniques for distributed contexts, such as the P2P, is related to the inherent dynamicity of the context, and to the need of matching techniques that are conceived to operate in a dynamic fashion. In this paper, we present H-MATCH, an algorithm for dynamically matching distributed ontologies. H-MATCH has been developed in the framework of HELIOS, the infrastructure we have conceived for supporting knowledge sharing and ontology-addressable content retrieval in peer-based systems [5, 6]. After introducing the reference architecture of a HELIOS peer ontology, we show how the ontology knowledge description can be exploited to perform dynamic ontology matching at different levels of depth, with different degrees of flexibility and accuracy.

The paper is organized as follows. In Section 2, we provide the main motivations of our work. In Section 3, we present the HELIOS ontology model for knowledge representation. In Section 4, we describe the foundations of our approach for ontology matching. In Section 5, we present the H-MATCH algorithm for semantic affinity evaluation. In Sections 6 and 7, we compare our approach with other recent approaches for distributed ontology matching, by showing the original contribution of our work. Finally, in Section 8, we give our concluding remarks.

2 Motivating scenario

To address the requirements of knowledge sharing and ontology matching in distributed systems, we consider a typical P2P scenario, where a number of peers can acquire or extend their knowledge by interacting with other peers of the network. As shown in Figure 1, we suppose that the peer A wants to enlarge its knowledge about the concept of **Book** by learning which nodes own concepts with semantic affinity with it. This requires capability to describe the knowledge owned by a peer and to match an incoming request against the knowledge of a peer, to find semantically related information to be returned to the requesting peer. The HELIOS (Helios Evolving Interaction-based Ontology knowledge Sharing) framework has been conceived to enable knowledge sharing and evolution considering a P2P system where nodes are equipotential in terms of functionalities and capabilities. The knowledge sharing and evolution processes in HELIOS are based on *peer ontologies*, describing the knowledge of each peer, and on *interactions among peers*, allowing information search and knowledge acquisition/extension, according to pre-defined *query models* and semantic techniques for ontology matching. Each peer has a different amount of knowledge, that depends on the interactions it has performed in the network. Each peer can acquire new knowledge and/or extend his knowledge only by querying peers which have this information. *Probe queries* are sent by a peer interested in extending its knowledge of the network. Each peer having concepts matching the target concept(s) of a probe query can answer to the requesting peer. When the peer A asks for semantically related contents about the target **Book** concept, peer B and peer C evaluate the semantic affinity between **Book** and the concepts contained in their respective peer ontology. The semantic affinity evaluation pro-

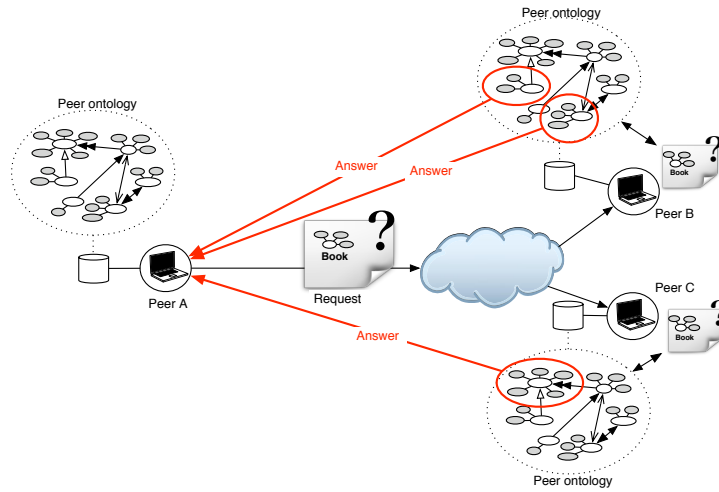


Fig. 1. Example of request/answer in the HELIOS network

cedure is based on the execution of the H-MATCH algorithm which determines the level of affinity of each concept in the peer ontology of peer B and peer C and the Book concept. Concepts having a high affinity value with Book are finally returned by peer B and peer C to the requesting peer A.

In the remainder of the paper, we focus on the formalization of the peer ontology knowledge model and on the H-MATCH algorithm for ontology matching.

3 Peer ontology representation

In this section, we provide a description of the architecture of a peer ontology and we formalize the peer ontology model adopted in HELIOS.

3.1 Ontology architecture

The ontology of a HELIOS peer is organized as a two-layer ontology, where the upper layer represents the *content knowledge* and the lower layer represents the *network knowledge* (See Figure 2).

The **Content Knowledge Layer** describes the knowledge of a peer, namely the knowledge a peer brings to the network and the knowledge the peer has of the network contents. We conceptualize the content knowledge layer as a network of *content concepts*, where each content concept is characterized by a set of *properties* and a set of *semantic relations* with other content concepts. A generic peer P can increase its content knowledge by adding new content concepts and/or by enriching existing content concept descriptions in terms of

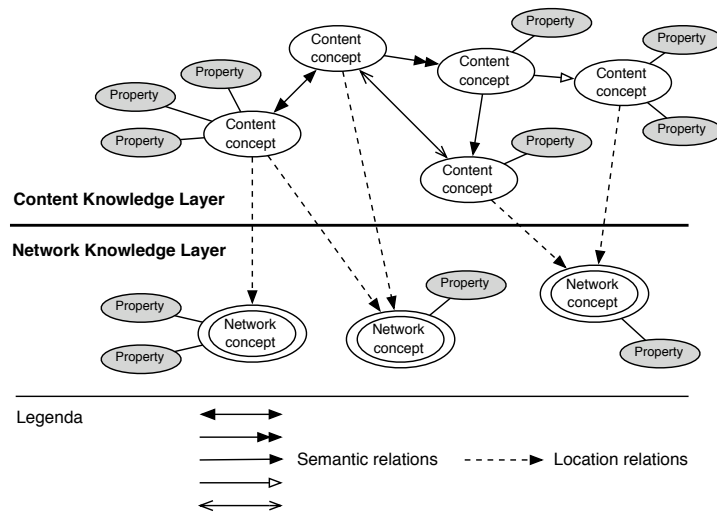


Fig. 2. Architecture of a peer ontology of a generic peer P

new properties and/or of new semantic relations, based on the answers acquired by other peers.

The **Network Knowledge Layer** describes the knowledge that a generic peer P has of other peers of the network it has interacted with. When a peer P receives a content concept from another peer P1, it stores in the network knowledge layer a description of the peer P1. Peer descriptions are given in form of *network concepts*, characterized by a set of properties describing the network features of a peer (e.g., IP-address).

An inter-layer relation, called *location relation* associates a content concept *cc* in the content knowledge layer with all network concept(s) describing peers storing concepts having semantic affinity with *cc*.

3.2 Peer ontology model

The peer ontology model organizes ontology knowledge in terms of *concepts*, *properties*, *semantic relations* and *location relations*, and is formally defined as follows.

Definition 1 (Peer Ontology). A peer ontology *PO* is a 4-tuple of the form $PO = (C, P, SR, LR)$, where:

- $C = CC \cup NC$ is a set of concepts of *PO*, where *CC* is a set of content concepts of the content knowledge layer, and *NC* is a set of network concepts of the network knowledge layer.

- P is a set of concept properties. A property $p \in P$ is defined as a unary relation of the form $p(c)$, where $c \in C$ is the concept associated to the property p .
- $SR = \{\text{same-as, kind-of, part-of, contains, associates}\}$ is a set of semantic relations between content concepts. A semantic relation $sr \in SR$ is defined as a binary relation of the form $sr(c, c')$, where c and $c' \in CC$ are the content concepts related through sr .
- LR is a set of location relations between content concepts and network concepts. A location relation $lr \in LR$ is defined as a binary relation of the form $lr(c, c')$, where $c \in CC$ is a content concept in the content knowledge layer and $c' \in NC$ is a network concept in the network knowledge layer, respectively.

To obtain a semantically rich and expressive representation of the knowledge in a peer ontology, we introduce the following semantic relations ¹:

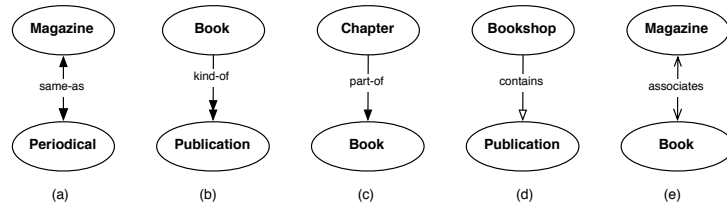


Fig. 3. Examples of semantic relations in a HELIOS peer ontology

- **Same-as.** The **same-as** relation is defined between two concepts c and c' which are considered semantically equivalent, that is, which denote the same real world entity or have the same meaning. As an example, we have **Same-as(Periodical, Magazine)** shown in Figure 3(a), referring to a peer ontology describing knowledge on publications.
- **Kind-of.** The **kind-of** relation defined between two concepts c and c' states that the concept c is a specialization of the concept c' . As an example, consider the case of **Kind-of(Book, Publication)** in Figure 3(b).
- **Part-of.** The **part-of** relation defined between two concepts c and c' states that the concept c represents a component of the concept c' as in the case of **Part-of(Chapter, Book)** shown in Figure 3(c).
- **Contains.** The **contains** relation defined between two concepts c and c' states that the concept c contains the concept c' as in the case of **Contains(Bookshop, Publication)** shown in Figure 3(d).

¹ The set SR of semantic relations has been defined according to relation classifications in ontology modelling [14] and metadata management [16] literature.

- **Associates.** The **associates** relation defined between two concepts c and c' states that a generic positive association is defined between c and c' . We use this relation when no other semantic relations hold between two concepts. As an example, consider the case of **Associates(Magazine, Book)** in Figure 3(e).

4 Foundations of ontology matching in HELIOS

The general goal of ontology matching techniques is to find concepts that have a semantic affinity with a target concept ². In this section, we propose an algorithm, called H-MATCH, for evaluating semantic affinity between concepts of different ontologies. In the context of HELIOS, we are interested in matching a target concept described in a query against a peer ontology (knowledge sharing), or in assimilating new concepts returned as the answer to probe queries into a peer ontology (knowledge evolution). H-MATCH grounds on the techniques developed in the ARTEMIS tool environment [1, 4] for the integration of heterogeneous data sources. In ARTEMIS, the semantic affinity evaluation is performed in the context of the schema matching process, in order to find mappings among elements of different source schemas that are semantically related for subsequent unification. In HELIOS, we extend and enrich the ARTEMIS techniques to address the typical problems of the ontology matching. In particular, the H-MATCH algorithm is based on the idea of considering both the linguistic features of concepts as well as the semantic relations among concepts in a peer ontology. Linguistic features are constituted by the semantic content of terms used as names of concepts and properties. The meaning of concepts is not established according to a given definition, but depends on the network of relations holding among terms (i.e., terminological relationships) and among concepts (i.e., semantic relations), respectively. Based on these considerations, the evaluation of the linguistic features is not based on a dictionary, where the meaning of each term depends on its definition, but on a thesaurus, where the meaning of each term is represented by the set of terminological relationships that it has with other terms in the thesaurus. Following the same approach, we assume that the meaning of a concept depends not only on its name, but also on its properties and on its semantic relations with other concepts in the ontology. To this purpose, the H-MATCH algorithm explicitly considers the *context* of each concept given by the set of its properties and of its adjacents (i.e., concepts which have a semantic relation with the considered concept), allowing a deep evaluation of semantic affinity between ontology concepts.

4.1 Linguistic interpretation

To capture the meaning of terms used as names of concepts and properties in a peer ontology, we exploit the terminological relationships among terms. In HELIOS, the network of terminological relationships is represented by a thesaurus,

² When speaking of concepts for matching, we refer to content concepts although not explicitly specified.

which is built by exploiting WordNet [13] as a source of lexical information, which can be possibly enriched by the ontology designer, if required. In particular, we consider a subset of the relations provided by WordNet represented by the following terminological relationships: {SYN (Synonym-of), BT/NT (Broader/Narrower Terms), RT (Related Terms)}, where the SYN relationship corresponds to the Synonym relation of WordNet, the BT/NT relationships correspond to the Hypernym/Hyponym relations of WordNet, and the RT relationship corresponds to the Meronym relation of WordNet, respectively. In the following, we denote by TR the set of terminological relationships in the HELIOS thesaurus.

4.2 Context interpretation

The H-MATCH algorithm evaluates the semantic affinity between two concepts by taking into account the affinity between their contexts. Given a concept $c \in CC$, we denote by $P(c) = \{p_i \mid p_i(c)\}$ the set of properties of c , and by $SR(c) = \{c_j \mid sr_j(c, c_j)\}$ the set of adjacents of c , namely all concepts c_j which have a semantic relation sr_j with c . The context of a concept is defined as follows:

Definition 2 (Concept context). *The context $Ctx(c)$ of a concept $c \in CC$ is defined as the union of the properties and of the adjacents of c , that is, $Ctx(c) = P(c) \cup SR(c)$.*

An example of concept context for the **Volume** concept is shown in Figure 4, where content concepts are represented as white ovals, properties are represented as grey ovals, and relations as arrows, respectively.

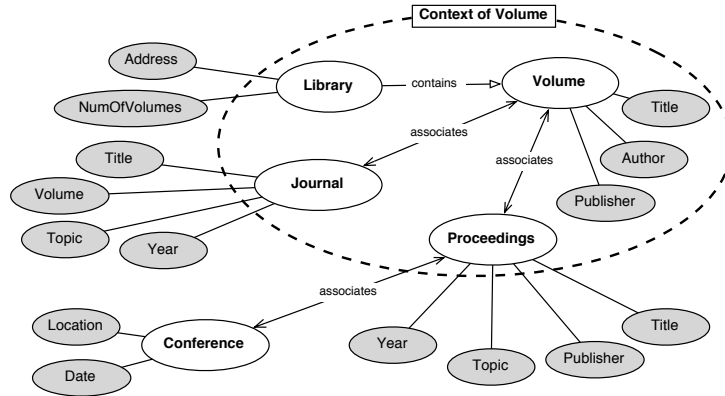


Fig. 4. Example of context for the **Volume** concept in a peer ontology

5 The H-MATCH algorithm

The semantic affinity between two ontology concepts c and c' is evaluated in HELIOS by weighting both the terminological relationships in the thesaurus and the semantic relations in the contexts of c and c' , respectively. In Table 1, we report the weights associated with each kind of terminological relationship and semantic relation, respectively. The weights associated with the terminological

	Relation	Weight
Linguistic interpretation	SYN	1.0
	BT/NT	0.8
	RT	0.5
Context interpretation	property	1.0
	same-as	1.0
	kind-of	0.8
	part-of	0.7
	contains	0.5
	associates	0.3

Table 1. Weights associated with terminological and semantic relations

relationships are taken from ARTEMIS, where they have been tested on several real integration cases. The weights associated with semantic relations have been defined in HELIOS to express a measure of the strength of the concept connection posed by each relation for semantic affinity evaluation purposes. The higher is the weight associated with a semantic relation, the higher is the strength of the semantic connection between concepts. Furthermore, we associate the weight 1.0 with properties since they are strongly related to a concept and provide its structural description. The weight associated with the terminological relationships are exploited for performing linguistic affinity evaluation, while the weights associated with properties and semantic relations are exploited for performing contextual affinity evaluation, respectively.

5.1 Linguistic affinity

The aim of the linguistic affinity is to evaluate the semantic affinity between two concepts by considering the semantic contents of their names as terms in the thesaurus. An affinity function $LA(t, t')$ is defined to evaluate the affinity between two terms t and t' , as shown in Figure 5. The affinity $LA(t, t')$ of two terms t and t' is equal to the highest-strength path of terminological relationships between them in the thesaurus, if at least one path exists, and is zero otherwise. Given t and t' and a path of terminological relationships between them, the strength of

```

function  $LA(t, t')$ 
input two terms  $t$  and  $t'$ 
output linguistic affinity value between  $t$  and  $t'$ 
begin function
  def  $x = 0, y = 1;$ 
  if exists a path  $P$  of terminological relationships  $tr_i \in TR$  between  $t$  and  $t'$ 
    /*  $\sigma_{tr_i}$  is the weight associated with each  $tr_i \in P$  */
    for each  $P$ 
       $y = 1;$ 
      for each  $tr_i \in P$ 
         $y = y \cdot \sigma_{tr_i};$ 
      if  $y \geq x$ 
         $x = y;$ 
  return  $x;$ 
end function

```

Fig. 5. The $LA()$ function for linguistic affinity evaluation

this path is computed by multiplying the weights of all terminological relationships forming the path.

Example 1. As an example of linguistic affinity evaluation, we consider the portion of thesaurus shown in Figure 6. Suppose we are interested in the linguistic affinity of concepts **Book** and **Publication**. Two paths exist between **Book** and **Publication** in the thesaurus. The first path P1 is $\{NT(\text{Book}, \text{Publication})\}$. The second path P2 is composed by $\{RT(\text{Book}, \text{Heading}), RT(\text{Heading}, \text{Publication})\}$. A graphical representation of the thesaurus graph and of the results of the linguistic affinity evaluation are shown in Figure 6. The linguistic affinity of **Book**

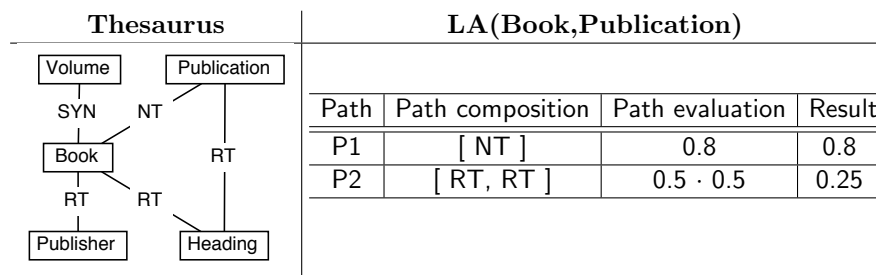


Fig. 6. Example of linguistic affinity evaluation between the **Book** and **Publication**

and **Publication** is 0.8, obtained by considering the path P1.

5.2 Contextual affinity

The aim of the contextual affinity is to calculate a measure of affinity between concepts based on their contexts. To this purpose, we evaluate the linguistic affinity of properties and adjacents, as well as the degree of closeness between the semantic relations that are involved in concept contexts.

Relation affinity function. The aim of the relation affinity function is to calculate a measure of closeness between two semantic relations or between a semantic relation and a property, based on their associated weights (see Table 1). Function $RA(r, r')$ is defined to evaluate the affinity between r and r' , where r and r' are either two semantic relations or a semantic relation and a property, respectively. The relation affinity function $RA(r, r')$ is reported in Figure 7. The relation

```
function  $RA(r, r')$ 
input relations  $r$  and  $r'$ 
output relational affinity value between  $r$  and  $r'$ 
begin function
  def  $\sigma_r, \sigma_{r'}$  as the weights associated with  $r$  and  $r'$ , respectively
  def  $x = 0$ ;
   $x = 1 - |\sigma_r - \sigma_{r'}|$ ;
  return  $x$ ;
end function
```

Fig. 7. The $RA()$ function for relational affinity evaluation

affinity is a value in the range $[0,1]$ and is proportional to the level of closeness of the considered relations. The highest value (i.e., 1.0) is obtained when r and r' have the same weight. The higher the difference between the weights associated with the relations, the lower the relation affinity value.

Evaluation of the contextual affinity. The contextual affinity evaluation is performed by exploiting a function $CA(CV(c), CV(c'))$ on the contexts of two concepts c and c' . In this function, context $Ctx(c)$ of a concept c is represented through a context vector $CV(c) = (cv_1, \dots, cv_n)$, where $\forall i \in (1, \dots, n), cv_i = (f_i, r_i)$, where f_i denotes either a property or an adjacent concept of c , and r_i denotes the semantic relation between c and f_i . The contextual affinity function is defined as shown in Figure 8.

Based on some experimental results, we noted that in the contextual affinity evaluation the impact of the concepts with low affinity is stronger than the impact of the concepts with a high affinity, thus originating biased measures. For this reason, a control factor F_k has been introduced for refining the results of the contextual affinity evaluation. In particular, in presence of very low affinity values, F_k proportionally increases them, in order to better balance all affinity values in the context and avoid too large gaps between affinity results.

```

function CA(CV(c), CV(c'))
input the context vectors CV(c) and CV(c') representing the contexts of
the concepts c and c', respectively
output contextual affinity value of c and c'
begin function
  def x = 0, y = 0, z = 0;
  foreach cv ∈ CV(c) | cv = (f, r);
    foreach cv' ∈ CV(c') | cv' = (f', r');
      y = LA(f, f') · RA(r, r');
      z = z + y;
  z = z ÷ (length(CV(c)) · length(CV(c')));
  /* Fk = 1 + (1 - z) is a control factor */
  x = z · Fk;
  return x;
end function

```

Fig. 8. The CA() function for contextual affinity evaluation

Example 2. As an example of the contextual affinity evaluation, we consider the concepts Book and Volume shown in Figure 9, with their respective contexts:

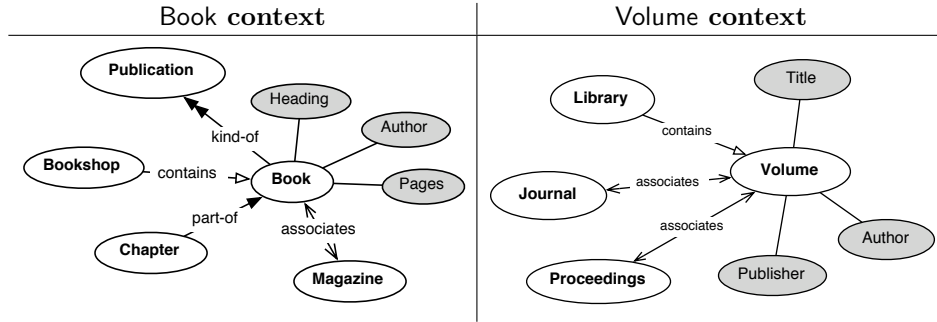


Fig. 9. The contexts of the Book and Volume concepts

$$CV(\text{Book}) = [(\text{Heading, property}), (\text{Author, property}), (\text{Pages, property}), (\text{Magazine, associates}), (\text{Chapter, part-of}), (\text{Bookshop, contains}), (\text{Publication, kind-of})]$$

$$CV(\text{Volume}) = [(\text{Title, property}), (\text{Author, property}), (\text{Publisher, property}), (\text{Proceedings, associates}), (\text{Journal, associates}), (\text{Library, contains})]$$

The linguistic affinity and the relation affinity are evaluated as shown in Table 2. The contextual affinity $CA(CV(\text{Book}), CV(\text{Volume}))$ is evaluated by exploiting

Linguistic affinity ($CV(\text{Book}), CV(\text{Volume})$)							
$LA()$	Heading	Author	Pages	Magazine	Chapter	Bookshop	Publication
Title	0.5	0.25	0.25	0.25	0.25	0.25	0.4
Author	0.25	1.0	0.25	0.25	0.25	0.25	0.4
Publisher	0.25	0.0	0.25	0.5	0.25	0.0	0.5
Proceedings	0.25	0.25	0.0	0.64	0.25	0.0	0.8
Journal	0.25	0.25	0.0	0.64	0.25	0.0	0.8
Library	0.25	0.25	0.0	0.5	0.25	0.5	0.5

Relation affinity ($CV(\text{Book}), CV(\text{Volume})$)							
$RA()$	property	property	property	associates	part-of	contains	kind-of
property	1.0	1.0	1.0	0.3	0.7	0.5	0.8
property	1.0	1.0	1.0	0.3	0.7	0.5	0.8
property	1.0	1.0	1.0	0.3	0.7	0.5	0.8
associates	0.3	0.3	0.3	1.0	0.6	0.8	0.5
associates	0.3	0.3	0.3	1.0	0.6	0.8	0.5
contains	0.5	0.5	0.5	0.8	0.8	1.0	0.7

Table 2. Linguistic and relation affinity evaluation for the contexts of Book and Volume

the $LA()$ and $RA()$ results, according to the function definition shown in Figure 8:

$$CA(CV(\text{Book}), CV(\text{Volume})) = (9.4 / 42) \cdot 1.78 = 0.40$$

5.3 The H-MATCH algorithm

The H-MATCH algorithm evaluates the semantic affinity between two concepts by considering both their linguistic and contextual affinity. H-MATCH can be configured for differently evaluating concept semantic affinity, by setting the impact of the linguistic and the contextual affinity, and by choosing dynamically which part of concept context has to be considered in the matching process. This flexibility of H-MATCH has the aim of facing two different requirements of the ontology matching process. The first requirement regards the balance between the linguistic and the contextual features of concepts in a peer ontology. The meaning of the peer ontology concepts depends basically on the terms used for their definition and on the relations they have with other concepts in the ontology. In HELIOS, we are interested in addressing the fact that those features can have a different impact in different ontology structures. A second requirement regards the context evaluation, in which we distinguish between properties and concepts. The role of the properties in the concept definition might have a different relevance in different peer ontologies. As an example, if a peer ontology

is defined describing high structured data sources (e.g., relational databases), the properties which describe the structure of each concept have a high impact on the concept meaning evaluation. Furthermore, the composition of the context and its extension in terms of number of adjacents have an impact on the matching quality and on its performance. The aim of H-MATCH is to allow a dynamic choice of the kind of features to be considered in the semantic affinity evaluation.

Matching models. In order to address these requirements, three different matching models are proposed in HELIOS to configure H-MATCH.

- *Shallow matching.* The shallow matching is performed by considering only the linguistic information provided by the concept names and by the reference thesaurus. The precision of the semantic affinity evaluation depends on the choice of the concept names in the ontology definition. Meaningful and precise names will guarantee more appropriate results. Being based only on linguistic information, the shallow matching guarantees a high performance since requires less computation than the other two models, and is recommended when only concept names are specified in a query.
- *Intermediate matching.* The intermediate matching is performed by considering concept names and also concept properties. With this model, we want a more accurate level of matching by taking into account the property part of the concept context.
- *Deep matching.* The deep matching model considers concept names and the whole context of concepts. The deep matching requires a complete description of target concept in the query and guarantees the highest level of precision in the semantic affinity evaluation. As such, it requires more computation than the other two, and is recommended when the accuracy is more important than the response time.

Linguistic and contextual information balancing. The problem of dynamically setting the balance between the linguistic and the contextual features of a peer ontology in the matching process is addressed in HELIOS by setting a weight $W_{LA} \in [0,1]$ which measures the degree of the impact of the linguistic affinity in the semantic affinity evaluation process.

H-MATCH algorithm. The input of the H-MATCH algorithm is constituted by: two concepts c and c' ; the matching model; the value of the weight W_{LA} . Deep and 0.5 are the default values for the matching model and W_{LA} , respectively. $W_{LA} = 0.5$ ensures that the linguistic affinity and the contextual affinity have the same impact in the semantic affinity evaluation. The output of H-MATCH is the semantic affinity value of c and c' , calculated as the weighted sum of their linguistic affinity and contextual affinity. The H-MATCH algorithm is shown in Figure 10. The algorithm exploits the $LA()$ and $CA()$ functions for evaluating the linguistic and the contextual affinity values, respectively. The choice of the matching model determines the composition of the context vectors used for the

```

algorithm H-MATCH( $c, c', model = \text{"deep"}, W_{LA} = 0.5$ )
input the concepts  $c$  and  $c'$ , the matching model  $\in [ \text{shallow}; \text{intermediate}; \text{deep} ]$ , and the weight  $W_{LA} \in [0,1]$ 
output the semantic affinity value between  $c$  and  $c'$ 
begin algorithm
  def  $t, t'$  as the names of  $c$  and  $c'$ , respectively;
  def  $CV(c) = [], CV(c') = []$  as the context vectors for  $c$  and  $c'$ , respectively;
  def  $context\_item = []$  as a pair of the form  $(f, r)$ , where  $f$  is a name associated with a property or a concept, and  $r \in \{ \text{property}; \text{same-as}; \text{kind-of}; \text{part-of}; \text{contains}; \text{associates} \}$ ;
  def  $x = 0, y = 0, semantic\_affinity = 0$ ;
   $x = LA(t, t')$ ;
  switch  $model$ 
    case "shallow" :
       $W_{LA} = 1$ ;
    case "intermediate" :
      foreach property  $p(c) \in Ctx(c)$ 
         $context\_item = [p(c), \text{"property"}]$ ;
        append  $context\_item$  to  $CV(c)$ ;
      foreach property  $p(c') \in Ctx(c')$ 
         $context\_item = [p(c'), \text{"property"}]$ ;
        append  $context\_item$  to  $CV(c')$ ;
    case "deep" :
      foreach property  $p(c) \in Ctx(c)$ 
         $context\_item = [p(c), \text{"property"}]$ ;
        append  $context\_item$  to  $CV(c)$ ;
      foreach concept  $c_i \in Ctx(c)$ 
        /*  $sr(c, c_i)$  is the semantic relation between  $c$  and  $c_i$  */
         $context\_item = [c_i, sr(c, c_i)]$ ;
        append  $context\_item$  to  $CV(c)$ ;
      foreach property  $p(c') \in Ctx(c')$ 
         $context\_item = [p(c'), \text{"property"}]$ ;
        append  $context\_item$  to  $CV(c')$ ;
      foreach concept  $c_j \in Ctx(c')$ 
        /*  $sr(c', c_j)$  is the semantic relation between  $c'$  and  $c_j$  */
         $context\_item = [c_j, sr(c', c_j)]$ ;
        append  $context\_item$  to  $CV(c')$ ;
   $y = CA(CV(c), CV(c'))$ ;
   $semantic\_affinity = W_{LA} \cdot x + (1 - W_{LA}) \cdot y$ ;
  return  $semantic\_affinity$ ;
end algorithm

```

Fig. 10. The H-MATCH algorithm

contextual affinity evaluation. If the shallow model is chosen, W_{LA} is set to 1, and only the linguistic affinity is considered. Otherwise, W_{LA} is exploited in order to correctly combine the linguistic affinity value with the contextual affinity value.

Example 3. Consider the concepts of **Book** and **Volume** of Example 2. Below, we report the semantic affinity of **Book** and **Volume** obtained by exploiting the H-MATCH algorithm according to the three different matching models, with $W_{LA} = 0.5$.

- **Shallow matching.** The shallow matching returns a semantic affinity value which coincides with the linguistic affinity value, that is:

$$\text{H-MATCH}(\text{Book}, \text{Volume}, \text{"shallow"}, 0.5) = 1$$

- **Intermediate matching.** The intermediate matching evaluates the linguistic and the contextual affinity, by considering only the properties in the contexts of **Book** and **Volume**, that is:

$$\text{H-MATCH}(\text{Book}, \text{Volume}, \text{"intermediate"}, 0.5) = 0.5 \cdot 1 + 0.5 \cdot 0.55 = 0.78$$

- **Deep matching.** The deep matching evaluates semantic affinity by considering the whole contexts of **Book** and **Volume**, that is:

$$\text{H-MATCH}(\text{Book}, \text{Volume}, \text{"deep"}, 0.5) = 0.5 \cdot 1 + 0.5 \cdot 0.40 = 0.7$$

Considerations. We note that in our example the deeper is the matching model used for semantic affinity evaluation, the lower is the semantic affinity returned for **Book** and **Volume**. It depends on the fact that considering the context of the concepts to be matched, H-MATCH is able to capture more precisely the differences between them than considering only the linguistic affinity. In particular, H-MATCH is useful in order to address the fact that the same concept can have a different meaning if used in different contexts. In our example, the **Book** and the **Volume** concepts, which are synonyms from a linguistic point of view, are used in a bookstore context and in a library context, respectively. The differences between the kind of publications contained in the bookstore context and in the library context are the reason of the decreasing value of semantic affinity when applying the deep match.

6 Related work

In this section, we overview the main approaches for ontology matching in distributed systems.

Edamok [17] is a research project focused on semantic interoperability issues in P2P systems. The project implements the KEx (Knowledge Exchange) P2P system which aims to realize knowledge sharing among peer communities of interest (called federations). The system is based on the concept of context of a peer, to represent the interests of the peer. KEx implements specific tools (e.g., context

editors, context extractors) to extract the context of a peer from the peer knowledge (e.g., file system, mail messages). In order to point out semantic mapping between concepts stored in distinct peers, the system uses the CTX-MATCH algorithm. This algorithm compares the knowledge contained in different contexts looking for semantic mappings denoting peers interested in similar concepts. These mappings are stored in order to assist the query resolution components to direct queries to peers which store relevant information. The CTX-MATCH is based on a *semantic explicitation* phase where concepts are associated with the correct meaning with respect to their context and on a *semantic comparison* phase where concepts are translated in logical axioms and matched. The algorithm implements a description logic approach: mapping discovering is reduced to the problem of checking a set of logical relations.

Chatty web [2] represents a novel approach for obtaining semantic interoperability among data sources in a semi-automatic manner. This approach applies to any system which provides a communication infrastructure (e.g., decentralized systems, P2P systems) and offers the opportunity to study semantic interoperability as a global phenomenon in a network of information sharing communities. Each peer offers data which are organized according to some schema expressed in a data model (e.g., relational, XML, RDF). Semantic interoperability is accomplished by assuming the existence of local agreements provided as mappings between different schemas. Peers introduce their own schemas and exchanging translations between them; then peers can incrementally come up with an implicit “consensus schema” which gradually improves the global search capabilities of the system. The paper identifies different methods that can be applied to establish global forms of agreement starting from a graph of local mappings among schemas and presents the *gossiping* algorithm which is used to identify the sufficiently large set of peers capable of rendering meaningful results on a specified query.

GLUE [7] is a system that employs machine learning techniques to find semantic mappings between concepts stored in distinct and autonomous ontologies. Given two distinct ontologies, the mapping discovery process between their concepts is based on the measure of similarity which is defined through the *joint probability distribution*. GLUE follows a probabilistic approach: the measure of similarity between the concepts A and B is computed as the likelihood that an instance belongs to both the concepts ($P(A \cap B)$). According to these probabilistic measurements, two base learning techniques are applied in order to build a similarity matrix expressing the prediction of semantic affinity between concepts. A *relaxation labeling* procedure is performed in order to improve the matching accuracy of the affinity predictions. Domain-independent and domain-dependent constraints are introduced to evaluate such kind of refinement process.

KAON [14] is an ontology and Semantic Web tool suite. In [14], the authors discuss the problem of ontology representation and querying for semantics-driven applications, describing a prototype implementation within the KAON system. In particular, the paper presents the mathematical definition of the KAON modeling language, and the denotational semantics for it. The ontology structure is

presented as a view of a general model, called OI-model, which consists of entities and may include a set of other OI-models. The ontology structure contains definitions specifying how instances should be constructed, and is composed by concepts and properties. The properties can have domain concepts, and relational properties can have range concepts. Relational properties may be marked as transitive and/or symmetric and it is possible to define inverse properties for each relational property. The emphasis of this system is on ontology definition and on formal properties for correctness and completeness.

The original contribution of our ontology matching techniques, with respect to these approaches, is the use of combined semantic affinity evaluation strategies to obtain a flexible and dynamic algorithm. The H-MATCH algorithm is able to discover the location of semantically related concepts to a target argument without requiring a complete description and matching procedure between independent ontologies. In the next section, we deeply compare H-MATCH with the approach adopted in Edamok, by discussing our contribution in more detail.

7 Applicability issues

We made a comparison of H-MATCH and the matching techniques developed in the Edamok [17] project, which are more strictly related to our approach. In particular, the aim of the comparison is to verify which mappings are discovered by the two techniques for a given concept, by considering as the reference case study the Art domain concept hierarchies of Google³ and Yahoo⁴ shown in Figure 11. In particular, we are interested in discovering which concepts of the Yahoo hierarchy match the Art history concept of the Google hierarchy. In [17], the following relations are discovered for the Art history concept:

$$\begin{aligned} \text{Arts/Art history} &\equiv \text{Arts \& Humanities/Art History} \\ \text{Arts/Art history} &\supseteq \text{Arts \& Humanities/Design Art/Architecture/History} \end{aligned}$$

In HELIOS, the H-MATCH algorithm is exploited to discover the semantic affinity value between the Art history concept and each concept of the Yahoo hierarchy. In this example, we set H-MATCH by choosing the deep matching model and $W_{LA} = 0.5$. In order to address the fact that the concept hierarchies are resource directories in Google and Yahoo, in HELIOS, we represent the is-a relations by means of the contains semantic relation. The linguistic and contextual affinity are evaluated as described in Section 4. In particular, the H-MATCH algorithm is performed by considering the context of the concept Art history and the contexts of the concepts in the Yahoo hierarchy, as shown in Figure 12. The results obtained with H-MATCH are the following:

³ www.google.com

⁴ www.yahoo.com

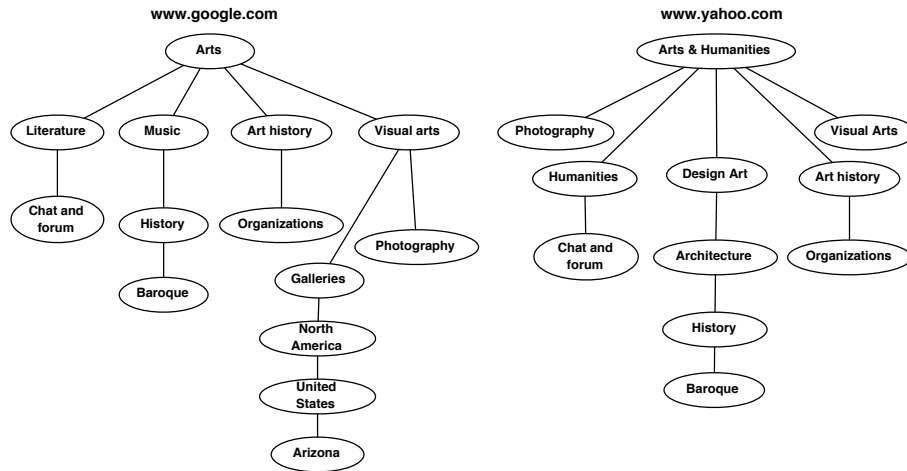


Fig. 11. The Art domain concept hierarchies of Google and Yahoo

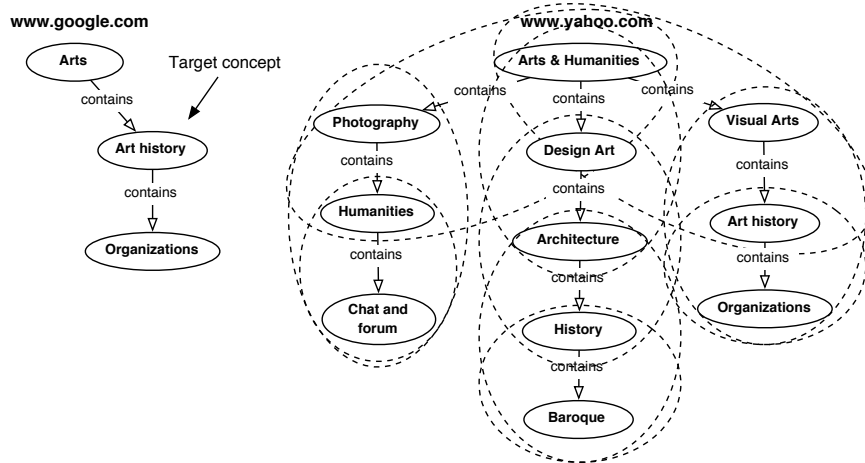


Fig. 12. Concept contexts involved in the semantic affinity evaluation between Art history of Google and the Yahoo concept hierarchy

H-MATCH(Art history, Art history)	= 1
H-MATCH(Art history, History)	= 0.72
H-MATCH(Art history, Photography)	= 0.57
H-MATCH(Art history, Visual arts)	= 0.57
H-MATCH(Art history, Design art)	= 0.55
H-MATCH(Art history, Arts & humanities)	= 0.54
H-MATCH(Art history, Humanities)	= 0.54
H-MATCH(Art history, Architecture)	= 0.5
H-MATCH(Art history, Baroque)	= 0.47
H-MATCH(Art history, Organizations)	= 0.22
H-MATCH(Art history, Chat & Forum)	= 0.21

A full comparison between our results and those discussed in [17] is not possible, because the H-MATCH algorithm results cannot be interpreted as semantic relations among the considered concepts. An interesting point about the comparison, is the fact that the concepts having the highest semantic affinity value with Art history in the H-MATCH results (i.e., Art history and History) are the same concepts discovered by the CTX-MATCH algorithm presented in [17]. In conclusion, the H-MATCH algorithm is a valid support for discovering, given a concept ontology, a set of corresponding concepts in another ontology. The main contribution of our techniques is the fact that H-MATCH gives a measure of correspondence in terms of semantic affinity among concepts. On these measures, a set of different interpretations are possible in order to define mappings between the considered concept ontologies. For instance, when using H-MATCH for query resolution a threshold is used in order to select the concepts which have the highest semantic affinity with the target concept in the query.

8 Concluding remarks

In this paper, we have presented the H-MATCH algorithm for dynamic distributed ontology matching. Considering the linguistic affinity evaluation as an atomic step, H-MATCH has a complexity of $O(N^2)$, being N the number of elements in the contexts of the concept to be matched. We are working in the direction of testing the algorithm on real matching cases in the context of HELIOS, to evaluate and experiment performance and scalability issues posed by ontology-based query resolution considering large ontologies.

The accuracy of the matching results depends on the thesaurus (e.g., WordNet) which may not be sufficient to precisely evaluate semantic similarity between two terms in different vocabularies. Such vocabulary heterogeneity may cause a loss of information. This problem has been discussed in [12], where measures and metrics are used to select results having the desired quality of information. To this end, future work will be devoted to the extension of our techniques taking into account aspects related to the quality of information.

References

1. The ARTEMIS project web site. <http://islab.dico.unimi.it/artemis/d2i/>.

2. K. Aberer, P. Cudrè-Mauroux, and M. Hauswirth. The chatty web: Emergent semantics through gossiping. In *Proc. of the Twelfth International World Wide Web Conference, (WWW2003)*, Budapest, Hungary, May 2003.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
4. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Data and Knowledge Engineering*, 13(2):277–297, 2001.
5. S. Castano, A. Ferrara, S. Montanelli, E. Pagani, and G.P. Rossi. Ontology-addressable contents in P2P networks. In *Proc. of WWW'03 1st SemPGRID Workshop*, Budapest, May 2003. <http://www.isi.edu/stefan/SemPGRID/proceedings/proceedings.pdf>.
6. S. Castano, A. Ferrara, S. Montanelli, and D. Zucchelli. HELIOS: a general framework for ontology-based knowledge sharing and evolution in P2P systems. In *Proc. of DEXA'03 2nd Web Semantics Workshop*, Prague, Czech Republic, September 2003.
7. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the Eleventh International World Wide Web Conference, (WWW2002)*, Honolulu, Hawaii, USA, May 2002.
8. D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, 2001.
9. D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *In Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000)*, pages 1–16, Juan-les-Pins, France, October 2000. Springer-Verlag.
10. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE'03*, Bangalore, India, March 2003.
11. J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent System*, 16:54–59, May 2001.
12. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Imprecise answers on highly open and distributed environments: An approach based on information loss for multi-ontology based query processing. *International Journal of Cooperative Information Systems (IJCIS)*, 9(4):403–425, December 2000.
13. A.G. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
14. B. Motik, A. Maedche, and R. Volz. A conceptual modeling approach for semantics-driven enterprise applications. In *Proc. of the First International Conference on Ontologies, Databases and Application of Semantics (ODBASE-2002)*, 2002.
15. Nejdl et al. EDUTELLA: a P2P networking infrastructure based on RDF. In *Proc. of the Eleventh International World Wide Web Conference, WWW2002*, Honolulu, Hawaii, USA, May 2002.
16. R.A. Pottinger and P.A. Bernstein. Merging models based on given correspondences. Technical report, University of Washington, February 2003. Available at <ftp://ftp.cs.washington.edu/tr/2003/02/UW-CSE-03-02-03.pdf>.
17. L. Serafini, P. Bouquet, B. Magnini, and S. Zanobini. An algorithm for matching contextualized schemas via SAT. Technical report, DIT University of trento, Italy, January 2003. Available at <http://eprints.biblio.unitn.it/archive/00000348/>.
18. F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, and L.A. Stein. OWL (march 2003) reference description. <http://www.w3.org/TR/2003/WD-owl-ref-20030331/>.

A Collaborative Approach for Query Propagation in Peer-to-Peer Systems¹

Anne Doucet, Nicolas Lumineau

LIP6 Laboratory
University of Paris 6
8, Rue du Capitaine Scott, 75015 Paris, France
FirstName.LastName@lip6.fr

Abstract. Sharing resources on a world-wide scale is a current research topic. Nowadays, peer-to-peer architecture is considered as a scalable solution to this issue. However, a lot of problems to extend this architecture for data storage remain open. When information is not highly replicated, localization of nodes storing relevant data becomes essential to avoid covering completely the network. To this purpose, we try to propagate a query towards nodes potentially storing relevant data. Information about nodes relevance is obtained by users' experiences. In an applicative context where "communities of interest" exist, we create logical and semantic links not only to specify the nodes relevant for a community, but also to link communities with a related interest. The proposed pattern has been created with respect to Peer-to-Peer philosophy and so as to consider the evolution of communities.

1 Introduction

With the development of Internet, the amount of available resources has drastically increased. This leads to revisit the notion of information access in this context. Techniques used in federated or distributed databases, which were developed to manage data distributed on a limited number of servers, have reached their limits in the context of Internet, and do not support scalability. A new class of distributed architecture, peer-to-peer (P2P) systems [1], is designed to support thousands of nodes, providing thus interesting solutions to scalability. In such systems, nodes are indifferently client and server. A query on P2P systems is based on the propagation from node to node until the number of rebounds between nodes is considered sufficient. The node receiving a query solves it locally, and propagates it to a set of neighbors.

Peer-to-peer systems offer many advantages [9], by allowing to access distributed information without specifying the server containing it. However, the principle of propagation has some drawbacks [6]. A first problem is that the definition of a neighbor of a node is done randomly and does not rely on any particular semantics. Another problem is that propagation generates an important number of messages through the network [10].

In this paper, we propose a solution to reduce the number of messages used during the query propagation, as well as the response time [16]. We consider that a user looking for specific data will first query metadata [11][3], which give the user a description of the available resources. Thus, queries we consider here only concern metadata. They are used as a first step to discover sites storing relevant data. The idea used in our approach is based on using some knowledge to adjust the rebounds. Different knowledge levels can be used: user knowledge (profile, experience, history of past queries, belonging to a community of users having the same kinds of interests, etc.) as well as network knowledge (based on the semantics of the node content [4][5]).

We focus in this paper on the user knowledge, and particularly on his/her belonging to a community of users. We claim that a user interested by meteorology will find relevant data on the servers already and successfully queried by users of the same community (for instance meteorologists) or by users of related communities (for instance climatology). Our

¹ This research is done in the context of the PADOUE project (<http://www-poleia.lip6.fr/padoue>) financed by ACI GRID : <http://www-sop.inria.fr/aci/grid/public>

purpose is to introduce this kind of knowledge into P2P systems, in order to direct as better as possible the search, i.e. to propagate the query directly to a relevant node (i.e. containing relevant data). As P2P systems are very evolutive, we insist in this paper on the dynamicity of our approach.

This approach is inspired by works about collaborative filtering [14][2][15]. In our solution, we do not filter on users, as it is generally the case in other related works, but rather on the community, which makes our work original. Our goal is to discover related communities, which will recommend sites containing relevant information. For this purpose, we label the nodes in order to describe the domain of the data they store. This is similar to the notion of semantic labeling used in Semantic Overlay Network [5] where nodes are logically linked according to their contents. But this approach requires a global hierarchy of concepts to classify all items (data, query, nodes, and connections), while our system only uses information on membership of a community and users feedbacks, which can easily be obtained.

The paper is organized as follows. After defining the notion of community in Section 2, we explain in Section 3 how to model it in order to introduce it into P2P systems. We show how to use the notion of links between communities (several communities have similarities, and can share interests) for query propagation in Section 4. We conclude in Section 5, and present some perspectives for this work.

2 Community

We suppose that users are clustered around the nodes of the network, according to their physical localization. For example, a node gathers a set of users belonging to the same organism, or the same department, etc. The notion of community we introduce here, allows to cluster again users linked by the same node, according to more logical features. Users belong to the same community if they are bound to the same node (as we previously mentioned) and if they share the same field (or related fields) of interest. To build the links between users, a community is defined by a small set of themes (ie: keywords) which characterize a field of interest. Figure 1 shows examples of such themes concerning specific domains of sciences such as oceanography, oceanology, hydrology, etc which define the fields of interest of the communities.

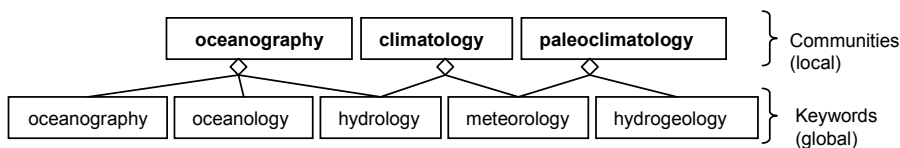


Fig 1 : Definition of communities according to themes (keywords). In this example, the community “climatology” is defined for a node by the two keywords “hydrology” and “meteorology”. We note that the set {hydrology, oceanology} may define the community “oceanography” on another node.

We underline the difference between the communities which are local for each node and the use of keywords to define communities. In fact, these themes, established by human experts, are a global resource shared by all nodes and known by all users of the network. This gives the node the autonomy to build its own community and avoids imposing a global definition. We also note that all community definitions are built on a single space of topics, allowing the system to compare them and to create links between them.

3 Source of community information

In order to better exploit community experiences, two different kinds of logical links between nodes can be built. On the one hand, we consider *relevance-based links* pointing out potentially interesting nodes for the query execution, and on the other hand, *inter-community links* mapping two similar communities on two different nodes. We present these two kinds of links in the following.

3.1 Pattern of relevance-based links

We first consider a set of users which membership of a community of interest is clearly established and easy collectable to avoid discovering it by an elaborate process of users' clustering [13]. For example, a set of researchers in environment may be clustered into several communities: hydrogeologists, climatologists, ecologists, etc. Each user is free to specify his/her community among the list of communities defined on the node (see Figure 2-a) (if there is no relevant community, a user can create a new community).

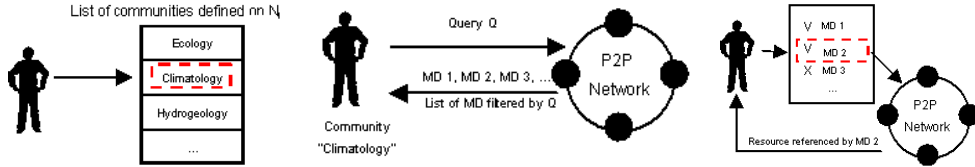


Fig 2.a: A user on node N_i specifies his community among the list of available communities

Fig 2.b: The user queries the network and the system returns the metadata potentially relevant for the query

Fig 2.c: The user gives his feedback about the returned metadata, and selects the corresponding relevant resources

Moreover, to simplify the process of resource retrieval, we use metadata [7] which describe the resources (data or programs) shared in the network. These metadata allow the user to straightly and quickly build his/her own opinion about the relevance of the resource. Thus, the process of resource retrieval is now treated as a process of metadata localization as shown on Fig 2.b. When resources are discovered, the user gives some feedback (1 if it is relevant, -1 if it is not relevant) on these metadata and selects, among the set of metadata he/she obtained, the relevant resources to load (see Fig 2.c).

For example, a resource as a “pluviometric reading”, which contains a set of statement, is described by a metadata defined by a title, a date, an author, a localization, ... In this way, users can build structured queries by keywords (for example: “title: pluviometric reading, date:2003, localization: France”) to filter relevant metadata.

In the following, we suppose that resources stored on a same node are related to same topics². We thus consider that a node is relevant, if relevant resources (i.e. metadata) have been found on it.

Given C_{ij} , the i^{th} community *defined locally* on the node N_j , we can now define the relevance-based links allowing to come out community knowledge. Such semantic link is defined by the following triple (1), stored on node N_j to express that the members of the community C_{ij} , consider the node N_q is relevant, with a given aggregated feedback fa_{ij}^q .

$$(N_q, C_{ij}, fa_{ij}^q) \quad (1)$$

² This hypothesis is realist in our context where each node is a specific environmental organism which shares its own resources on specific themes.

where in (1), N_q is the IP address of the node N_q , C_{ij} is the label of the community C_{ij} . We specify that $fa_{ij}^q = \text{agg}(f_{u_1}^q, \dots, f_{u_n}^q)$ is the feedback achieved by the function agg of aggregation (average) based on individual feedbacks $f_{u_1}^q, \dots, f_{u_n}^q$ awarded by some users of the community C_{ij} about node N_q . (see the right side of figure 3). A community can thus maintain a list of relevant nodes, which are recommended to users belonging to this community. In the same way, this knowledge can be used to avoid querying a node if the community considers it as irrelevant (when the aggregated feedback is negative).

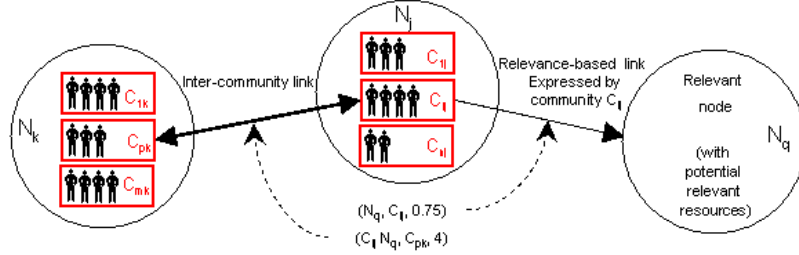


Fig 3 : Different links between nodes. The relevance-based link between N_j and N_q expresses that N_q is relevant for the community C_{ij} of N_j . The inter-community link expresses the similarity between communities C_{ij} and C_{pk} on nodes N_j and respectively N_k .

3.2 Pattern for inter-community links

Considering only the experiences of a local community is not satisfying. For this reason, we propose to take advantage of experiences of other communities defined on other neighbor nodes. Our idea is to build a link between two nodes on which similar communities exist. Before explaining what similar communities means, we underline the fact that the purpose of these links is not the straight localization of relevant nodes, but only the localization of communities competent to recommend relevant nodes. Our approach of inter-community links gets in two steps: the static creation and the dynamic evolution.

In a first step, we temporarily define the inter-community links between community C_{ij} defined on node N_j and community C_{pk} defined on node N_k , by a triple (2) stored on node N_j :

$$(C_{ij}, N_k, C_{pk}) \quad (2)$$

where in (2) C_{ij} and C_{pk} are the names of the communities (i.e. themes which specify them) and N_k is the IP address of the node N_k . To create these links, we introduce a definition-based similarity which allows to compare the themes defining the communities. In order to define a similar community for the community C , we compare the themes defining C with the sets of themes defining the other communities on a given node. The community having the highest number of common themes with C is considered as similar to C .

Given $D(C)$ which returns the set of themes defining the community C , we say that:

$$C_{ij} \text{ is } \textit{def-similar} \text{ to } C_{pk} \text{ if: } \circ D(C_{ij}) \cap D(C_{pk}) \neq \emptyset \quad \text{and,} \quad (3)$$

$$\circ C_{pk} = \arg \max_{c \in \{\text{communities of } N_k\}} |D(C_{ij}) \cap D(c)| \quad (4)$$

In other words, we search on N_k a community having at least one common keyword (condition 3) and having the highest number of common themes (condition 4). For example, let us consider the community “climatology” (considered as C_{ij}) defined by $D(\text{climatology}) = \{\text{hydrology, meteorology}\}$ on N_j . We consider only communities on N_k having at least the keyword “hydrology” or “meteorology” in their definition. The

communities having these two keywords in their definition are considered as *def-similar*. We underline that the *def-similar* definition depends on the original choice of themes. Moreover, these links can be established at the creation of a community or when two nodes become neighbors.

We have shown how to relate communities by two different kinds of links, the relevance-based link and the inter-community link. Figure 3 points out the difference between these two kinds of links. However, this mapping between communities is only static, and does not reflect the evolution of communities. Indeed, we have shown that the knowledge of a community comes out from user feedbacks, which is a dynamic knowledge. We explain in the following section how to take into account the dynamic evolution of communities into the handling of inter-community links.

3.3 Handling inter-community links

As experiences of each community evolve, the static inter-community links become obsolete. Their relevance is called into question after being modified many times (consideration of new individual feedbacks). Thus, the use of *def-similar* is not adapted to this evolution. Therefore, we introduce a new definition of the similarity based on the experiences of the community.

As we want to compare dynamically communities, we consider the Pearson correlation coefficient [2]. Indeed, to establish the correlation between C_{ij} and the communities of N_k , node N_j sends to node N_k the experiences of the community C_{ij} (i.e. the list of evaluated nodes and their aggregated feedback) to compute all the correlations. The correlation between C_{ij} and a community C_{xk} of N_k is defined by the following formula:

$$\omega(C_{ij}, C_{xk}) = \frac{\sum_{h/N_h \in NC} (\bar{f}_{a_{ij}}^h - \bar{f}_{a_{ij}}) (\bar{f}_{a_{xk}}^h - \bar{f}_{a_{xk}})}{\sqrt{\sum_{h/N_h \in NC} (\bar{f}_{a_{ij}}^h - \bar{f}_{a_{ij}})^2} \sqrt{\sum_{h/N_h \in NC} (\bar{f}_{a_{xk}}^h - \bar{f}_{a_{xk}})^2}} \quad (6)$$

where NC is the set of common nodes evaluated by the two communities, and $\bar{f}_{a_{ij}}$ is the average of all aggregated feedbacks provided by the community C_{ij} .

Given $Ne(C)$, which returns the set of nodes evaluated by the community C , the use of the correlation between communities allows us to define the experience-based similarity as follows :

$$C_{ij} \text{ is } \textit{exp-similar} \text{ to } C_{pk} \\ \text{if : } \begin{cases} \circ |Ne(C_{ij}) \cap Ne(C_{pk})| > \delta \\ \circ C_{pk} = \arg \max_{c \in \{communities \text{ of } N_k\}} \omega(C_{ij}, c) \end{cases} \quad , \text{ and} \quad (7)$$

$$\circ C_{pk} = \arg \max_{c \in \{communities \text{ of } N_k\}} \omega(C_{ij}, c) \quad (8)$$

Condition (7) allows considering this similarity only if there are enough common evaluated nodes between C_{ij} and C_{pk} . If the comparison is well-founded, we update the inter-community link with the community specified by (8). Thus, we keep the links created according to the definition of the community until the set of common evaluated nodes is under a threshold δ .

However, calculating *exp-similarity* again for each new feedback is not interesting, because the number of computations used for this task is not justified to express the light evolution of the community knowledge. We introduce in our definition of inter-community link a parameter of freshness. An inter-community link between the community C_{ij} on N_j and C_{pk} on N_k with a freshness t , is defined by the quadruplet (9), stored on N_j :

$$(C_{ij}, N_k, C_{pk}, t) \quad (9)$$

where $t \in \{1, \dots, \theta\}$, with θ being a parameter used as initial value of freshness fixed in such a way that the system does not continuously cast doubt on the relevance of the links.

To sum up, as long as a community C does not have any experiences, the inter-community similarity with C is based on *def-similarity*; otherwise, the system periodically establishes the community the closest of C with the *exp-similarity* according to the evaluated nodes and their aggregated feedback.

4 Query propagation and exploitation of community information

Now that the notions of static and dynamic links between communities are defined, we focus on how to exploit all these information stemming from a community. In particular, we will explain to which nodes a query is propagated and why. According to Figure 4 and given the user U belonging to community C_{ij} on the node N_j , the query Q is handled by node N_j .

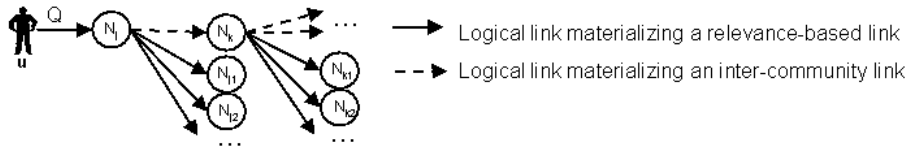


Fig 4: Illustration of query propagation

In a first step, the system returns the nodes N_{j1} , N_{j2} , etc, specified by some relevance-based links of C_{ij} i.e. nodes obtained by filtering the stored triplets (1) on N_j having a positive aggregated feedback. Then, node N_j sends the query Q to these nodes supposed to store relevant resources. Moreover, the inter-community links stored on N_j specify that there are a community C_{pk} on N_k related to C_{ij} . The query Q is sent on N_k and will not be executed on it, but on the nodes that the community C_{pk} considers relevant (N_{k1} , N_{k2} , ... obtained by relevance-based link of the community C_{pk}). According to the principle of peer-to-peer, the node N_k treats Q as its proper query. Therefore, Q will be straightly propagated towards relevant nodes, in order to find relevant resources.

5 Conclusion and future work

Information retrieval in large distributed systems introduces new challenges for database technology. Peer-to-peer architectures offer interesting solutions to support scalability. However, they must be adapted to increase their functionalities and their performance. We propose in this paper to improve the performance of peer-to-peer systems by introducing knowledge in the process of query propagation. This knowledge is used to select the nodes to which the query will be redirected. The kind of knowledge we introduce here is the belonging of a user to a community, and the links which can be established between related communities. This knowledge is both static and dynamic. We have shown how to define and how to use these links to direct query propagation.

This work has been done in the framework of the PADOUE project [12], which aims at building a complete system for sharing heterogeneous and distributed environmental information and in which we exploit the experience of scientific communities. Further work is still to be done. We are currently validating our proposition by integrating this knowledge into the route tables of a peer-to-peer system. An important perspective is to consider other kinds of knowledge. In this work, we focused on community knowledge, but we intend to take into account other knowledge levels, such as user profile, or network knowledge. Our purpose consists of exploiting complementary sources of semantic to smartly extract information from a peer-to-peer system.

References:

1. Aberer, K., Hauswirth, M., Peer-to-Peer Information Systems: Concepts and Models, state-of-the-art, and Future Systems, Tutorial IEEE ICDE, 2002.
2. Breese, J.S., Heckerman, D., Kadie, C., Empirical analysis of predictive algorithms for collaborative filtering. In Proc. of the 14th conference on uncertainty in artificial intelligence, pp. 43-52, Madison, Wisconsin, July 1998.
3. Coulondre, S., Libourel, T., Spéry, L., Metadata and GIS : a classification of metadata for Gis, Gis Planet'98, International Conference and Exhibition on Geographic Information, Lisbon, Portugal, September 1998
4. Crespo, A., Garcia-Molina, H., Routing Indices for Peer-to-Peer Systems, In Proc. of the 22th International Conference on Distributed Computing Systems (ICDCS) Vienna, Austria, 2002
5. Crespo, A., Garcia-Molina, H., Semantic Overlay Networks for P2P Systems, submitted for publication, (<http://www-db.stanford.edu/peers>) 2003
6. Daswani, N., Garcia-Molina, B., Yang, B., Open Problems in Data-Sharing Peer-to-Peer Systems, In Proc. of the 9th International Conference on Database Theory (ICDT), Siena, Italy, 2003
7. Galhardas, H., Simon, E., Tomasic, A., A framework for classifying scientific metadata, AAAI Workshop on AI and Information integration, Madison, Wisconsin, August 1998.
8. Golberg, K., Roeder, T., Gupta, D., Perkins, C., Eigentaste: A Constant Time Collaborative Filtering Algorithm, Technical report IOER and EECS Departments, University of California, Berkeley, August 2000.
9. Gribble, S., Halevy, A., Ives, Z., Rodrig, M., Suci, D., What Can Peer-to-Peer Do for Databases, and Vice Versa?, In Proc. of the 4th International Workshop on the Web and Databases (WebDB '2001), Santa Barbara, California, May 2001.
10. Jovanovic, M.A., and al, Scalability Issues in Large Peer-to-Peer Networks – A Case Study of Gnutella, Research report, Univ. Cincinnati, 2001
11. Moura, A., Perez, H., Tanaka, A., Metadata model for supporting data extraction from environmental information systems, In Proc of the International Conference On Geographic Information Science, Savannah, Georgia, October 2000
12. PADOUE Project (<http://www-poleia.lip6.fr/padoue>) multifield project of ACI-GRID: <http://www-sop.inria.fr/aci/grid/public>
13. Seng, S.H., Han, J., Wang, K., RecTree: An efficient collaborative filtering method, in Proc. The conference on data Warehouse and Knowledge Discovery, (DaWaK), Munich, Germany, 2001
14. Resnick, P., Varian, H.R., Recommender Systems. In Communications of the ACM , Vol 40. N°3, March 1997
15. Ungar, L., Foster, D., Clustering methods for collaborative filtering, In Workshop on Recommender systems at the 15th National Conference on Artificial Intelligence, Madison, Wisconsin, July 1998.
16. Yang, B., Garcia-Molina, H., Improving Search in Peer-to-Peer Systems, In Proc. of the 22th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, July 2002

OntoMiner: Bootstrapping and Populating Ontologies From Domain Specific Web Sites

Hasan Davulcu, Srinivas Vadrevu, and Saravanakumar Nagarajan

Department of Computer Science and Engineering,
Arizona State University,
Tempe, AZ, 85287, USA
{hdavulcu, svadrevu, nrsaravana}@asu.edu

Abstract. RDF/XML has been widely recognized as the standard for annotating online Web documents and for transforming the HTML Web to the so called Semantic Web. In order to enable widespread usability for the Semantic Web there is a need to bootstrap large, rich and up-to-date domain ontologies that organize most relevant concepts, their relationships and instances. In this paper, we present automated techniques for bootstrapping and populating specialized domain ontologies by organizing and mining a set of relevant Web sites provided by the user. We develop algorithms that detect and utilize HTML regularities in the Web documents to turn them into hierarchical semantic structures encoded as XML. Next, we present tree-mining algorithms that identify key domain concepts and their taxonomical relationships. We also extract semi-structured concept instances annotated with their labels whenever they are available. Experimental evaluation for the News and Hotels domain indicates that our algorithms can bootstrap and populate domain specific ontologies with high precision and recall.

1 Introduction

RDF and XML has been widely recognized as the standard for annotating online Web documents and for transforming the HTML Web to the so called Semantic Web. Several researchers have recently questioned whether participation in the Semantic Web is too difficult for “ordinary” people [1–3]. In order to enable widespread usability for the Semantic Web there is a need to bootstrap large, rich and up-to-date domain ontologies that organizes most relevant concepts, their relationships and instances. In this paper, we present automated techniques for bootstrapping and populating specialized domain ontologies by organizing and mining a set of relevant taxonomy-directed Web sites provided by the user. A Web site is said to be “taxonomy-directed” if it contains at least one taxonomy for organizing its contents and it presents the instances belonging to a concept in a regular fashion. Notice that, neither the presentation of the taxonomy among different pages, nor the presentation of instances among for different concepts need to be regular for a Web site to be classified as “taxonomy-directed”. Almost all scientific, news, financial, travel, shopping and search/community portals that

we are aware of are indeed “taxonomy-directed”. As an example application, a user of the OntoMiner can use the system to rapidly bootstrap and ontology populated with instances and they can tidy-up the bootstrapped ontology to create a rich set of labelled examples that can be utilized by supervised machine learning systems such as the WebKB[4].

The user of the OntoMiner system only need to provide the system the URLs of the Home Pages of 10 to 15 domain specific Web sites that characterizes her domain of interest. Next, OntoMiner system detects and utilizes the HTML regularities in Web documents and turns them into hierarchical semantic structures encoded as XML by utilizing a hierarchical partition algorithm. We present tree-mining algorithms that identifies most important key domain concepts selected from within the directories of the Home Pages. OntoMiner proceeds with expanding the mined concept taxonomy with sub-concepts by selectively crawling through the links corresponding to key concepts. OntoMiner also has algorithms that can identify the logical regions within Web documents that contains links to instance pages. OntoMiner can accurately separate the “human-oriented decoration” such as navigational panels and advertisement bars from real data instances and it utilizes the inferred hierarchical partition corresponding to instance pages to accurately collect the semi-structured concept instances.

A key characteristic of OntoMiner is that, unlike the systems described in [5, 6] it does not make any assumptions about the usage patterns of the HTML tags within the Web pages. Also, OntoMiner can separate the data instances from the data labels within the vicinity of extracted data and attempts to accurately annotate the extracted data by using the labels whenever they are available. We do not provide algorithms for extracting and labelling data from within HTML tables since there are existing solutions for detecting and wrapping these structures [7, 8].

Other related work includes schema learning[9–11] for semi-structured data and techniques for finding frequent substructures from hierarchical semi-structured data[12, 13] which can be utilized to train structure based classifiers to help merge and map between similar concepts of the bootstrapped ontologies and better integrate their instances.

The rest of the paper is organized as follows. Section 2 outlines the hierarchical partitioning, Section 3 discusses taxonomy mining, and Section 4 describes instance mining. Experimental evaluation for the News and Hotels domains indicates that our algorithm can bootstrap and populate domain specific ontologies with high precision and recall.

2 Semantic Partitioning

2.1 Flat Partitioner

Flat Partitioner detects various logical partitions of a Web page. For example, for the home page of <http://www.nytimes.com>, the logical partitions are marked in boxes B1 through B5 in Figure 1. The boxes in snapshot of Web page in Figure 1 correspond to the dotted lines shown in tree view of Web page in Figure 1..

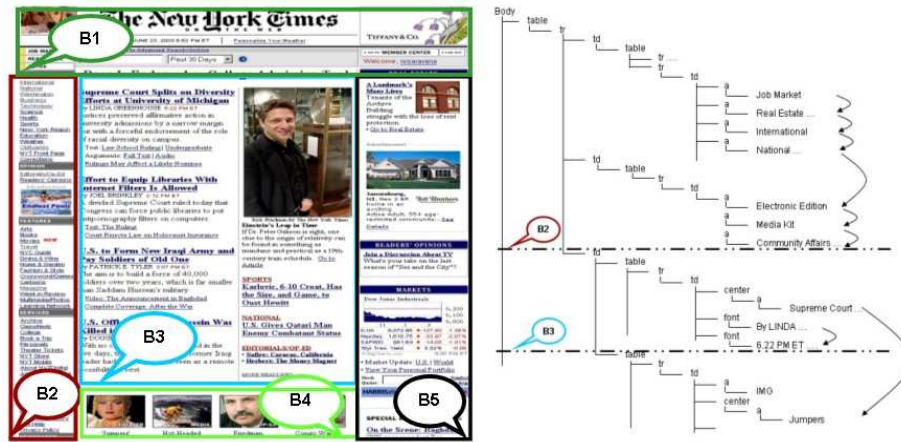


Fig. 1. Snapshot of New York Times Home Page and Parse Tree View of the Home Page

The Flat Partitioner Algorithm takes an ordered DOM tree of the Web page as input and finds the flat partitions in it. Intuitively, it groups contiguous similar structures in the Web pages into partitions by detecting a high concentration of neighboring repeated nodes, with similar root-to-leaf tag-paths. First, the partition boundary is initialized to be the first leaf node in the DOM tree. Next, any two leaf nodes in the tree are linked together with a "similarity link" if they share the same path from the root of the tree and all the leaf nodes in between have different paths. Then the ratio of number of "similarity links" that crosses the current candidate boundary to the total number of "similarity links" inside the current partition is calculated. If this ratio is less than a threshold δ , the current node is marked as the partition boundary. Otherwise, current node is added to the current partition and the next node is considered as the partition boundary. The above process terminates when the last element in the list of leaf nodes is reached. A Path Index Tree (PIT) is built from the DOM tree of the Web page, which helps to determine all the "similarity links" between the leaf nodes within a single traversal. The PIT is a tree based data structure which is made up of all unique root to leaf tag-paths and, in its leaf nodes PIT stores the "similarity links" between the leaf nodes of the DOM tree.

The tree view in Figure 1 illustrates the Flat Partitioning Algorithm. The arrows in the tree view in Figure 1 denote the "similarity links" between the leaf nodes. Let's assume the threshold δ is set to 60%. Then, when the current node is "Job Market" the total number of outgoing unique "similarity links" (out in line 9) is 1 and total number of unique "similarity links" (total in line 10) is 1. Hence the ratio of out to total is 100% which is greater than threshold. Hence current in line 6 becomes the next leaf node. At node "International", out becomes 1 and total is also 1. Hence the ratio is still greater than threshold.

Algorithm 1 Flat Partition Algorithm*Flat Partitioner**Input: T: DOM Tree**Output: $\langle b_1, b_2, \dots, b_k \rangle$: Flat Boundaries*

```

1: PIT := PathIndexTree(T)
2: current := first leaf node of T
3: Partition_Nodes :=  $\phi$ 
4: Partition_Boundaries :=  $\phi$ 
5: for each lNode in Leaf_Nodes(T) do
6:   current := lNode  $\rightarrow$  next
7:   if  $N = \text{PIT.next\_similar}(\text{Current})$  exists then
8:     Partition_Nodes := Partition_Nodes  $\cup$  N
9:   end if
10:  out =  $|\{\text{path}(m) | m \in \text{Partition\_Nodes and } m > \text{current}\}|$ 
11:  total =  $|\{\text{path}(m) | m \in \text{Partition\_Nodes}\}|$ 
12:  if out/total  $\leq \delta$  then
13:    Partition_Boundaries := Partition_Boundaries  $\cup$  current
14:    Partition_Nodes :=  $\phi$ 
15:  end if
16: end for
17: Return Partition_Boundaries

```

When *current* reaches "Community Affairs", *out* becomes 0 whereas *total* is 1 and hence the ratio is less than threshold δ . Now, "Community Affairs" (*B2* in Figure 1) is added to the set of *partition_boundaries* in line 12 and all the "similarity links" are removed from the *partition_nodes* in line 13. The same boundary detection condition is satisfied once again when the algorithm reaches "6.22 PM ET" where *out* becomes 1 and *total* is 3. Hence "6.22 PM ET" (*B3* in Figure 1) is added to the *partition_boundaries*.

2.2 Hierarchical Partitioning

Hierarchical Partitioner infers the hierarchical relationships among the leaf nodes of the HTML parse tree where all the page content is stored. The Hierarchical Partitioner achieves this through sequence of three operations: Binary Semantic Partitioning, Grouping and Promotion.

Binary Semantic Partitioning The Binary Semantic Partitioning of the Web page relies on a dynamic programming algorithm which employs the following cost function. The dynamic programming algorithm determines the nodes that need to be grouped together, by finding the grouping with the minimal cost. The cost for grouping any two nodes in the HTML parse tree is recursively defined as follows.

$$Cost(L_i, L_j) = \begin{cases} 0 & \text{if } i=j \\ \min_{i \leq k < j} \{Cost(L_i, L_k) + Cost(L_{k+1}, L_j) \\ + Grouping_Cost(L_{i\dots k}, L_{k+1\dots j})\} & \text{if } i < j \end{cases}$$

where L_i, L_j are two leaf nodes in the HTML parse tree.

The cost function calculates the measure of dissimilarity between two nodes i.e. a high value of cost indicates that these two nodes are highly dissimilar. Hence the dynamic programming algorithm finds the lowest cost among the various possible binary groupings of nodes and parenthesizes them into a binary-tree. The cost for grouping two consecutive sub trees is calculated as the sum of four sub-cost factors. Let A, B be the least common ancestor of nodes L_i to L_k and L_{k+1} to L_j respectively. Then,

$$\begin{aligned} & Grouping_Cost(A, B) = \\ & \text{Sum of distances of A and B to their LCA, } C_{LCA}(A, B) + \\ & \text{Similarity of the paths from A and B to their LCA, } C_{PSIM}(A, B) + \\ & \text{Similarity of the paths in the sub trees of A and B, } C_{STSIM}(A, B) + \\ & \text{Order similarity of the paths in the sub trees of A and B, } C_{ORD}(A, B) \end{aligned}$$

The first cost factor $C_{LCA}(A, B)$ calculates how far the two nodes are apart from their least common ancestor. The cost for similarity between paths to the least common ancestor is determined by the second cost factor $C_{PSIM}(A, B)$. The third $C_{STSIM}(A, B)$ and fourth $C_{ORD}(A, B)$ cost factors computes the cost for similarity in the sub trees of the two nodes, former computes the similarity in the paths whereas the later computes the ordering of paths in the sub tree.

Let S_1 be the set of all paths in the sub tree of A, S_2 be the set of all paths in the sub tree of B, d_1 be the number of tags on the path from LCA to A, d_2 be the number of tags on the path from LCA to B and max depth be the maximum depth of the DOM tree.

$$C_{LCA}(A, B) = \sqrt{\frac{d_1 + d_2}{2 * max_depth}}$$

$$C_{PSIM}(A, B) = 1 - \frac{\text{Similarity between Paths } P_1 \text{ and } P_2}{\max(d_1, d_2)}$$

$$C_{STSIM}(A, B) = 1 - \max(\text{Separation}, \text{Overlap}),$$

$$\text{where Separation} = \frac{|(S_1 - S_2) \cup (S_2 - S_1)|}{|S_1 \cup S_2|} \text{ and Overlap} = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

$$C_{ORD}(A, B) = 1 - Sim(A, B),$$

$$\text{where } Sim(A, B) = \frac{\text{Number of Paths similar in order in Sub Trees of A and B}}{\text{Max of No of Paths in Sub Trees of A and B}}$$

For example, let a / b / c be the three tags on the path from LCA to A and a / b / d be the tags on the path from LCA to B. Let P_1, P_2, P_3 be the set of

paths in the sub tree of A and P_1, P_2, P_4 be the set of paths in the sub tree of B.

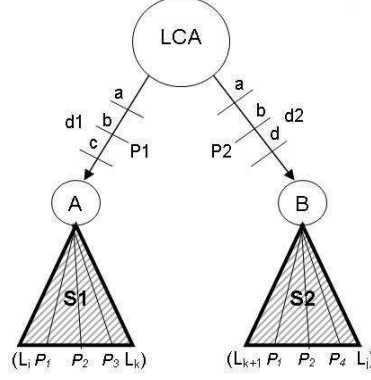


Fig. 2. Sample Tree

$$d_1 = |a, b, c| = 3, d_2 = |a, b, d| = 3$$

$$S_1 = P_1, P_2, P_3, S_2 = P_1, P_2, P_4$$

$$C_{LCA}(A, B) = \sqrt{\frac{3+3}{2 * max_depth}}$$

$$C_{PSIM}(A, B) = 1 - \frac{|\{a, b, c\} \cap \{a, b, d\}|}{max(d_1, d_2)} = \frac{1}{3}$$

$$Separation = \frac{|\{P_3\} \cup \{P_4\}|}{|\{P_1, P_2, P_3, P_4\}|} = \frac{1}{2}$$

$$Overlap = \frac{|\{P_1, P_2\}|}{|\{P_1, P_2, P_3, P_4\}|} = \frac{1}{2}$$

$$C_{STSIM}(A, B) = 1 - max(Separation, Overlap) = \frac{1}{2}$$

$$Sim(A, B) = \frac{|\{P_1, P_2\}|}{max(|S_1|, |S_2|)} = \frac{1}{2}$$

$$C_{ORD}(A, B) = 1 - Sim(A, B) = \frac{1}{3}$$

These cost functions are adjusted to fit for different cases in the HTML Parse Tree. The three different cases that may arise during the cost function evaluation are shown in Figure 2.

- Case 1: LCA of the two nodes which are to be grouped in one partition is one of the nodes itself and the other node is not a leaf node.
- Case 2: LCA of the two nodes which are to be grouped in one partition is one of the nodes itself and the other node is a leaf node.
- Case 3: the LCA nodes for the ranges are identical.

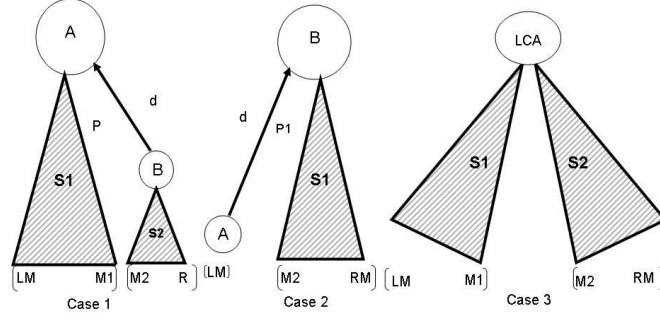


Fig. 3. Different Cases during Cost Function Evaluation

In all the three cases, the second and fourth cost factors are irrelevant and hence they are ignored. For Case 3, first cost factor is also ignored. Accordingly, the first and third cost factors are modified as follows. For Case 1,

$$C_{LCA} = \frac{d}{maxdepth} \quad C_{STSIM} = 1 - max(Separation, Overlap)$$

For Case 2,

$$C_{LCA} = \frac{d}{maxdepth}, \quad C_{STSIM} = 1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \text{ where } S_2 \text{ is } \{P_1\}$$

For Case 3,

$$C_{SIM}^{ST} = 1 - max(Separation, Overlap)$$

The total cost is divided by the number of applicable cost factors to normalize the cost to a value between 0 and 1. The above dynamic programming algorithm takes the DOM tree as input and produces semantic binary-tree partitions of its leaf nodes. The Column 1 of the Figure 4 represents the DOM tree of the HTML page and Column 2 represents the binary Partition Tree. For example the nodes 68 through 82 are grouped into one partition which has internal binary partitions as shown in Figure 3.

Grouping The next step in the Hierarchical Partitioning is grouping of similar binary partitions into group nodes. Grouping Algorithm first finds pairs of

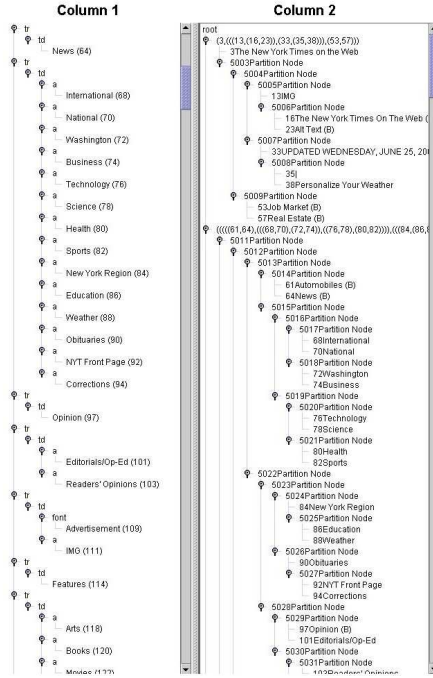


Fig. 4. Dynamic Programming

partitions which are similar by post order traversal of semantic binary partition tree. Intuitively the grouping step creates "Group" nodes made up of "Instance" nodes as its children. The Instances are identified during the post order traversal of the semantic binary partition tree whenever the "similarity" between a right sibling and its left sibling is above a certain threshold δ . The "similarity" between siblings is based on the Grouping_Cost explained in Binary Semantic Partition section. Then, the parent of the "Instance" nodes is marked as "Group" node. During the rest of the post order traversal the similarity between an internal node and a "Group" node is calculated by evaluating the similarity between the unmarked node and the first Instance of the "Group" node.

The Grouping Algorithm first initializes the type of the leaf nodes in the binary partition tree as "simple". While traversing the tree, if it finds two "simple" sibling nodes and if the cost for grouping these two nodes is less than a threshold δ , then it marks these nodes as "Instances" and their parent as "Group" node. For example, in Figure 5, nodes "Sports" and "Health" are sibling nodes and the cost for grouping these two nodes is also less than the threshold δ . Hence both are marked as "Instance" node and their parent is marked as "Group" node. Similarly, if it finds two sibling nodes that are marked as "Group" and if the cost for grouping their instances is less than threshold δ , then it marks the parent of these sibling nodes as "Group". For example, the parent of nodes

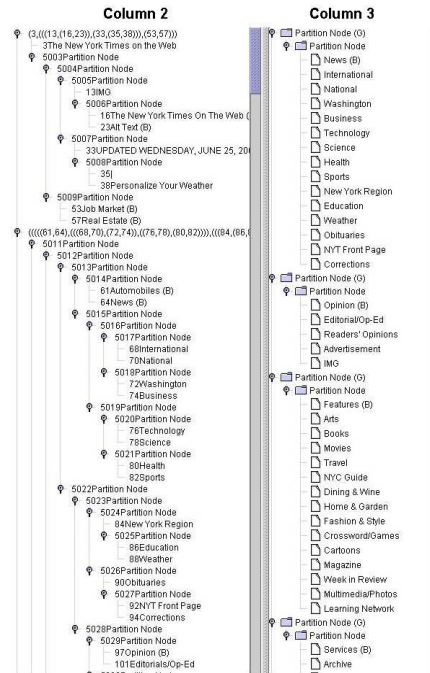


Fig. 5. Grouping

”Health” and ”Sports” is already marked as ”Group” node. Similarly, the parent of ”Science” and ”Technology” is also marked as ”Group” nodes. Then, if the cost for grouping all the instances ”Technology” through ”Sports” is also less than threshold δ , then grand parent of these instances is marked as ”Group” node and their instances are merged as seen in Column 3 of Figure 4. Next, if one of the sibling nodes is ”simple” and the other node is ”Group” and the cost for grouping the ”simple” node with the instances of the group node is also less than threshold δ , then it changes the type of ”simple” node to ”Instance” and marks their parent as ”Group” and merges the ”Instance” node with instances of the ”Group” node. This operation is continued until the root of the binary Partition tree is reached and all markings are done. Figure 4 shows the conversion of binary partition tree to Group Tree. The Column 2 and Column 3 represent the binary partition and Group trees.

Promotion After Grouping, the final step in Hierarchical Partitioning is promotion. The promotion algorithm identifies the leaf nodes that should be promoted above their siblings. A node is considered for promotion only if it applies to the following rules.

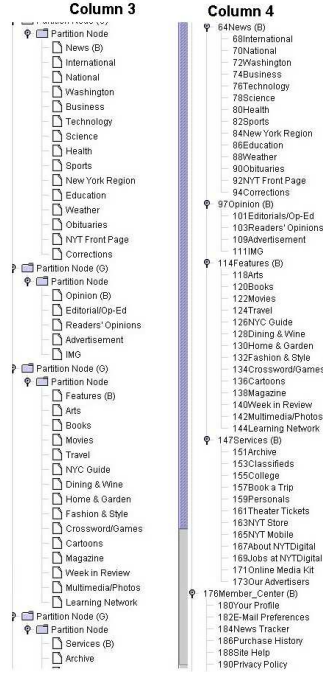


Fig. 6. Promotion

- **Rule 1:** A node can be marked as BOLD and it is the first child of its parent and the parent is not marked as "Group" node. A node is marked as BOLD, if it satisfies any of the following conditions.
 1. If there is a bold tag like ``, `<bold>` etc. on its path from the root node
 2. If any bold tag is defined for the "class" attribute value of this node.
 3. If the labelled text of the node is fully capitalized.
- **Rule 2:** A node can be promoted if it is the first child of its parent and its parent is not marked as "Group" node and the only other sibling to this node is a "Group" node.

The nodes which satisfy the bold conditions are marked as BOLD nodes (indicated by letter (B) in column 3). The BOLD node replaces its parent "Partition Node". If the promotion rules can be applied again, the BOLD node is promoted once more. Figure 6 illustrates the Promotion Algorithm. Column 3 represents the Group Tree and Column 4 represents the Hierarchical Partition Tree. The Node "News" is marked as BOLD and it is the first child of its parent as shown in Column 3 of Figure 6. Hence it is promoted on top of all the nodes "International" through "Corrections". Similarly the nodes "Opinion", "Features" and "Services" are also promoted.

Experimental Results In order to calculate precision and recall for the generated Hierarchical Partitioning Trees, Ideal Hierarchical Semantic Partitioned Trees are manually generated for every page. Next, transitive closure of all parent-child relationships implied by each tree is generated. The Precision and Recall are calculated as follows.

The algorithm is applied for home pages of the following 13 Websites and the experimental results are shown in the Table 1.

Domain	Precision	Recall
www.abcnews.go.com	0.79	0.86
www.ninemsn.com	0.75	0.72
www.cbc.ca	0.94	0.73
www.cnn.com	0.81	0.75
www.foxnews.com	0.83	0.82
www.msnbc.com	0.80	0.86
www.nytimes.com	0.83	0.88
www.time.com	0.80	0.86
www.timesonline.co.uk	0.84	0.92
www.un.org	0.79	0.86
www.usnews.com	0.85	0.64
www.washingtonpost.com	0.77	0.71
www.washingtontimes.com	0.75	0.73
Average	0.81	0.80

Table 1. Precision and Recall for Hierarchical Partitioner

3 Taxonomy Mining

The taxonomy mining involves several tasks including separating important concepts (the categories that define the context) from instances (the content that are members of each concept), identification of similar concepts, and mining relationships among the concepts. Our goal is to automatically mine the taxonomy for a domain given relevant web pages from the domain. To demonstrate the efficacy of our algorithms we implemented and tested our approach with two separate domains; News Web pages and Hotel Web pages. The key ideas in taxonomy mining are illustrated in Figure 7. Various phases involved in taxonomy mining are explained in the following subsections.

3.1 Frequency Based Mining

The inputs to our system are the Home Pages of the co-domain Websites. We first preprocess the HTML documents using “Hierarchical Partitioning” (as described in Section 2) to generate semi-structured XML documents. We use these XML documents to mine the taxonomy. We exploit the observation that important concepts in a given domain are often frequent. Using this observation,

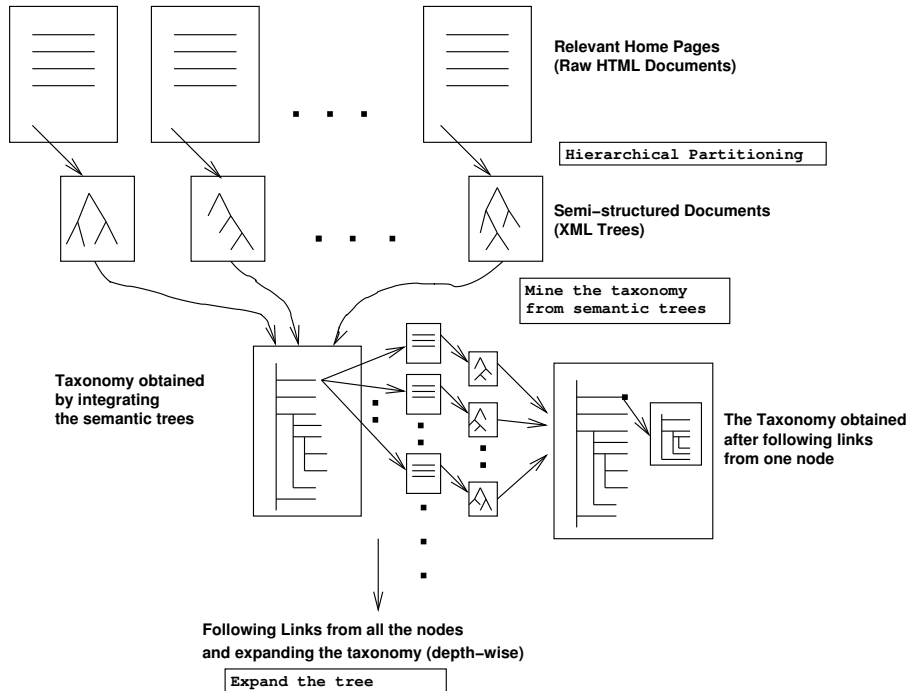


Fig. 7. Main Idea of the algorithm

our system mines frequent labels in the input XML documents among Home Pages of co-domain Websites. By using an experimentally determined threshold for support (the minimum number of times a label should occur in order to be frequent), we separate concepts from the rest of the document. For example in News domain, our system identifies Business, Sports, Politics, Technology, Health, Entertainment, etc. as important concepts as they are frequent across various news home pages.

3.2 Candidate Label Extraction Phase

In our simple frequency based mining, our system may miss some labels that are relevant but are not frequent (not present in many Home Pages pages for the domain). For example in <http://www.washtimes.com/>, our system identified “Entertainment” to be a frequent label but it missed “Civil War”, “Culture, etc.”. To identify such relevant labels our system learns attributed tag paths to the labels in the obtained from frequency based mining and applies these paths on the corresponding regions of the Home Pages to retrieve more labels. An attributed tag path of a label has XPath syntax and it is made up of HTML tag names along its path from the root of the HTML tree to the label itself with attributes and their values. For example the

attribute tag path for “Entertainment” in `http://www.washtimes.com/` is `//HTML//BODY[@bgColor]//TABLE[@cellpadding=0 and @cellspacing=0 and border=0 and @width=760]//TR//TD[@width=140 and @rowspan=8 and @valign=top]//TABLE[@cellpadding=0 and @cellspacing=0 and @border=0 and @width=140]//TR//TD[@height=20 and @class=twt-menu1a]//A//SPAN[@class=twt-mentext1]//#TEXT.`

3.3 Abridgment Phase

During the extraction phase, it is possible to identify some labels that are irrelevant to the domain (for example, “NYT Store” in `http://nytimes.com/`). To eliminate these irrelevant labels, we adopt the following rules.

- Eliminate a label if it does not have a URL or if the URL goes out of the domain.
- Eliminate a label if its URL does not have new frequent labels and valid instances (as described in 4).

3.4 Grouping the Labels into Concepts

From the above phases, we collect the important labels (keywords) from the relevant Home Pages. But the same label may appear different in various documents and this introduces redundancy. To accommodate for this, we group them according to their lexicographic similarity. First our system stems the labels using Porter Stemming Algorithm [14] and then it applies Jaccard’s Coefficient [15] (calculated as $\frac{|A \cap B|}{|A \cup B|}$, where A and B are stemmed vectors of words in two labels) on them to organize into groups of equivalent labels. We denote each such group of corresponding labels as a *concept*. This simple similarity measure is able to group the labels that are only lexicographically related (like “Sport” and “Sports”) but does not identify labels that are semantically related (like “World” and “International”). The issue of identifying and merging semantically related labels is beyond the scope of this paper and we plan to investigate it later.

3.5 Mining Parent-Child Relationships From Hierarchically Partitioned Web Pages

The concepts obtained from the grouping phase are flat (there are no is-a relationships among them). In order to organize the concepts in a taxonomy, we need ‘is-a’ relationships among the concepts. These relationships are mined from the hierarchically partitioned web pages generated from “Semantic Partitioning” (Section 2 using the algorithm described in Algorithm 2. After this phase, we have a taxonomy of concepts that represents the input home pages.

Algorithm 2 Mine is-a Relationships

*is-aMiner**Input: C: Set of Concepts, S: Set of Semantically Partitioned Web Pages, Sup: Support**Output: Tree representing the hierarchy of concepts*

- 1: Compute parent-child relationships R among set of concepts C along with their frequencies: $a - b$ is a direct parent-child relationship if a and b are concepts and b is immediate child of a
 - 2: Separate frequent relationships (those that satisfy the minimal support Sup), FR from the non-frequent ones, NFR .
 - 3: Identify the grand-parent relationships from NFR : for all non-frequent relationships $a - b$ and $b - c$, increment the count for $a - c$ in R .
 - 4: Populate frequent relationships FR from R again based on the support, Sup .
 - 5: Construct a tree, T from the relationships FR
 - 6: Return the tree T
-

3.6 Expanding the Taxonomy beyond Home Pages

The taxonomy obtained from the previous phase represents only the Home Pages in the domain. In order to expand the domain taxonomy, we follow the links corresponding to every concept c , find sub concepts and expand the taxonomy depth-wise and repeat the above phases (from Section 3.1 to 3.5) to identify sub concepts of c . For example, “Sports” is a concept in the taxonomy obtained from the previous phase. If we follow these links corresponding to “Sports” and repeat the above procedure we get taxonomy for “Sports”, that contains concepts like “Baseball”, “Tennis”, and “Horse Racing”. After this phase, we will have a complete taxonomy that represents the key concepts in the domain along with their taxonomical relationships. The entire Taxonomy Mining algorithm is detailed in Algorithm 3.

Experimental Results for Ontology Mining

The mined ontology is evaluated in the same way as explained in the experimental section for the semantic partitioning. An ideal ontology is manually created and the parent/child relationships for both the ontologies are determined. The precision for the mined ontology is 75% and the recall is 92%.

A fragment of the taxonomy that we mined from the news domain is shown in Figure 8.

4 Instance Extraction

This section describes our approach to extract instances from Web pages. Instances correspond to members of concepts. Our system can extract flat instances made up of list attribute name-value pairs as well as complex, semi-structured instances. Our system is able to extract the labels whenever they are available. We first present our approach to extract appropriate instance segments from HTML documents. Later we describe our instance extraction algorithm and conclude with discussion on experimental results.

Algorithm 3 Algorithm to Mine Hierarchy of Concepts From Home Pages

*TreeMiner**Inputs: N: The Root Node, H: Set of Pages, Sup: Support**Output: The Taxonomy of concepts*

- 1: Semantically Partition the input Pages to obtain semi-structured XML documents and add them to S
 - 2: Collect all the labels along with their URLs and their base URLs from all the XML documents (each label l has a text value, its URL and its base URL, the URL of the web page that it is present in)
 - 3: Frequency Based Mining: Separate frequent labels, L (frequent ones are those that satisfy the support, Sup) from the non-frequent ones.
 - 4: Label Extraction Phase: Learn the attribute paths to each label in L and apply these paths to the document in which it is present and get the candidate labels and add them to L
 - 5: **for** each element l in L **do**
 - 6: Remove l from L if it does not have a valid url
 - 7: Get instances for l by invoking instance miner, $l.Instances = InstanceMiner(l, S, Sup)$
 - 8: **if** $l.Instances = \phi$ **then**
 - 9: Remove l from L
 - 10: **end if**
 - 11: **end for**
 - 12: Grouping: Group the frequent labels FL according to their lexicographic similarity into concepts C .
 - 13: Mine Relationships: Get the taxonomy of concepts in home pages by invoking relationship miner with the set of concepts C , $T = RelationshipMiner(C, S, Sup)$.
 - 14: **for** each concept c in the taxonomy T **do**
 - 15: Follow the links corresponding to the labels in this concept, fetch the Web pages and add them to S' .
 - 16: Invoke the taxonomy miner to get sub taxonomy for c , $T' = TaxonomyMiner(c, S', Sup)$
 - 17: Attach the sub taxonomy T' under the concept c in the taxonomy T .
 - 18: **end for**
 - 19: Return N
-

4.1 Instance Segments Extraction

The HTML documents usually contain many URLs and we describe an approach to extract the appropriate URLs that point to the instances in this section. To extract the correct instance URLs that point to instances, we adopt the following algorithms. Our system first partitions the Web page using the flat partitioning algorithm described in Section 2.1 and selects the segments of URLs that point to instances. A collections of links from a segment point to instances if the target pages contain similarly structured distinct instances. Our system first extracts the dissimilar regions from each of the collections of urls. This is done by flat partitioning each URL in the collection and aligning the segments based based on the content similarity in the segments. Our system uses Levenstein Distance

Home	Sports	Tennis
Business	Soccer	2003 Wimbledon
World	Golf	Men
Entertainment	Tennis	Women
Sports	Horse Racing	TV Schedule
Weather	College Basketball	
Technology	Hockey	
Education	Olympics	
Health	Olympic Sports	
Privacy Policy	Pro Hockey	

Fig. 8. A fragment of the taxonomy obtained using the approached described for the News domain

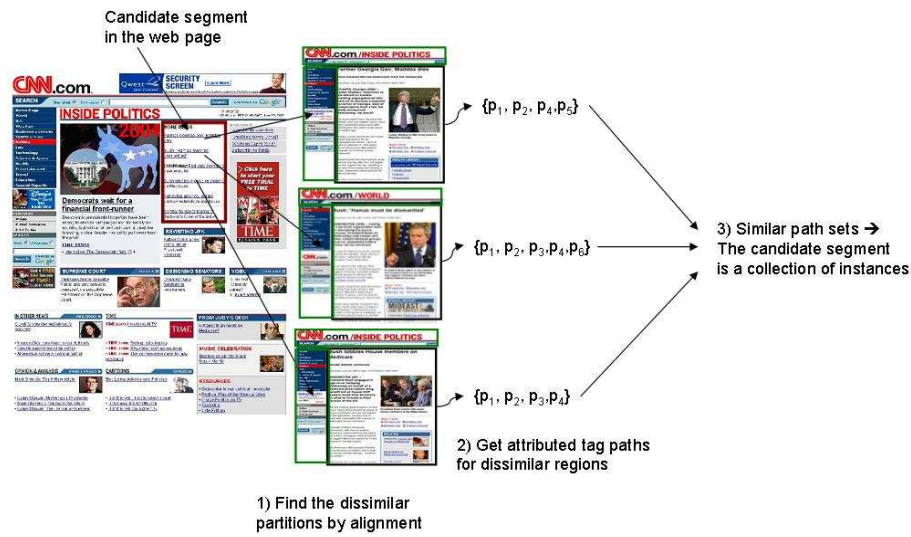


Fig. 9. Finding the segment that contains collection of instance URLs. Each candidate segment in the web page is examined. Each URL in the candidate segment is flat partitioned and the segments are aligned according to their content similarity. The green segments in the URL pages indicate the similar segments and they will be eliminated. The black segments which correspond to dissimilar segments are used to extract attributed tag paths. If the path sets are similar (have more common paths) then the candidate segment in the web page is chosen as the one with instances in it.

(aka edit distance) Measure to align the segments, by making use of Jaccard's Coefficient [15] (calculated as the ratio of the common words to the total number of words in the two segments). After aligning the segments, our system finds the dissimilar segments that are not aligned properly (the places where an insertion, deletion or a replacement has occurred). Next it utilizes Hierarchical Partitioning algorithm described in 2.2 to convert these dissimilar regions into semi-structured XML documents. Later it extracts the attributed tag paths from the LCA (lowest common ancestor) of the leaf nodes in the segment to leaf nodes themselves.

Fig. 10. A Sample Hotel Page that we used in our experiments and the attributes that we extracted from the page encoded as XML. The attribute name labels are capitalized in the XML file to distinguish them from attribute value labels

These path sets extracted from dissimilar regions (instance segments) represent the signature of the instances and our system chooses those instance segments for which these path sets are similar. This process is illustrated in Figure 9 for News Websites.

4.2 Instance Extraction for Labelled and Unlabelled Attributes

From the previous phase we have the hierarchically partitioned instance segments of the instance URLs. To extract instances from these segments, we use the tree miner algorithm described in Section 3. The tree miner algorithm provides us with the hierarchy among the frequent concepts among the instance segments. These frequent concepts correspond to the names of the attributes of the segments. For example, in hotels domain our system identified “Room Amenities”, “Hotel Services”, “Local Attractions”, etc. as frequent labels. These labels correspond to the attribute names and we extract the values for these attributes by finding the children of these labels in the hierarchically partitioned instance segments. We organize these labelled attributes along with their hierarchy in XML documents. Figure 10 shows one of the hotel pages that we used in our experiments and the attributes that we extracted from that hotel page.

But some of the labels may not have any frequent label above or below them. For example in our experiments with News domain, we found that there are no frequent labels across News instance segments. Therefore the labels in these segments correspond to the values of attributes (such as the title of the article, body of the article, author of the article, city, etc.) whose names are not explicitly available in the instance segments. In this case we organize the attributes according to their paths, i.e., we maintain a table of attributes where columns correspond to the paths and rows correspond to each instance segment. Here we the attribute names are unknown and we plan to investigate techniques to mine the attributes by training classifiers for each path.

Experimental Results for Instance Extraction

The precision and recall for the Instance mining for unlabelled attributes is 64% and 97% respectively. Similarly, for the hotel pages the precision and recall values are 80% and 91% respectively.

References

1. Luke McDowell, Oren Etzioni, Steven D. Gribble, Alon Halevy, Henry Levy, William Pentney, Deepak Verma, and Stani Vlasseva. Evolving the semantic web with mangrove.
2. B. McBride. Four steps towards the widespread adoption of a semantic web. In *International Semantic Web Conference*, 2002.
3. S. Haustein and J. Pleumann. Is participation in the semantic web too difficult? In *International Semantic Web Conference*, 2002.
4. Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Natl. Conf. on Artificial Intelligence*, pages 509–516, Madison, US, 1998. AAAI Press, Menlo Park, US.
5. Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.
6. A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *ACM SIGMOD*, 2003.
7. William Cohen, Matthew Hurst, and Lee Jensen. A flexible learning system for wrapping tables and lists in html documents. In *Intl. World Wide Web Conf.*, 2002.
8. Y. Wang and J. Hu. A machine learning based approach for the table detection on the web. In *Intl. World Wide Web Conf.*, 2002.
9. Svetlozar Nestorov, Serge Abiteboul, and Rajeev Motwani. Extracting schema from semistructured data. In *ACM SIGMOD*, 1998.
10. Minos Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. Xtract: A system for extracting document type descriptors from xml documents. In *ACM SIGMOD*, 2000.
11. Yannis Papakonstantinou and Victor Vianu. Dtd inference for views of xml data. In *ACM PODS*, 2000.
12. Gao Cong, Lan Yi, Bing Liu, and Ke Wang. Discovering frequent substructures from hierarchical semi-structured data. In *Proceedings of the Second SIAM International Conference on Data Mining*, 2002.
13. M. Zaki. Efficiently mining frequent trees in a forest, 2002.
14. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
15. R.R. Korfhage. *Information Storage and Retrieval*. John Wiley Computer Publications, New York, 1999.

Can Data Mining Techniques Ease The Semantic Tagging Burden?

Fabio Forno¹, Laura Farinetti¹, Sean Mehan²

¹ Politecnico di Torino, Dipartimento di Automatica ed Informatica, Torino, Italy
fabio.forno@polito.it, laura.farinetti@polito.it

² SMO, University of the Highlands and Islands, Sleat, Isle of Skye, UK
sean@smo.uhi.ac.uk

Abstract. The effective implementation of the Semantic Web vision is highly dependent upon the widespread availability of large collections of semantically rich resources which are trustworthy and meaningful. Since semantic classification is dependent upon complex ontologies, a recognised difficulty is the steep learning curve presented to human classifiers when attempting to utilise such ontologies. One important method to foster an increase in web accessible, semantically tagged resources is to make available tools which allow users to explore and understand relevant ontologies and to present relevant categories with which to tag new data. In this paper we investigate how an important and powerful data mining technique, Latent Semantic Indexing (LSI), might help in the design and implementation of tools that guide users in semantic tagging tasks. We applied LSI to a large portion of the Open Directory Project (ODP) catalogue, one of the largest repositories of semantically tagged resources available today. We computed statistical information concerning category relationships in the ODP data set, and we incorporated structural information by modifying the construction process of the LSI space. Using this basis, we conducted a comparative experiment where a machine generated classification of new documents was evaluated against a classification created by a group of human users. This paper includes an evaluation and discussion of the experimental results.

1 Introduction and goals

The power of the World Wide Web as a mechanism for sharing information in a globally connected network has impacted on wide sectors of society, but methods to increase the effectiveness of the Web are required which should combine machine understandable content and machine reasoning capabilities with the data. To meet the need, researchers have called for the next generation of the web, the Semantic Web (SW) (Berners Lee et al., 2001), where web information objects transform into web knowledge objects.

In this new web generation, web documents will not only be designed for human reading, but also for machine processing. Additional knowledge will be available to provide context and relationships about information objects. The SW will be

built on three components: structured collections of machine understandable information, inference rules with which to conduct machine reasoning, and software agents able to process information and exchange results with other programs.

The most commonly envisioned SW implementation of these components requires: a systematic, shareable, computer-oriented representation of the world (often referred to as an ontology), semantic annotations of web resources, and software agents to retrieve and manage knowledge instead of unstructured data. Many languages for semantic tagging of resources can be found and this has been an active research area (Gomez-Perez and Corcho, 2002), however, to date, very few large-scale *(semi)-structured* collections of resources have used these languages. Examples of such a large scale collection are the Open Directory Project (ODP, 2003) and news and weblog syndications through RSS (Winer, 2002). We feel that this lack of large scale implementation is one of the reasons why available search engines are far from being “semantic”, but rather still use a “keyword” approach. ODP, in particular, is based on the efforts of a small community of users which has produced and currently maintains a general ontological structure in which web resources are manually annotated in RDF. The manual annotation of resources, as a constraint, results in only a few documents, 3 million at present, being semantically tagged compared to the the total web document population; ODP currently holds some 3 million tagged resources out of a conservative estimate of 4 billion documents on the web, meaning the ODP collection is less than 0.075% of the total web population.

A trivial solution would be to enlarge the community of annotators, leading to decentralised management, but this would lead to problems related to trust and to the usage and maintenance of ontologies. The first problem concerns the trust of annotations actually matching between document content and document semantic tags; failure in this regard could lead to spamming, among other problems. The proposed SW architecture addresses this problem, and proposes the creation of a “network of trust”, but Tim Berners-Lee himself (2001) foresees the stage of “trusted web resource” only being established after year 2010. The second problem is a practical issue, concerning a common agreement and understanding of ontologies in use by a large community of annotators, which often have very specific resources to deal with. Automatic or even semi-automatic tagging of resources can be proposed to counteract these issues, yet large-scale semi-automatic tagging of resources is still far from being a reality, mainly due to a lack of user-friendly tagging tools that are effectively linked to ontologies.

In this paper we start with a description of the some of the observed problems that human classifiers have when trying to catalogue a web resource, especially when they are expert in the site domain but not generally expert in classification tasks and are not familiar with the domain ontologies, which often are broader than their direct knowledge of the subjects. This scenario fits well with a highly likely future scenario of the SW, where people will catalogue their own web resources using ontologies designed by others.

For this purpose, after a short introduction to the context of our research in section 2 and a review of related works found in the literature in section 3, we

describe an experiment we conducted to empirically measure the difficulties involved in manual semantic tagging, where the tagging is oriented to document classification for information retrieval and not simply for adding annotations useful to human readers planning to share information. This section includes discussion of the experimental results and argues for the need for an automatic tool capable of easing the semantic tagging burden. Such a tool would allow for convenient browsing of the ontological elements and propose a list of suitable categories from among which to select.

The paper then considers the potential of applying data mining techniques to semantic tagging tasks, and in section 4 we propose preliminary results of an approach that applies a powerful and well assessed data mining technique, Latent Semantic Indexing (Deerwester et al., 1990), to the ODP catalogue. This is done in order to extract information concerning the match between document content and selected ontology categories. The proposed tool can exploit this information for extending the cataloguing to a much larger number of documents, through a learn-by-example approach, proposing the best match ontology elements to human classifiers as part a semi-automatic tagging process. Finally, in this section, a preliminary test of the designed tool is described and discussed.

We conclude the paper with a brief section summarizing our conclusions in section 5.

2 Context of research

The literature has numerous reports of efforts and experiments which extract information from unstructured data; this extracted information is then used to subsequently build some semantically meaningful elements with which to tag the indexed resources. These efforts in fact span over more than 20 years of efforts to extract classification information from such unstructured data sets. Many modern methods use various machine learning techniques, including feature vector representations similar to the approach used in our experiment. For example, in one series of experiments (see Grobelnik et al., 1998), a naive Bayesian classifier was used on text data derived from three domains in the Yahoo hierarchy; from this an n-gram feature-vector document representation was constructed and application of this lead to a high correlation between the human built classification and the classification predicted by the machine learning algorithm. Another approach (Li et al., 2001) was to use a machine learning technique which analysed natural language sentences in documents and annotated, using RDF tags, each prime sentence in the document. This was based on semantic analysis on these natural language sentences using Conceptual Graphs. The major distinction between these kinds of approaches and our own is that these approaches are targeted at automatic classification of existing and new resources whereas our overall aim is to provide a supporting tool to allow human users to use ontologies to semi-automatically tag existing and new resources. Vargas-Vera et al. (see Vargas-Vera et al., 2001) describe a semantic annotation tool for extraction of knowledge structures from web pages through the use of simple

user-defined knowledge extraction patterns. The semantic annotation tool they built contains an ontology based mark up component which allows the user to browse and to mark up relevant pieces of information. Also it contains a learning component which learns rules from examples and an information extraction component which extracts the objects and relations between these objects. Another example is the CREAM framework (see Handschuh et al.,2001). CREAM (Creating RElational, Annotation-based Metadata) is a framework for an annotation environment that allows construction of relational metadata, i.e. metadata that comprises class instances and relationship instances. These instances were not based on a fixed structure, but on a domain ontology. This approach, similar to other similar annotation efforts which are being pursued, differ from the first class of applications in that they are human centric. They do differ from our own approach in that a major aim of these systems is to populate extendible ontologies through the use of the information extraction component, whereas our own is to use an ontology to guide user selection of semantic tags.

We decided to apply LSI to the ODP, since it is a well assessed technique widely described in information retrieval literature. Since its potentials and limitations are well known, it is, therefore, easy to understand the added value of its application to already semantically structured data as opposed to flat corpora. Among the several web directories available on the net, e.g. Yahoo!, we selected the ODP catalog, since it is the only resource annotated with RDF and thus suitable for automatic processing. ODP data is available for download in two large RDF files containing respectively structure definitions and URL annotations. Like the other web directories, ODP can be labelled as a taxonomy with more accuracy than as an ontology, since it does not define strict semantic relationships between nodes, but rather describes generic topic connections (e.g. "Subtopic", "Related") as a graph. URLs are then annotated with one or more topics and a textual description. Though this configuration does not allow agent based reasoning yet, due to its weak semantic components, we believe it provides a good example of the benefits that could derive from semantically structured data in general.

3 Manual tagging

3.1 Experiment description

The goal of the experiment was to replicate and measure the difficulties that humans have when trying to semantically classify web resources. We believe that the real exploitation of the potential of the SW will only happen when large numbers of web resources are semantically tagged, and that this requires that many humans be involved in the tagging task. This is true because many more people are creating web resources who do not have expertise and experience with classification and cataloguing documents than those who do have such skills. Therefore, in our experiment, we purposefully involved people that have a good general knowledge of a subject domain but have no previous experience in cataloguing resources or using ontologies. This profile, in our opinion, fits one

of the largest user groups which will require support in the new SW. Our subjects were graduate students in the fields of engineering, economics and social sciences; 21 students participated in the experiment, and had to perform four tasks as well as completing a questionnaire after each task. They each had to complete and return the questionnaire after each task before going to the following one, to prevent their results being used in the following task.

Task 1 The subjects had a list of web sites to visit, the topics of which were wide ranging; They included sites for computer science, social science, astronomy, science fiction, ethnographic studies and humor. The students had to record the three keywords that, in their opinion, were the most appropriate match to the subject of each site. No list of keywords was provided, so they were completely free to choose any keyword. They also had to provide a written explanation about the strategy they used in performing this task.

Task 2 For each of the 11 web sites previously considered, the subjects had to find the best matching category among the ODP categories. In this task, however, they could only browse the categories and sub-categories without looking at the categories' descriptions nor accessing already catalogued documents as examples. Therefore they could only use category name and the path from the root to the category to determine semantic information about a category. Again, they also had to provide a written explanation about the strategy they used in performing this task.

Task 3 This task was similar to task 2; for each of the 11 web sites the subjects had to find the best matching category among the ODP categories. This time, however, they had access to all the ODP information about the category meaning and could look at the documents catalogued in the categories as examples. Yet again, they had to write a few words about the strategy they used in performing this task.

Task 4 In this task, the students had to record the main difficulties they had experienced while cataloguing the web resources in general, and in performing the previous three tasks in particular. We used this information together with the other collected data to interpret the experimental results.

The following section reports the experimental results and attempts to interpret them, providing an understanding of the difficulties that humans have in cataloguing web resources when they are not classification experts, when they are not familiar with ontologies and when they have no clear guidelines for performing these kinds of tasks.

3.2 Results interpretation

Since we want to provide both a quantitative and a qualitative interpretation of the experimental results, before proceeding with the analysis we try define some numerical quantities in order to provide an objective data framework. However, in general, numerical quantities relating to the process of manual tagging are not used, therefore we first introduce the indices we have used, summarized in table 1. Each table row refers to a web site (11 in total) which has been classified by the subjects that took part in the experiment, while the last row contains the averaged data.

Total keyword number The first three columns represent the total number of keywords used by the subjects in tasks 1, 2 and 3 respectively; they represent a first measure of the dispersion introduced by manual tagging. In tasks 2 and 3, users had to specify one ODP category using the complete path from the root of the ontology, i.e. */Science/Technology/Education/*. In order to be able to compare the dispersion of keywords with task 1, we split the full path into its components, thus obtaining a set of keywords from each topic, e.g. (*Science*, *Technology* and *Education* for the previous example).

Keyword intersection Columns labeled *I12*, *I13*, *I23* contain the number of keywords in common between the sets collected in the different tasks. For example, row 1 states that the keyword set resulting from task 1 has only 7 keywords in common with task 2 and only 6 keywords in common with task 3 respectively, while the latter tasks have an intersection of 19 keywords.

Entropy The last three columns, *H1*, *H2*, *H3*, represent the entropy of each set of keywords in the three tasks. In order to calculate the entropy of a set of keywords we have considered each keyword as a code symbol of a language, and calculated its frequency $p_i = n_i/N$, where n_i is the number of occurrences of the keyword and N the total number of occurrences. Then we have applied Shannon's definition of entropy of a codebook (1963):

$$H = - \sum_i p_i \log_2(p_i)$$

Information Theory interprets H as the total amount of information a codebook can carry. Physics interprets H , as in statistical mechanics, as the measure of the disorder of a system; importantly, the two interpretations are not in conflict (Ott, 1993). If we consider a set of symbols as growing more ordered as any particular symbol's frequency increases, then we may also interpret this as less information overall contained in the set from an information theoretic standpoint, with both of these perspectives being labelled as low entropy in the set. Hence we can use H as a measure of the degree of agreement between tags selected: high levels of agreement between selected keywords produces a smaller set of unique keywords, or a codebook with low H ; Alternatively, low agreement between keywords selected results in a large set of unique keywords, or a codebook with high H . We choose to

use H , rather than n_i , because it is a more meaningful measure. A codebook with a large N , but a small number of them recurring with high frequency, is more ordered than the same codebook with symbols recurring with equal frequency.

Keyword distribution Continuing with the codebook analogy, in figure 1 we plot the keyword recurrence frequency distribution obtained in each of the three tasks of the experiment. For each web site we have sorted the set of keywords in reverse order according to their frequency. The distribution magnitude relates to the size of the core of a keyword set, i.e. the set of the most commonly recurring keywords according to a parameterized metric.

Table 1. Keyword total number, intersection and entropy for each task of the experiment.

Site	T1	T2	T3	I12	I13	I23	H1	H2	H3
1	34	22	22	7	6	19	4.39	3.76	3.81
2	36	28	33	9	10	20	4.55	4.22	4.52
3	28	27	30	7	10	18	3.98	4.02	4.31
4	34	17	7	5	4	7	4.44	3.08	2.53
5	38	20	19	4	4	14	4.50	3.43	3.43
6	40	15	14	4	8	7	4.66	2.83	2.58
7	26	26	22	10	10	16	3.70	3.80	3.44
8	40	37	32	10	11	24	4.58	4.80	4.57
9	28	29	23	7	5	20	3.75	4.17	3.60
10	38	24	24	3	4	17	4.45	3.96	3.88
11	28	44	38	6	6	33	4.21	4.65	4.38
Avg	33.63	26.27	24.00	6.54	7.09	17.72	4.29	3.88	3.73

The first characteristic of the tagging process determined from the indices is the high number of keywords used by the classifiers in the three tasks. In the first task, where students freely selected three keywords describing the web sites, there was an average of 33 unique keywords used to describe a site. This means that the each student chose, on average, 1.5 unique keywords to describe the same site. A high keyword dispersion level was foreseeable and our data supports the hypothesis that classification is a highly subjective task. More interesting was the relatively high dispersion of keywords in tasks 2 and 3. The total number of selected keywords decreased, but each user still contributed more than one new keyword to the set. This fact could be interpreted in the following ways:

- the use of ontologies is of little help for users who have little experience in classification tasks;
- in certain situations ontologies may represent an obstacle for users (see, for example, row 11 in table 1) in that when they start looking for a suitable category, they have already mentally selected a classification, perhaps even

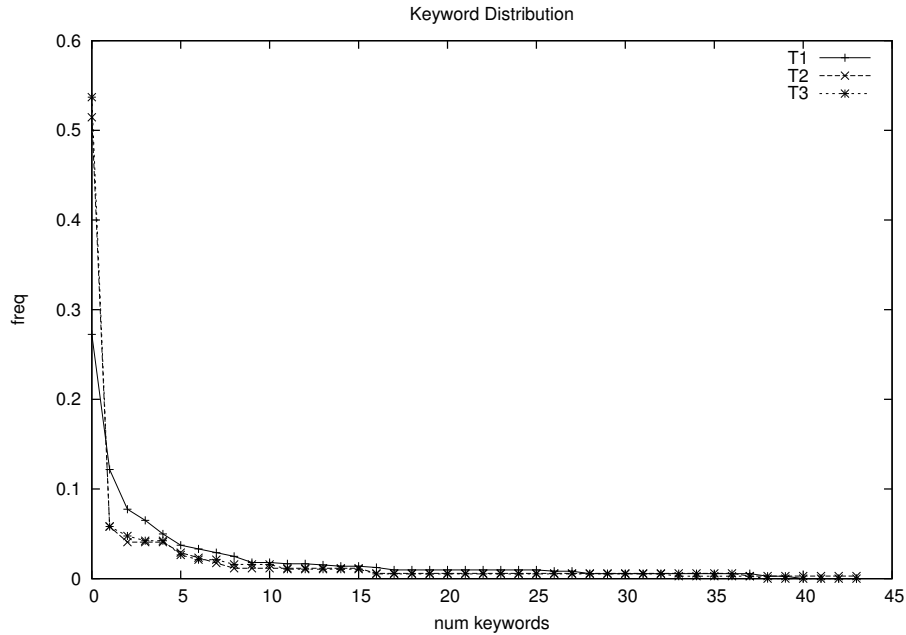


Fig. 1. Keyword distribution according to usage

- subconsciously. Subsequently, they are influenced by their own predetermined category and are unable to find a matching category;
- the ODP ontology is a wide and dispersive taxonomy, with the aim of classifying almost everything present on the web: this may lead to incongruities, and also to classification areas with inappropriate granularity.

These results are not surprising, since ontologies have been designed by experts to facilitate automated reasoning, not to ease the human tagging burden. We may even theorize that the more an ontology is specialized and finely grained, the more human taggers will have difficulties in selecting the correct categories from it. Therefore, there is an evident mismatch between the need of precision and specialization of ontologies destined for machine reasoning, and the ability of humans to utilize them.

Another indication of the mismatch between the human classifier’s mental processes and the ontology they are forced to use may be derived from our intersection indices. In average only 6 keywords out of the 33 selected in task 1 have been picked also in task 2, less than 20%. This percentage is similar but slightly higher to the intersection between tasks 1 and 3. If confirmed with a larger number of test cases, this increase could be interpreted that when given more time to explore the ontologies, human classifiers are inclined to look for categories matching their first choice.

Comparing tasks 2 and 3 we note two more trends: the small decrease of keyword

dispersion and the higher intersection of chosen keywords. Possible interpretations include:

- the diminution of the dispersion may suggest that users, with adequate support to navigate the ontology, may choose the appropriate categories more easily, with less frustration during the process;
- examples of classification, which are the only additional support that users have when moving from task 2 to task 3, are not sufficient to improve significantly the similarity of classification outcomes; as confirmed by user comments, support tools for ontology exploration could lead to more agreement in tagging; lack of concordance, in fact, seems to be not due to low user concept comprehension, but rather to user difficulty in acquiring a broad understanding of the ontology.
- during tasks 2 and 3 users may have explored only a small part of the ontology, and in both cases they learn to not explore it further; this is another indication that there is a need for tools assisting in the navigation of ontologies.

The analysis of the H_i supports the keyword scattering we have already observed. The calculated entropy in all the tasks always remains high. If all the human classifiers had selected the same three keywords, the H would equal 1.58; with a completely random keyword selection, i.e. all keywords are unique, H would equal 5.97. The average values in table 1, 4.29, 3.88 and 3.73, testify to the high degree of dispersion in all the three tasks (since H is logarithmic, these values are significantly different).

With respect to figure 1, we find that, even if the keyword dispersion is high there is a very small set of keywords (one or two) that nearly half of the subjects identify in all the tasks. This is reasonable for tasks 2 and 3, since the sets must always contain the same root keywords. It is more interesting to remark that also in task 1 there is a kernel of common keywords; as a consequence we may imagine that tools suggesting categories of the ontology starting from keywords expressed in free form by users may facilitate the tagging process and lead to more concordant results.

The analysis of the last experimental task supports some hypotheses in our model describing the difficulties involved in manual tagging, and offers a qualitative description of the problems from the point of view of human classifiers. Here we summarize the main difficulties the subjects reported encountering:

- the deep structure and the presence of many internal links connecting different tree branches allows the user to become easily lost in the structure;
- many categories are too semantically similar and classifiers have difficulties in selecting just one topic;
- category names and associated keywords sometimes do not provide enough hints about the contents, thus hindering navigation;
- some classifiers complain that the ontology is not exhaustive, but we think rather that the correct interpretation is given by other classifiers reporting that the ontology does not cope with their own personal classification.

Several classifiers also described the strategy that they adopted in the process:

- when exploring the ontology users start from the task 1 keywords;
- the exploration usually proceeds depth first;
- the presence of the *related* topics links also allows some exploration in breadth;
- some users autonomously followed an approach similar to our proposed method, using the Google search engine with the keywords found in task 1 in order quickly find some matching categories; therefore, we think that the support of tagging tools operating on user suggested keywords will be intuitive to use.

In summary, classifiers often start from a set of keywords they choose independently from the given ontology. This set derives from their own experience with the web site topic and from their personal usage of the language, and thus is very likely to *not* match with the terms present in the given ontology. Then they explore the ontology in order to find the best match for their own keywords, but they encounter many difficulties due to the ontology topics not matching well with their own, preselected topics and also due to difficulties encountered while exploring the ontology.

These last conclusions support our hypothesis that a tool is needed which is able not only to facilitate users browsing ontologies (which, in itself, would provide a good level of support), but also able to automatically suggest a restricted list of categories from which the human classifier can select the best one.

4 Automated support for manual tagging

4.1 Latent Semantic Indexing

We first give a brief introduction to LSI. In information retrieval a query is usually expressed in a vector space (Salton, 1975), where vectors represent documents and the vectorial components formed from the document constituent terms. Given a similarity measure, the document vectors best matching a query vector can be selected. LSI operates in a similar fashion, except that a transformation to the *term* \times *document* space is achieved through the application of a Singular Value Decomposition (SVD). A corpus of n documents with m indexing terms is represented by the matrix $A^{(m \times n)}$, where each element a_{ij} represents the weight of i^{th} term in document j . Several weighting schemes for term weighting have been proposed in the literature 1996 and LSI normally uses a functional combination of local and global weights; thus $a_{ij} = L(i, j) * G(i)$, where $L(i, j)$ is the weight of the i^{th} term in document j , usually its frequency, and $G(i)$ is a function of its global weight in the corpus, usually term entropy dependent upon inverse document frequency. Subsequently the truncated SVD is applied to the matrix A , yielding:

$$A_k = U_k \times \Sigma_k \times V_k^T$$

where $\Sigma_k \equiv \text{diag}(\sigma_1 \cdots \sigma_k)$ is a diagonal matrix, with the k largest singular values of A , and $U_k \equiv (u_1 \cdots u_k)$ and $V_k \equiv (v_1 \cdots v_k)$ are the associated left and right singular vectors respectively. In (Berry, 1995) it is shown that A_k is the best approximation in k dimensions of the original matrix A . The result of this transformation is that the original vector space model is folded into a much smaller subspace, which should better summarize word and document relationships, capturing hidden *semantic* links in smaller feature vectors (Berry et al., 1995, Hoffmann, 1999, and Papadimitriou et al., 1998).

In the LSI space, queries are expressed as in the original vector space, where, given a vector q representing the set of query terms, each document score is given by $s_i = q^T x_i$ (x_i is the feature vector of document i). In LSI the query vector is simply projected into the k subspace, via: $s = (q^T U_k)(\Sigma_k V_k^T)$.

4.2 Application of Latent Semantic Indexing to the Open Directory

Our tagging tool is based on the application of LSI to a large portion of previously tagged resources in the ODP; specifically, we selected all the Science, Computers, Arts, Society and Recreation subtrees, resulting in a collection containing more than 50,000 topics and 1,000,000 URLs. In this ontology we find two distinct types of resource: (i) topics organized in a tree-like structure, all having a short textual description of the content, and (ii) links tagged with one or more topics and a brief description. LSI is normally applied to flat document corpora, without any structure; since the associated vector space model does not take into account existing semantic relations and not lose this critical information, we had to make some changes to the process whereby the *term* \times *document* matrix is constructed. In our experiment we have built several vector spaces following different methods, in order to evaluate the best ways to incorporate structural information.

In all matrices we have used the same scoring function for terms:

$$a_{ij} = f_{ij} \cdot G(\log_2(\text{idf}(i)), \log\text{Mean})$$

where f_{ij} is the i^{th} term frequency in the document j , and the second term gives its global score, computed as i^{th} term inverse document frequency (*idf*) and weighted by the Gamma distribution centered in *logMean*. *logMean* is a parameter dependent upon the size of the document collection. In this modified weighting scheme, based on Salton (1975), our previous experiments outperformed entropy based measures, since the modified scheme penalizes terms recurring with a very high frequency (stopwords were already pruned in document preprocessing).

In the following we describe the four different methods we used to attempt to incorporate the structural information into the LSI spaces, and we shall refer to these as $A_k^{(1)}$, $A_k^{(2)}$, $A_k^{(3)}$ and $A_k^{(4)}$, respectively. In $A_k^{(1)}$ we considered topic descriptions as documents, enriched with the descriptions of contained URLs. Since the quantity of associated text was not large, the matrix scaled well; in fact, with this strategy it is conceivable to process the entire ODP with consumer level hardware.

In $A_k^{(2)}$ we inserted as documents both topic descriptions and text extracted from indexed URLs. In this case, in order to scale to the dimension of our experiment, we have applied a random selection of documents and terms, as normally performed in LSI.

In $A_k^{(3)}$ we used only topic descriptions again, similar to $A_k^{(1)}$, but we started incorporating structural information. We made the assumption that, since the ODP has a tree-like structure, a topic is qualified not just by its description, but also by the description of its child nodes. Thus, in building $A_k^{(3)}$, we associated to each topic also the text of its children (we ignored *related* topics and *symbolic* links). The resulting matrix was less sparse, but since for our SVD implementation the occupied memory is given by $k \cdot \min(m, n) + nnzeros$, where k is the LSI dimension and $nnzero$ the number of non zero values in the matrix, and also, since the first addend usually grows faster, this approach continued scaling well.

In our last matrix, $A_k^{(4)}$, we used text from indexed documents, but we employed a different method from random selection. Random selection, in fact, is based on the assumption that we do not know if there is a structure in the indexed corpus, thus a completely random choice of the documents and keywords is the best guess we can make in order to obtain a representative subset. On the other hand, ODP catalogued URLs are already grouped by topics, and they form a bunch of small clusters of manually selected examples for each category. We exploited this information, extracting the core of most frequent words for each topic, and we then used this new lexicon as the associated text for the topics. We weighted words with their frequency in each set, and we recomputed their global weight according to the new lexicon. Using the codebook analogy described in section 3, and performing some tests, we have established a heuristic that keeping the words contributing to the central 75% of the associated cumulative distribution function is a good compromise between keyword set size and retained information. Without a given structure, this selection would have been ineffective or even counterproductive, since we would have retained only common usage words and not topic representative keywords.

4.3 A preliminary experiment and discussion

In the experimental phase we have evaluated two distinct aspects of our LSI semantic tagging tool, namely: (i) whether the incorporated structure information improves LSI performance in terms of recall and precision (Raghavan, 1989, and (ii) whether it can be exploited as a support for manual tagging.

Benchmarks on text retrieval performance, in terms of precision and recall, are usually performed on a fixed set of queries on standard corpora. To assess the capability of our LSI implementation in capturing deal with structural information, we defined a set of queries composed of few keywords and spanning several ODP terms and we applied them to all the $A_k^{(i)}$. During the evaluation of the results we had to consider that the ODP structure presents many overlaps, and the same topic may be present in different branches (e.g. computational linguistic resources are present both under */Computers/Artificial_Intelligence/* and

/Science/Social_Sciences/Languages_and_Linguistics/). Therefore, in contrast to a simple count of returned pertaining topics (leaves) which would not have resulted in significant understanding, we concentrated mostly on the relevance of the returned major topics (internal nodes).

Evaluating query results, we found that both $A_k^{(3)}$ and $A_k^{(4)}$, built using structural information, outperform the corresponding $A_k^{(1)}$ and $A_k^{(2)}$, built with flat corpora in both precision and recall. Specifically, $A_k^{(1)}$ and $A_k^{(2)}$ behaved well for certain topics and completely missed others. The reason for their irregular behavior must be investigated further, but we believe it may derive from the lack of sufficient representative keywords for some topics; with random selection, the method results in topics with more catalogued resources being better represented, since their elements have more chances to be selected. In $A_k^{(3)}$ and $A_k^{(4)}$, we improve the selection of meaningful keywords, since they are either inherited from child nodes ($A_k^{(3)}$) or selected using topic aggregation of resources ($A_k^{(4)}$). Finally matrix 4, besides scaling very well in terms of memory occupation, since we can considerably reduce the lexicon size, scores a better relevance than matrix 3. We think that the worse performance of matrix 3 might be found in the heterogeneity of document lengths. Its building process, the inheritance of describing text from child nodes, in fact, produces short documents for nodes deep in the structure, and very long ones for nodes near to the root. Since the behavior of LSI is not well understood when document length has a variance of magnitude orders, we believe that the vector space represented by matrix 3 needs more investigation.

We then evaluated whether obtained results could be used in order to help manual tagging. The main issues which emerged in the 3.2 related to user difficulties with their orientation in the ODP structure, as opposed to the need of exploring it in depth or breadth. Thus, human taggers could exploit LSI derived information, in order to obtain a few meaningful suggestions as starting points in the ODP structure. There are two possible approaches: (i) using documents to be tagged as queries in the LSI space, (ii) allowing user to express some keywords, or a short document description and use these as queries. Returned results would then be sent to users as possible starting points for ODP exploration.

We experimented with both methods on our $A_k^{(i)}$, obtaining promising results. As an example, in table 2 we list the suggested categories using strategy (i) for one of the documents we used in the experiment of paragraph 3. The document is the homepage of L. M. Krauss, the author of books about science in science fiction, where it can be seen that the suggested categories are very relevant. In contrast, the same document gave several problems to manual taggers, since they looked for Star Trek (the main topic of the books) under the ‘TV Shows’ area, and they had difficulties in finding relevant topics (alternative science) under the ‘Science’ section.

Our experiments demonstrate that queries composed of a few keywords usually matches to relevant categories more efficiently than using only documents themselves. This may be due to the fact that the few keywords should give a better

definition of a document in the LSI space rather than the entire document, since many contained words may be considered *noise*. We observed also that intermediate nodes of proposed topics are the best candidates to suggest to users as starting points. Deep topic nodes, in fact, can suffer from some noise that LSI has not been able to eliminate and their precision may be low; in contrast the precision of intermediate nodes in the results in performed queries is very high.

Table 2. Some of the suggested categories for the homepage of L.M. Krauss.

/Science/Anomalies_and_Alternative_Science/Astronomy,_Alternative/Cosmology/
/Science/Social_Sciences/Ethnic_Studies/
/Science/Physics/Relativity/People/Hawking,_Stephen/
/Science/Math/Applications/Mathematical_Economics_and_Financial_Mathematics/
/Science/Astronomy/Education/
/Science/Astronomy/History/People/Kepler,_Johannes/
/Science/Physics/Alternative/Superluminal_Physics/
/Science/Earth_Sciences/Geology/Geochronology/Radiometric_Dating/
/Society/Future/Predictions/Scientific/
/Society/History/By_Topic/Science/

5 Conclusions

For the SW to become a reality, there needs to be an extensive and rich set of web objects which have associated semantic information. It is generally accepted that there will be common repositories of domain specific semantic information, namely those collections which are generally referred to as ontologies within the relevant communities. This paper has identified and analysed some of the problems that educated but non-expert users experience in trying to apply existing ontology information when classifying and tagging data resources. Using the ODP, we observed a number of users completing classification tasks with ODP categories.

The results of our user experimentation indicate that, even with access to structured ontological knowledge, the classifiers chose many different resource descriptive tags from one another, resulting in a high level of dispersion amongst the final tag set. It seems that ontologies are not immediately useful to experienced users who do not have extensive experience with classification tasks. Most normal users have a preconceived term in mind when tagging a resource and when this preconception does not map to the ontology, conflict often occurs.

It is reasonable to speculate that the more detailed the ontology, the more serious this conflict with normal users of the ontology can be. Complex ontologies seem to lose users who are trying to navigate them, and this leads to demotivation on the part of the manual classifier to explore the ontology fully. Users

often complain of lack of completeness in ontologies, and we interpret this as a clash between the ontology conceptualization and the user's predetermined conceptualization in the classification task. Classifiers seem to start with a set of terms from their own experience and language registers, independent of those found within the ontology.

These difficulties support our hypothesis that a tool is needed which is able to ease the manual semantic tagging burden by the widest population of content creators when using standardized ontological resources. This tool should help users to explore ontologies, but also should automatically suggest a restricted set of tagging terms drawn from the ontology to help guide the user in their classification tasks. We have experimented with a number of different applications of LSI to ODP tagged resources. Importantly, we have experimented with new methods of incorporating semantic structural information from the initial vector space into the transformed LSI space and we have tested whether the incorporated information improves querying performance and whether it can be exploited as a support mechanism for manual semantic tagging. We found that through the incorporation of this structural information, we were able to improve both query precision and recall. This results in more meaningful selection of keywords from the ontology, which would result in more effective semantically tagged resources. In pursuit of this, we experimented with two different approaches, namely to use documents which were to be tagged as query vectors in the $A_k^{(i)}$ space, or allowing classifiers to supply some of their own preconceived terms as keywords or a short description and constructing query vectors from these. LSI seems to bridge the gap between the ontology terms and terms preconceived by human classifiers. LSI projects in the same space $term \times documents$ relations which were learned already from tagged documents using the ontology and those terms in the mind of the original ontology compilers.

References

- Berners-Lee, T., Hendler, J.: Scientific Publishing on the "Semantic Web". Nature Webdebates, <http://www.nature.com/nature/debates/e-access/Articles/bernerslee.htm>, April 2001.
- Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, 5/01, May 2001.
- Berry, M.W., Dumais, S.T., O'Brien, G.W.: Using Linear Algebra for Intelligent Information Retrieval. SIAM Review 37(4):573-595, 1995.
- Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41(6), 391-407, 1990.
- Gomez-Perez, A., Corcho, O.: Ontology languages for the Semantic Web. IEEE Intelligent Systems, Volume 17(1):54-60, Jan/Feb 2002.
- Grobelnik, M., Mladenic, D.: Efficient text categorization. Text Mining workshop on the 10th European Conference on Machine Learning ECML98.
- Handschuh, S., Staab S., and Maedche, S.: CREAM- Creating relational metadata with a component-based, ontology-driven annotation framework. Proc. of ACM K-

- CAP 2001 - First International Conference on Knowledge Capture, Victoria, BC, Canada, October 21-23, 2001.
- Hoffman, T.: Probabilistic Latent Semantic Indexing. Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval (1999) pp. 50-57.
- Li, J., Zhang, L., and Yu, Y.: Learning to Generate Semantic Annotation for Domain Specific Sentences. K-CAP 2001 workshop on Knowledge markup and semantic annotation, Victoria, BC, Canada, 21 October, 2001.
- Ott, E. Entropies. 4.5 in Chaos in Dynamical Systems. New York: Cambridge University Press, pp. 138-144, 1993.
- Open Directory Project. <http://www.dmoz.org>. 2003.
- Papadimitriou, C.H., Raghavan, P., Tamaki, H., and Vempala, S.: Latent Semantic Indexing: A Probabilistic Analysis. In Proceedings of the ACM Conference on Principles of Database Systems (PODS), Seattle, 1998.
- Raghavan, V. V., Jung, G. S, and Bollmann, P.: A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance. ACM Transactions on Office Information Systems, pages 205–229, July 1989.
- Salton G., Wong, A. and Yang, C. S: A Vector Space Model for Automatic Indexing. CACM, Vol. 18, No. 11, 1975, 613-620.
- Salton, G. and Buckley, C.: Term Weighting Approaches in Automatic Text Retrieval. Technical Report TR87-881, Department of Computer Science, Cornell University, 1987. Information Processing and Management Vol.32 (4), p. 431-443, 1996.
- Shannon, C. E. and Weaver, W.: Mathematical Theory of Communication. Urbana, IL: University of Illinois Press, 1963.
- Vargas-Vera, M., Motta E., Domingue S., Buckingham Shum S., and Lanzoni M.: Knowledge extraction by using an ontology-based annotation tool. K-CAP 2001 workshop on Knowledge markup and semantic annotation, Victoria, BC, Canada, 21 October, 2001.
- Winer, D.: RSS 2.0. <http://backend.userland.com/rss>, Aug 2002.

Formal aspects of querying RDF databases

Claudio Gutierrez¹, Carlos Hurtado¹, and Alberto Mendelzon²

¹ Department of Computer Science, Universidad de Chile
{cgutierr,churtado}@dcc.uchile.cl

² Department of Computer Science, University of Toronto
mendel@cs.toronto.edu

Abstract. We study formal aspects of querying databases containing RDF data. We present a formal definition of a query language for RDF and compare it with other proposals. Our language is intended to make it easy to formalize and prove results about its properties. We study novel features of query languages derived from the presence of blank nodes and reification. Finally we provide complexity results for query processing, static optimization of queries, and redundancy elimination in answers.

1 Introduction

The *Resource Description Framework (RDF)* [10] is the proposal of the W3C for a standard metadata model and language. RDF follows the W3C design principles of interoperability, evolution and decentralization. The RDF model is simple: the universe to be modeled is a set of *resources* (essentially anything that can have a *universal resource identifier*, URI); the language to describe them is a set of *properties* (technically binary predicates); descriptions are *statements* very much in the subject-predicate-object structure, where predicate and object are resources or strings. Both subject and object can be undetermined objects, known as *blank nodes*. The subject or object of an RDF statement can be another statement, a feature known as *reification*. A vocabulary of *properties* for this language can be defined along the lines given in the RDF Schema language [12].

Languages for querying RDF have been developed in parallel with RDF itself. We can mention rdfDB [3], an influential simple graph-matching query language from which several other query languages evolved. Among them, SquishQL [5] is a graph-navigation query language that was designed to test some of the functionalities of an RDF query language. It adds constraints on the variables and returns as results a table. SquishQL has several implementations like RDQL and Inkling [5]. RQL [4] has a very different syntax based on OQL, but can perform similar sorts of queries. It is a typed language following a functional approach and supports generalized path expressions. Other languages are Triple [9], a query and transformation language, QEL [7], a query-exchange language designed to work across heterogeneous repositories, and DQL [15], a language and protocol for querying DAML+OIL knowledge bases. Good surveys are [16, 17].

1.1 Problem Statement

There is very little research so far on foundational aspects of these languages, such as query semantics and the complexity of query processing. Such research is made necessary by the new features that arise in querying RDF graphs as opposed to standard databases; in particular, two main differences that deserve formal study are blank nodes and reification.

The presence of blank nodes in RDF graphs introduces redundancy. Furthermore, queries themselves, as we will show later, can create redundancy. Central issues in RDF query processing are how to keep RDF graphs as concise as possible, and what is the computational cost of obtaining such representations.

In order to support reification, the language needs expressiveness beyond what is encountered in classical databases. Another open issue is whether this extra expressiveness boosts the complexity of query processing and size of answers compared to classical databases.

1.2 Contributions

We study formal aspects of querying databases containing RDF data. We view RDF specifications as data (although we keep its knowledge base semantics), and study how to efficiently retrieve information from them.

This paper presents:

- A formal definition of a query language for RDF and comparison with other proposals (such as DQL). We present the language in a streamlined form that is not intended for practical use, but to make it easy to formalize and prove results about its properties.
- A formal study of novel features of query languages derived from the presence of blank nodes and reification, and the differences with standard languages studied in the database community.
- Complexity results for query processing, static optimization of queries, and redundancy elimination.

1.3 Related Work

One point of view has considered RDF metadata as a knowledge base and applied knowledge representation and reasoning techniques to RDF metadata. An example of this approach is DQL, a query language for the Semantic Web proposed in [15]. We discuss the database aspects of DQL in Section 3.4.

Another point of view follows the SQL/XQL approach, which views RDF metadata as a relational or XML database. We already mentioned several proposals and working implementation of such languages. They mainly concentrate on expressiveness and implementation issues. Although very rich from these points of view, none of them address formal issues. There are studies comparing features of these languages, such as syntax (body, head and variables in the query), serialization (XML, N3, ASCII), implementation aspects, etc. See for example [16, 17].

2 Preliminaries

In this section we present the RDF model following the W3C documents [10–12] and discuss different variants of some notions.

2.1 RDF graphs

Assume there is an infinite set U (RDF URI references); an infinite set $B = \{b_j : j \in \mathbb{N}\}$ (Blank nodes); and an infinite set L (RDF literals). A triple $(v_1, v_2, v_3) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an *RDF triple*. We often denote by UBL the union of the sets U , B and L .

Definition 1. *An RDF graph (just graph from now on) is a set of RDF triples. A subgraph is a subset of a graph.*

A graph is *ground* if it has no blank nodes.

Two graphs G_1, G_2 are *isomorphic*³ if G_2 is obtained from G_1 by renaming its blank nodes by blank nodes in a consistent manner (i.e. avoiding name clashes). (Note that if G_2 can be obtained from G_1 in this way, then so can G_1 from G_2 .)

The *merge* of two graphs G_1, G_2 is defined as the union of the set of triples of G_1 and G'_2 , where G'_2 is an isomorphic copy of G_2 whose set of blank nodes is disjoint with that of G_1 . (Note that the merge is unique up to isomorphism).

2.2 RDF graphs and standard graphs

RDF graphs are not quite classical graphs. They resemble labelled graphs with the particularity that edge labels are chosen from the set of nodes of the graph.

Definition 2. *1. A pseudograph is a triple (V, E, f) where V is a finite set of nodes, E is a finite set of edge names, and f is a function from E to $V \times V$. (That is, a directed graph that allows self-loops and multiple edges between pairs of nodes).*

An edge-labeled pseudograph (just pseudograph from now on) is a pseudograph with an additional labeling function $\ell : E \rightarrow V$.

2. Two pseudographs (V_i, E_i, ℓ_i) are isomorphic if (1) There is a (directed) graph isomorphism $\phi : (V_1, E_1) \rightarrow (V_2, E_2)$, and (2) $\phi \circ \ell_1 = \ell_2 \circ \phi$.

With the previous definition, an RDF graph is a pseudograph where V is a disjoint union of three sets: the URI References, Blank nodes, and Literals, respectively, that appear in any triple in the RDF graph, and with the following restrictions: (1) source nodes cannot be literals; (2) The image of the labeling function ℓ is contained in the set of URI references. The next theorem follows directly from the definitions.

Theorem 1. *Two RDF graphs G_1, G_2 are isomorphic if and only if there is an isomorphism $\phi : G_1 \rightarrow G_2$ of pseudo-graphs such that ϕ preserves URI references and literals.*

³ In the RDF Concepts document [13] this notion is called “equality” of graphs.

Note 1 (Encoding of Standard Graphs). Note that standard graphs can be encoded by RDF graphs as follows. Choose a distinguished URI reference e . For a graph $G = (V, E)$, choose a set S of distinct blank nodes of the same size as V , and a bijection $c : V \rightarrow S$. The graph G is encoded by the RDF graph $\{(c(u), e, c(v)) : (u, v) \in E\}$.

2.3 Semantics of RDF graphs

In this section we deal with *simple* RDF Graphs, i.e. those that do not use a pre-defined vocabulary (class, subject, object, etc.) defined in RDF Schemas; in Section 3.3 we deal with a fragment of RDF(S) vocabulary. We implicitly use the same model-theoretic semantics as the W3C RDF Semantics document [11], although all we need for our purposes is to define entailment, which is characterized by Theorem 2 below, and corresponds roughly to logical consequence between the logical specifications defined by both graphs. The RDF Semantics document [11] describes entailment as follows: “Entailment is the key idea which connects model-theoretic semantics to real-world applications. If A entails B, then any interpretation that makes A true also makes B true, so that an assertion of A already contains the same ”meaning” as an assertion of B; we could say that the meaning of B is somehow contained in, or subsumed by, that of A.”

A *mapping* is a function $\mu : \text{UBL} \rightarrow \text{UBL}$ preserving URI references and literals (i.e., $\mu(u) = u$ and $\mu(l) = l$ for all $u \in U$ and $l \in L$). Given a graph G , we define $\mu(G)$ as the set of all $(\mu(s), \mu(p), \mu(o))$ such that $(s, p, o) \in G$. A mapping μ is *consistent* with G if $\mu(G)$ is an RDF graph (i.e., if s is a subject of a triple, $\mu(s) \in \text{UB}$). In this case, we say that the graph $\mu(G)$ is an *instance* of the graph G . An instance of G is *proper* if $\mu(G)$ has fewer blank nodes than G . (This means that either μ instantiates a blank node or identifies two blank nodes of G .)⁴

The following theorem characterizes entailment among RDF graphs. For the purposes of this paper, we can think of it as the definition of entailment.

Theorem 2 (cf. RDF Semantics [11], Interpolation Lemma). *Let G_1, G_2 be RDF graphs. Then G_1 entails G_2 (denoted $G_1 \models G_2$) if and only if an instance of G_2 is a subgraph of G_1 .*

We say that two graphs are *equivalent* (denoted $G_1 \equiv G_2$) if $G_1 \models G_2$ and $G_2 \models G_1$.

Example 1. A graph G entails any of its subgraphs.

Example 2. Consider the RDF graphs $G_1 = \{(a, b, c), (X, b, c), (a, b, Y)\}$ and $G_2 = \{(U, b, V), (V, b, c)\}$, and $G_3 = \{(a, b, c), (X, b, Y)\}$, where capital letters indicate blank nodes. Then $G_1 \models G_3$ but $G_1 \not\models G_2$.

⁴ In the RDF Semantics document [11] “proper instance” refers only to the case where a blank node is instantiated.

Note 2. Yang and Kifer, in [14], present an F-logic version of RDF. They define two notions of entailment: \models and \approx . The first corresponds to the standard notion defined in the RDF Semantics document of the W3C as defined in Theorem 2. The second corresponds to a notion obtained using the same characterization of Theorem 2, but considering only non-proper mappings in the definition of instance.

2.4 Minimal representations: lean graphs

Conciseness in RDF is bound to the notion of *lean* graphs. In this section we study lean graphs.

Definition 3. *A graph G is lean if no proper instance of G is a subgraph of G . (That is, there is no mapping μ such that $\mu(G)$ is a proper subgraph of G .)*

Note 3. There is a significant difference between this notion of “lean” and the one stated in the RDF Semantics document [11]. The document reads: “a graph is *lean* if none of its triples is an instance of any other.” Our definition captures more precisely the notion of “no redundancy” in an RDF graph which is the idea behind the concept of “lean”. While every graph that is lean in the sense of that document is lean in our sense, the converse is not true. See for example the graph G_1 in Example 3, which under the RDF Semantics document definition is not lean.

With our definition we can still prove Anonymity Lemmas 1 and 2 in [11]:

Anonymity Lemma 1: A lean graph does not entail any of its proper instances.

Anonymity Lemma 2: If E' is obtained from a lean graph E by identifying two distinct blank nodes, then E does not entail E' .

Our notion has other desirable properties, e.g. Theorem 3 below.

Example 3. Let $G_1 = \{(X, b, d), (X, b, c), (Y, b, c), (Y, b, e)\}$ and $G_2 = \{(a, b, c), (X, b, c), (Y, b, c)\}$. Then G_1 is lean, but G_2 is not lean.

Fundamental issues regarding lean graphs have not been yet studied. The first fundamental question that arises is whether there is a unique lean graph equivalent to a given one. The following theorem answers this question:

Theorem 3. *Each RDF graph is equivalent to a unique (up to isomorphism) lean graph.*

Proof. Define in the set of RDF graphs, the relation $G \Rightarrow \mu(G)$, if μ is a mapping and $\mu(G)$ is a proper subgraph of G . The relation \Rightarrow has the property: if $B \Leftarrow A \Rightarrow C$, then there is D such that $B \Rightarrow^* D$ and $C \Rightarrow^* D$ (where \Rightarrow^* is the transitive closure of \Rightarrow). The proof of this goes as follows: Let $B = \mu_1(A)$ and $C = \mu_2(A)$, and consider the mapping $\mu_2\mu_1$. Then, because $(\mu_2\mu_1)(\mu_2\mu_1)^j(A)$ is a subgraph of $(\mu_2\mu_1)^j(A)$, for some finite $k \geq 1$ it holds that $(\mu_2\mu_1)^k(A)$ is isomorphic to $(\mu_2\mu_1)^{k+1}(A)$.

From its definition, it follows that the relation \Rightarrow clearly cannot have infinite chains $A_1 \Rightarrow A_2 \Rightarrow \dots$. From the above argument, it also follows that \Rightarrow is

confluent. (For rewriting concepts, see [2].) Hence for each G there is a unique G_* such that $G \Rightarrow^* G_*$ and G_* is irreducible with respect to \Rightarrow (i.e. is lean). This is the desired unique lean graph.

The following results show that it is hard to compute lean graphs.

Theorem 4. *Deciding if a graph is lean is coNP-complete.*

Proof. Recall that RDF graphs can encode standard graphs. Hence the proof is an encoding of the problem CORE:

Instance: A graph G

Question: Is there a homomorphism of G to a proper subgraph?

This problem was shown to be NP-complete by Hell and Nesetril [6].

From the above theorem it follows that finding minimal representations for graphs is hard.

3 Querying RDF databases

An RDF graph can be considered a standard relational database: a relation of triples with the attributes Subject, Predicate, and Object. The difference with standard relational databases is the presence of *blank nodes*, which stand for *anonymous* elements.

Thus, for us, an RDF *database* will be simply an RDF graph.

3.1 Query language

Let V be a set of variables (disjoint from UBL).

As query language, we will use the notion of *tableau* borrowed from the database literature (see for example [1]) but slightly extended to allow also a set of tuples in the head. A *tableau* is a pair (H, B) where H, B are RDF graphs over $V \cup \text{UBL}$ and all variables of H occur also in B . We often write a tableau in the form $H \leftarrow B$ to indicate the similarity with logic programming and Datalog.

For example, a tableau such as

$$(?Dept, \text{pays}, ?Instr) \leftarrow (?Instr, \text{lectures}, ?Course), (?Dept, \text{offers}, ?Course),$$

where identifiers preceded by ? are variables, intuitively defines the instructors paid by a department to be those who teach courses offered by the department.

Definition 4. *A query is a tableau (H, B) plus a set of constraints C , which is a subset of the variables occurring in H . In other words, a query is a triple (H, B, C) such that:*

1. H is an RDF graph over $\text{UBL} \cup V$, with $\text{var}(H) \subseteq \text{var}(B)$.
2. B is an RDF graph over $\text{UL} \cup V$. (i.e. has no blank nodes).
3. $C \subseteq \text{var}(H)$. (Constraints).

For example, the tableau above is a query with no constraints. We can add to it the constraint $\{?Instr\}$; intuitively, as we will formalize in the next subsection, this means that the $?Instr$ variable must be bound to a non-blank element in each answer to the query.

Note 4. The condition $var(H) \subseteq var(B)$ avoids the presence of free variables in the head of the query. The presence of blank nodes in the body of the query is unnecessary, because—as we will see—a variable plays exactly the same role in this position. However, we do allow blank nodes in the head of the query to support reification at the query level. (See Examples 7 and 8 in Section 3.3.) Finally, as shown in the example above, $C \subseteq var(H)$ will represent the set of variables in the query that we are forcing to be instantiated by constants.

Note 5. Blank nodes in the head of the query are technically free terms of the form $f(x_1, \dots, x_n)$, where x_1, \dots, x_k are variables occurring in the query, and f is a function symbol. Hence in our language, there is an arbitrary set of uninterpreted function symbols of different arities. We follow here the same approach as [8].

3.2 Answers to a query

Let D be a database, and V a set of variables.

A *valuation* is a function $v : V \rightarrow \text{UBL}$. For a set C of variables, the valuation v satisfies the constraint C (denoted $v \models C$) if for all $x \in C$, $v(x)$ is not blank.⁵

A *matching* of the graph B in database D is a valuation v such that an instance of $v(B)$ is a subgraph of D , i.e. such that $D \models v(B)$.

The matchings that interest us are those that satisfy the constraints C .

Definition 5. Let $q = (H, B, C)$ be a query and D a database. A pre-answer to q over D is the set

$$\text{preans}(q, D) = \{v(H) : v(B) \text{ is a matching in } D \text{ and } v \models C\}.$$

A graph $v(H)$ is called a *single answer* of the query q over D .

We can combine single answers in two different ways to obtain the answers to a query, leading to two main query semantics:

1. $\text{ans}_\cup(q, D)$ is the set-theoretic union of all single answers. With this approach, queries properly capture the information carried by blank nodes inside D (in particular when blank nodes play the role of bridges between two single answers).
2. An alternative approach, $\text{ans}_m(q, D)$, is to *merge* all single answers, which means to rename blank nodes if necessary to avoid name clashes.

Note that if there are no blank nodes in D , both approaches are the same and we are in the realm of classical databases.

⁵ This constraint is called a *must-bind variable* in DQL [15].

Proposition 1. *Let D, D' be databases, and q a query. Then for both semantics, if $D \models D'$ then $\text{ans}(q, D) \models \text{ans}(q, D')$.*

Proof. By hypothesis, there is a mapping μ such that $\mu(D')$ is a subgraph of D . It is enough to prove that every graph $G \in \text{preans}(q, D')$ is a subgraph of a graph in $\text{preans}(q, D)$. Let H the head and B the body of the query. Then $G = v(H)$ for a valuation v , with $v(B)$ a subgraph of D' . Then $\mu(v(B))$ is a subgraph of D , hence $\mu(G) = \mu(v(H)) \in \text{preans}(q, D)$.

Proposition 2. *For all queries q and databases D , $\text{ans}_{\cup}(q, D) \models \text{ans}_m(q, D)$.*

Proof. The statement follows from the fact that $G_1 \cup G_2 \models (G_1 \text{ merge } G_2)$. To check this last fact just consider the mapping from $(G_1 \text{ merge } G_2) \rightarrow G_1 \cup G_2$ that reverses the renaming of variables done in the merge.

Note 6. The converse of Proposition 2 does not hold. Consider the identity query q and the database $D = \{(X, b, c), (X, b, d)\}$. Then $\text{ans}_{\cup}(q, D) = D$ and $\text{ans}_m(q, D) = \{(X, b, c), (Y, b, d)\}$. Clearly there is no mapping from D to $\text{ans}_m(q, D)$.

In the sequel, unless stated otherwise, we will assume the union-semantics.

Example 4. Consider a database D which has a blank node B with several properties, i.e., there are in D several triples $(B, p_1, z_1), (B, p_2, z_3), \dots$. If we follow the merge-semantics, we cannot retrieve the properties of B with a data independent query. On the other hand, if we follow the union-semantics, the query $H = (X, \text{feature}, Y), B = (X, Y, Z), C = \emptyset$ will do it.

Example 5 (Identity query). With merge-semantics, there is no data-independent query that retrieves the whole database for all D 's. With the union-semantics, the query q defined by $H = (X, Y, Z), B = (X, Y, Z)$ and $C = \emptyset$ returns all triples in D , i.e. $\text{ans}(q, D) = D$.

Example 6. Consider an RDF database consisting of tuples of the following sort:

(Course, name, CourseName)
 (Lecturer, lectures, Course)
 (Lecturer, name, LecturerName)
 (Department, offers, Course)
 (Department, belongs, University)
 (University, located, Country)

In this database *name, lectures, offers, belongs, located* are RDF predicates defined in some ontology. Courses, Lecturers, Departments and Universities are URLs. CourseName is of type literal.

The predicates *belongs, worksAt* and *teachIn* which appear in the answers belong to another ontology.

1. Database courses in Canada. First consider the query defined by

$$\begin{aligned}
 H &= (?Department, \text{belongs}, ?University) \\
 B &= (?Course, \text{name}, \text{"Database"}), (?Department, \text{offers}, ?Course), \\
 &\quad (?Department, \text{belongs}, ?University), (?University, \text{located}, \text{"Canada"}) \\
 C &= \{?Department, ?University\}.
 \end{aligned}$$

This query returns all triples (Department, belongs, University), where Department belongs to University and offers a Database course, University is located in Canada, and enforcing that Department and University are not blank.

2. In what universities (and if known, what departments) does John Bassi lecture?

$$\begin{aligned}
 H &= (?Lecturer, \text{worksAt}, ?University), (?Lecturer, \text{teachIn}, ?Department) \\
 B &= (?Lecturer, \text{name}, \text{"John Bassi"}), (?Lecturer, \text{lectures}, ?Course), \\
 &\quad (?Department, \text{offers}, ?Course), (?Department, \text{belongs}, ?University) \\
 C &= \{?Lecturer, ?University\}.
 \end{aligned}$$

This query will return the name of all Universities at which John Bassi teaches. If the database contains information about the particular Department where Bassi teaches, it will be included. Otherwise, the Department will appear as a blank node in the answer.

Note 7 (Redundancy). We give some observations on redundancies in queries, databases and set of answers.

1. It is desirable to have queries with lean heads. Otherwise, the answer generated will have redundancies which could have been avoided.
2. It is not always possible to have lean graphs in body of queries. For example, consider the query $q = (H, B, \emptyset)$, where $H = (?Course, \text{related}, \text{"Database"})$ and $B = (?Department, \text{offers}, \text{"Database"}), (?Department, \text{offers}, ?Course)$. Clearly B is not lean and is equivalent to the lean graph $B' = (?Department, \text{offers}, \text{"Database"})$. It turns out that the query q cannot be reduced to one with body B' (see Note 9).
3. Even having lean databases and queries with lean heads and bodies does not avoid redundancies in the answer set. Consider the lean graph G_1 in Example 3, and the query $(Z, b, c) \leftarrow (Z, b, c)$. The answer set is $\{(X, b, c), (Y, b, c)\}$ which is not lean.

3.3 Reification

Now we will explore the previous concepts when some constant vocabulary is introduced. In this section we will restrict ourselves to a small subset of RDF's vocabulary description language, RDF Schema, which is an extension of RDF. It

provides mechanisms for describing groups of related resources and the relationships among these resources. We will be particularly interested in the vocabulary needed to do reification.

Consider the following statement:

$$\text{A triple } (a, b, c) \text{ is a } \textit{statement}. \quad (1)$$

To state this inside RDF one can say: “There is an object B that is a statement, whose subject is a , predicate is b , and object is c .” In order to write this down as a set of RDF statements, we need a vocabulary. We will use the following predicate constants:

```

rdf:statement
rdf:subject
rfd:predicate
rdf:object
rdf:type

```

They have a more or less self-explanatory semantics. For example, the triple $(a, \text{rdf:type}, c)$ means a is an instance of the class c . Using this vocabulary (whose formal definitions can be found in [12]), the sentence in (1) can be expressed as the following set of triples (when not needed, we will avoid the use of the namespace prefix `rdf`):

$$(B, \text{type}, \text{statement}), (B, \text{subject}, a), (B, \text{predicate}, b), (B, \text{object}, c)$$

This process is called *reification* of the statement. Our goal in this section is to study our query language extended with the vocabulary of reification in RDFSchema [12].

Note 8. In RDF semantics, statements are referred to by names, i.e. they are not by themselves objects. An implication of this is that from the existence of a triple (a, b, c) it does not follow that its reification also exists. Observe –using Theorem 2– that a triple does not entail its reification and its reification does not entail the triple. RDF follows the conservative approach that a statement is referred to, and there can be several such references, all distinct. Another alternative is to assume that the triple itself is an object of the universe. This is the approach of [14], where the advantages of such an approach are argued. From a database point of view, the current approach of RDF seems more adequate. With the current RDF semantics, an RDF specification, i.e. an RDF graph (database) is a *finite* set of objects, and answers to queries (as defined in this paper) are finite set of objects. However, if the triple itself is an object i_1 , then having (a, b, c) in a database D would imply that $(i_1, \text{subject}, a)$ is also a valid statement (and hence an object i_2), hence $(i_2, \text{subject}, i_1)$ is a valid statement, and so on.

Example 7. The query that reifies a triple (a, b, c) (and creates a blank node $f(a, b, c)$ to refer to it) is:

$$\begin{aligned} & (f(a, b, c), \text{type}, \text{statement}), (f(a, b, c), \text{subject}, a), \\ & (f(a, b, c), \text{predicate}, b), (f(a, b, c), \text{object}, c) \leftarrow (a, b, c). \end{aligned}$$

The answer to this query applied to a database D is exactly a reification of the triple (a, b, c) if it exists in the database D . Note that we need a Skolem function f to give a different blank node to each triple.

Example 8. A generalization of the previous example: the reification of all triples in the database D .

$$(f(X, Y, Z), \text{type}, \text{statement}), (f(X, Y, Z), \text{subject}, X), \\ (f(X, Y, Z), \text{predicate}, Y), (f(X, Y, Z), \text{object}, Z) \leftarrow (X, Y, Z).$$

Example 9. All properties of an object b (in the database to be queried):

$$(X, \text{type}, \text{property}) \leftarrow (b, X, Y)$$

It is interesting to note that now we can test Leibniz's identity on a database by just comparing the results of two queries. Recall Leibniz's identity Law: $a \equiv b$ if and only if for all properties $P(\cdot)$, $P(a)$ iff $P(b)$.

Example 10. (Following an example in Yang and Kifer [14]). All statements made by Encyclopedia Britannica are true. Note that we will need different queries depending on the structure of the database containing the information.

1. If we assume that Encyclopedia Britannica is a database containing all its statements (triples), the following query would do the work:

$$(f(X, Y, Z), \text{veracity}, \text{true}), (f(X, Y, Z), \text{type}, \text{statement}), \\ (f(X, Y, Z), \text{subject}, X), (f(X, Y, Z), \text{predicate}, Y), \\ (f(X, Y, Z), \text{object}, Z) \leftarrow (X, Y, Z).$$

2. If we assume that the statements of Encyclopedia Britannica are mixed in with several other statements from other sources (but already referred to as belonging to the E. Britannica) we need a query like:

$$(X, \text{veracity}, \text{true}) \leftarrow (X, \text{type}, \text{statement}), (X, \text{made}, \text{EncycB}).$$

3.4 The language DQL

We will discuss here aspects of DQL that are relevant from a database perspective and compare them with our approach.

1. DQL has a query pattern (our B), an answer pattern (our H) and a set of constraints very similar to our set of constraints C .
2. DQL has three sets of variables, "must-bind", "don't-bind" and "may-bind", which are a partition of the set of variables occurring in the query. Must-bind variables are those that must be bound in each answer. Don't-bind variables are those that must be not bound. In DQL this schema is oriented towards the type of answers the user is asking.

In our setting, "must-bind" variables correspond to the set of constraints C , "don't-bind" correspond to the the set $\text{var}(B) \setminus \text{var}(H)$ (they do not occur in the answer), and "may-bind" correspond to $\text{var}(H) \setminus C$.

3. In DQL the answer set is defined as the largest set of single answers that are entailed by the database such that no answer in the set is entailed by any other answer in the set.

We do not enforce this condition, due to complexity issues shown in Theorem 4. We think that redundancy cleaning should be an option, and in the next section study the implications of avoiding such redundancies.

4 Complexity issues

In this section we focus on the complexity of query answering. This process has three main components: computing matchings, minimization of queries, and redundancy elimination in answers.

4.1 Computing matchings

In order to understand the complexity of computing the set of matchings for a query over a database, we consider the simpler problem of testing emptiness of the query answer set. Following the usual database theory practice, we distinguish between *query complexity*, that is, evaluation time as a function of query size for a fixed database, and *data complexity*, evaluation time as a function of database size for a fixed query.

1. Query complexity version: For a fixed database D , given a query q , is $q(D)$ non-empty?
2. Data complexity version: For a fixed query q , given a database D , is $q(D)$ non-empty?

Theorem 5. *The evaluation problem is NP-complete for the query complexity version, and polynomial for the database complexity version.*

Proof. Reduction of 3SAT to the problem of evaluating a conjunctive query over a database. Membership in NP follows immediately.

Data complexity version: This follows from the fact that the number of potential matchings of the body of q in D is bounded by the number of subgraphs of D of size q .

From the proof it also follows that the size of the set of answers of a query q issued against a database D is bounded by $|D|^{|q|}$, where $|D|$ is the size of the database (number of triples) and $|q|$ is the number of symbols in the query.

Also note that reification does not play any relevant role in this, that is, even with reification the query language preserves the tractability of answers.

4.2 Minimization of queries

Since Theorem 5 implies that query evaluation is likely to be exponential in query size, static optimization of queries is an important goal. To perform this analysis, we apply techniques similar to classical tableau analysis [1].

A homomorphism $h : q' \rightarrow q$ is a substitution (of variables and blank nodes) θ such that $\theta(B') \subseteq B$ and $\theta(H') = H$ and $C' \subseteq C$. As usual, we define $q \subseteq q'$ if $\text{ans}(q, D) \subseteq \text{ans}(q', D)$ for all databases D .

Theorem 6. $q \subseteq q'$ if and only if there exists a homomorphism $h : q' \rightarrow q$.

Proof. Assume there exists a homomorphism $h : q' \rightarrow q$. Then $C' \subseteq C$ and using Theorem 2, for each valuation v , we have $v(B) \models v(\theta(B'))$. Then, for each database D , if $D \models v(B)$, then $D \models v(\theta(B'))$. Hence, each answer $v(H)$ of q will also be an answer $v(H')$ of q' (recall $\theta(H') = H$ and $C' \subseteq C$.)

Conversely, assume $q \subseteq q'$. Consider the database D_B obtained from B by replacing each variable x by a constant a_x . Let v the valuation assigning x to a_x . Then $v(H) \in \text{ans}(q, D_B) \subseteq \text{ans}(q', D_B)$. So, there is a valuation v' such that $D \models v'(B)$ and $v'(H) = v(H)$. Clearly $v = v'$ on the variables of H . Consider the substitution $\theta = v' \circ v^{-1}$. The condition $C' \subseteq C$ follows from $q \subseteq q'$.

We say that a query $q = (H, B, C)$ is *minimal* if there is no query $q' = (H', B', C')$ equivalent to q such that $|B'| < |B|$ (where $|X|$ means the size of the set).

Theorem 7. For each query $q = (H, B, C)$ there is a minimal query $q_m = (H, B_m, C_m)$ equivalent to q and $B_m \subseteq B$ and $C_m \subseteq C$.

Proof. Let $q' = (H', B', C')$ be a minimal query equivalent to q . Then there are homomorphisms $\theta_1 : q \rightarrow q'$ and $\theta_2 : q' \rightarrow q$. Consider $q_m = \theta_2 \theta_1(q)$.

Note 9. Observe that this minimization does not coincide exactly with the leanization of the body of the query, because the homomorphism that reduces query q to q_m poses another condition besides a mapping from body to body, namely that it must preserve heads. This is the reason why in Note 7, the query in item 2 cannot be further reduced.

Theorem 8. Let q, q' be two queries. The following problems are NP-complete.

1. Is $q \subseteq q'$?
2. Is $q \equiv q'$?

Proof. NP-hardness: coding of classical tableau. (Note that one relation with 3 attributes suffices.)

Membership in NP follows from noting that a witness is the homomorphism.

4.3 Redundancy elimination

Answers to queries in RDF usually have redundancies. Ideally, the answer set $\text{ans}(q, D)$ should reduce these redundancies to the minimum, i.e. to an equivalent lean graph.

Now we will apply these results to redundancy elimination in queries.

The naive approach to eliminate redundancy in answers is (1) to compute $\text{ans}(q, D)$, and (2) to compute a lean equivalent to $\text{ans}(q, D)$. Next theorem shows that in the worst case there is no better approach.

Theorem 9. *Given a lean database D and a query q , to decide whether $\text{ans}_\cup(q, D)$ is lean is coNP-complete (in the size of D).*

The Theorem directly follows the fact that there is a query that computes the identity and from Theorem 4.

For merge-semantics redundancy elimination can be done much more efficiently:

Theorem 10. *Given a lean database D and a query q , deciding whether $\text{ans}_m(q, D)$ is lean can be done in polynomial time in the size of D .*

Proof. Let $A = \text{ans}_m(q, D)$ and let us refer to mappings from single answers to A as *single mappings*.

The key observation is that, because single answers do not share variables in merge-semantics, mappings $\mu : A \rightarrow A$ are exactly unions of single mappings $\mu_j : G_j \rightarrow A$ for each G_j single answer. (Note that in the case of union-semantics the union of the μ_j would not be a function.)

Thus an algorithm for finding proper mapping $\mu : A \rightarrow A$ only needs to compute single mappings and check whether (1) at least a single mapping is proper, or (2) two of them share a blank node in their range. This can be done in time polynomial on the set of single mappings, which size is polynomial on the size of D . Thus the complete test can be done in polytime.

5 Conclusions and Future Work

RDF databases pose new challenges to query languages, which arise due to particularities of the RDF model, such as reification and blank nodes. This paper intends to provide conceptual insight into the problem of dealing with these new features in query languages. Our work also establishes theoretical foundations for further research in this area.

Blank nodes play a crucial role in the semantics of query answering, although they do not affect complexity bounds dramatically. In fact, the behaviour of blank nodes is at the heart of different interpretations, both in query languages and in the formal semantics of RDF itself. For example, the notions of *lean* and *proper instance* deserve further development.

The expressive power of reification in query languages needs further study. In particular, the proper fragment of logic in which RDF query languages must operate is still not well understood.

Our study brought forth the need to formalize richer properties and mechanisms of current working query languages for RDF. This formalization would establish a solid base to compare functionalities, features and limitations of these languages. For example, features like connectedness, reachability, paths, recursion, extended constraints, aggregation and views.

Acknowledgements C. Gutiérrez was partially funded by FONDECYT No 1030810. C. Hurtado was funded by Millenium Nucleus, Center for Web Research (P01-029-F), Mideplan.

References

1. S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases*, Addison-Wesley Publishing Co., 1995.
2. F. Baader, T. Nipkow, *Term Rewriting and All That*, Cambridge Univ. Press, 1998.
3. R. V. Guha, *rdfDB Query Language*, in <http://www.guha.com/rdfdb/query.html>
4. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, *RQL: A Declarative Query Language for RDF*, Proceedings WWW2002, Hawaii, USA, 2002.
5. L. Miller, A. Seaborne, A. Reggiori, *Three Implementations of SquishQL, a Simple RDF Query Language*, Proc. 1st. International Semantic Web Conference ISWC2002, Sardinia, Italy, 2002.
6. P. Hell, J. Neseřtil, *The core of a graph*, Discrete Math. 109 (1992), 117-126.
7. *RDF Query Exchange Language (QEL)*, Edit. M. Nilsson, W. Siberski, <http://edutella.jxta.org/spec/qel.html>
8. Y. Papakonstantinou, V. Vassalos, *Query Rewriting for Semistructured Data*, Proceedings of the ACM SIGMOD International Conference on Management of Data, Philadelphia, Pennsylvania, June 1999
9. M. Sintek, S. Decker, *TRIPLE—A Query, Inference, and Transformation Language for the Semantic Web*, Proc. International Semantic Web Conference (ISWC), Sardinia, June 2002.
10. *Resource description framework (RDF) model and syntax specification*, Edit. O. Lassila, R. Swick, Working draft, W3C, 1998.
11. *RDF Semantics, W3C Working Draft, 23 January 2003* Edit. Patrick Hayes
12. *RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft 23 January 2003*, Edit. Dan Brickley, R.V. Guha.
13. *RDF Concepts and Abstract Syntax*, Edit. G. Klyne, J. J. Carroll. W3C Working Draft 23 January 2003.
14. G. Yang, M. Kifer, *On the Semantics of Anonymous Identity and Reification*
15. *DAML Query Language (DQL), April 2003, Abstract Specification. DAML Joint Committee, R. Fikes, P. Hayes, I. Horrocks, Ed.*
16. *E. Prud'hommeaux, B. Grosz, RDF Query and Rules: A Framework and Survey*, <http://www.w3.org/2001/11/13-RDF-Query-Rules/>
17. A. Magkanaraki et al. *Ontology Storage and Querying, Technical Report No. 308, April 2002, Foundation for Research and Technology Hellas, Institute of Computer Science, Information System Laboratory.*

Event-Condition-Action Rule Languages for the Semantic Web

George Papamarkos, Alexandra Poulouvassilis, Peter T. Wood

School of Computer Science and Information Systems, Birkbeck College, University of London, London WC1E 7HX
email: {gpapa05,ap,ptw}@dcs.bbk.ac.uk

Abstract. The Semantic Web is based on XML and RDF as its fundamental standards for exchanging and storing information on the World Wide Web. Event-condition-action (ECA) rules are a natural candidate for supporting reactive functionality on XML or RDF repositories. In this paper we describe a language for ECA rules on XML and a prototype implementation of this language. We also discuss some preliminary ideas regarding a language for ECA rules operating on a graph/triple representation of RDF, and we describe the architecture of a distributed deployment of such RDF ECA rules.

1 Introduction

XML and RDF are becoming dominant standards for storing and exchanging information on the World Wide Web. With their increasing use in dynamic applications such as e-commerce and e-learning [9, 10, 14, 15, 1, 19, 16, 22], there is a need for the support of reactive functionality on XML and RDF repositories. Event-condition-action (ECA) rules are a natural candidate for this. ECA rules automatically perform actions in response to events provided that stated conditions hold. They allow an application's reactive functionality to be defined and managed within a single rule base rather than being encoded in diverse programs, thus enhancing the modularity and maintainability of the application. Also, ECA rules have a high-level, declarative syntax and so are amenable to analysis and optimisation techniques which could not be applied if the same functionality were expressed directly in programming language code.

ECA rules have been used in many settings, including active databases [25, 20], personalisation and publish/subscribe technology [4, 9, 10, 12, 21], and specifying and implementing business processes [3, 11, 15]. An ECA rule has the general syntax

on event if condition do actions

The event part specifies when the rule should be triggered, the condition part is a query which determines if the database is in particular state, and the action part states the actions to be performed automatically if the condition holds. Executing a rule's actions may in turn trigger further ECA rules, and

the rule execution proceeds until no more rules are triggered. Non-termination of rule execution is generally a possibility and thus much research has focussed on the development of static rule analysis techniques for detecting possibly non-terminating rule sets; a practical solution to this problem adopted by commercial DBMS is to set a predefined upper limit on the number of recursive rule firings, and to abort a transaction if this is exceeded. More details on the foundations of ECA rules in active databases, and descriptions of a range of implemented active database prototypes can be found in [25, 20].

We begin this paper with a brief review of related work on ECA rules for XML. We then describe our language for specifying ECA rules on XML repositories, and present a prototype implementation of it. This language can be used for RDF data which has been serialised as XML but we are also exploring ECA rule languages for RDF that will operate on a graph/triple representation. We present an archetypal such language and also an architecture for distributed deployment of such RDF ECA rules. Along the way, we discuss directions of further work for both languages.

The work reported here is part of the ongoing SeLeNe project, which is investigating techniques for managing RDF repositories of educational metadata, and providing syndication, personalisation and notification services over this metadata (see <http://www.dcs.bbk.ac.uk/selene>).

2 ECA Rules for XML

In recent work [7, 6], we specified a language for defining ECA rules on XML data, based on the XPath and XQuery standards. We also developed techniques for analysing the triggering and activation relationships between such rules¹ and showed how these techniques can be used to detect possibly non-terminating sets of ECA rules. A number of other ECA rule languages for XML have also been proposed, although none of this other work has focussed on analysing the behaviour of the ECA rules.

Reference [9] discusses extending XML repositories with ECA rules in order to support e-services. Active extensions to the XSLT [27] and Lorel [2] languages are proposed which handle insertion, deletion, and update events on XML documents. Reference [10] discusses a more specific application of the approach to push technology where rule actions are methods that cannot update the repository, and hence cannot trigger other rules.

Reference [8] also defines an ECA rule language for XML. The rule syntax is similar to ours, but the rule execution model is different. In our case we treat insertions or deletions of XML fragments as “atomic” updates and ECA rule execution is invoked only after the completion of such an update, whereas in [8] such updates are broken up into a sequence of finer granularity requests each

¹ A rule r_i *may trigger* a rule r_j if execution of the action of r_i may generate an event which triggers r_j . A rule r_i *may activate* another rule r_j if r_j 's condition may be changed from False to True after the execution of r_i 's action. A rule r_i *may activate* itself if its condition may be True after the execution of its action.

of which may invoke the ECA rule execution. In general, these semantics may produce different results for the same initial update.

ARML [13] provides an XML-based rule description for rule sharing among different heterogeneous ECA rule processing systems. In contrast to our language, conditions and actions are defined abstractly as XML-RPC methods which are later matched with system-specific methods.

GRML [24] is a multi-purpose rule markup language for defining integrity, derivation and ECA rules. GRML uses an abstract syntax based on RuleML, leaving the mapping to a real language for each underlying system implementation. GRML aims to provide semantics for defining access over distributed, heterogeneous data sources for rule evaluation and allows the user to declare most of the semantics necessary for processing a rule, and to evaluate events and conditions coming from heterogeneous data sources.

Finally, [23] proposes extensions to the XQuery language [28] to incorporate update operations. These are more expressive than the actions supported by our ECA rule language since they also include renaming and replacement operations, and specification of updates at multiple levels of documents. Triggers are discussed in [23] as an implementation mechanism for deletion operations on the underlying relational store of the XML. However, provision of ECA rules at the “logical” XML level is not considered.

2.1 Our XML ECA Rule Language

An XML repository consists of a set of XML documents. In our XML ECA rule language, we use XPath [26] and XQuery [28] to specify the event, condition and actions parts of rules. XPath is used for selecting and matching fragments of XML documents within the event and condition parts while XQuery is used within insertion actions, where there is a need to be able to construct new XML fragments.

The event part of an XML ECA rule is an expression of one of the following two forms:

INSERT e
DELETE e

where e is an XPath expression which evaluates to a set of nodes. The rule is *triggered* if this set of nodes includes any node in a new XML fragment, in the case of an insertion, or in a deleted fragment, in the case of a deletion.

The system-defined variable `$delta` is available for use within the condition and actions parts of the rule, and its set of instantiations is the set of new or deleted nodes returned by e .

The condition part of a rule is either the constant `TRUE`, or one or more XPath expressions connected by the boolean connectives `and`, `or`, `not`. The rule *fires* if it is triggered and its condition evaluates to true.

The actions part of a rule is a sequence of one or more actions:

$action_1; \dots; action_n$

where each $action_i$ is an expression of one of the following three forms:

INSERT r BELOW e BEFORE q

INSERT r BELOW e AFTER q
DELETE e

Here, r is an XQuery expression, e is an XPath expression and q is either the constant TRUE or an XPath qualifier.

In an INSERT action, the expression e specifies the set of nodes, N , immediately below which new XML fragment(s) will be inserted. These fragments are specified by the expression r . If e or r references the $\$delta$ variable, then one XML fragment is constructed for each instantiation of $\$delta$ for which the rule's condition evaluates to True. If neither e nor r references $\$delta$, then a single fragment is constructed. The expression q is an XPath qualifier which is evaluated on each child of each node $n \in N$. For insertions of the form AFTER q , the new fragment(s) are inserted after the last sibling for which q is True, while for insertions of the form BEFORE q , the new fragment(s) are inserted before the first sibling for which q is True. The order in which new fragments are inserted is non-deterministic.

In a DELETE action, the expression e specifies the set of nodes which will be deleted (together with their descendant nodes). Again, e may reference the $\$delta$ variable.

Example 1. Consider an XML repository containing metadata about learning objects (LOs) available on the web, as well as personal metadata about users of these LOs. The XML document `los.xml` contains information about the LOs, and we show below some of the information held for a particular book, "Data On the Web". Under `annotations`, a new `review` is appended every time a user submits a review of the book.

```
<LOs>
..
<LO type="book" title="Data On the Web">
  <subject>Computer Science</subject>
  <creator>S. Abiteboul</creator>
  <creator>P. Buneman</creator>
  <creator>D. Suciu</creator>
  <description>From Relations to Semistructured data and XML
</description>
  <publisher>Morgan Kaufmann</publisher>
  <isbn>1-55860-621-Y</isbn>
  <annotations>
    <review>
      <reviewer>Teacher Education Review Panel</reviewer>
      <date>2002-10-20</date>
      <rating>9</rating>
      <description>
        This book gives a comprehensive, state-of-the art
        discussion of data models, query languages and ...
      </description>
    </review>
  </annotations>
</LO>
</LOs>
```

```

    <review>
      <reviewer>John Smith</reviewer>
      <date>2002-12-20</date>
      <rating>10</rating>
      <description>
        I found this a great book to learn about querying
        semi-structured data, which I didn't know much about.
      </description>
    </review>
  </annotations>
</L0>
...
</L0s>

```

The XML document `users.xml` contains information about users, and we show below some of the information held for a particular user “Johnny Mnemonic”. Users can subscribe to be notified of the latest review submitted for books in subjects that they are interested in, and this information is used to automatically update their personal metadata:

```

<users>
  ...
  <user id="217">
    <name>Johnny Mnemonic</name>
    <profession>student</profession>
    <subjects>
      <subject>Computer Science</subject>
      <subject>Mathematics</subject>
      <subject>Economics</subject>
    </subjects>
    <L0s>
      <L0 type="book" title="Data On the Web">
        <isbn>1-55860-621-Y</isbn>
        <latest-review>
          <reviewer>John Smith</reviewer>
          <date>2002-12-20</date>
          <rating>10</rating>
          <description>
            I found this a great book to learn about querying
            semi-structured data, which I didn't know much about.
          </description>
        </latest-review>
      </L0>
      ...
    </L0s>
  </user> ...
</users>

```

Johnny Mnemonic is interested in “Computer Science” and the following rule replaces the current latest review (if there is one) of any Computer Science book in his personal metadata by a new review of that book:

```
ON INSERT document('los.xml')/LOs/LO/annotations/review
IF $delta/../../subject[.='Computer Science']
DO DELETE document('users.xml')/users/user[@id="217"]/LOs/LO
    [isbn=$delta/../../isbn]/latest-review;
    INSERT <latest-review>{$delta/*}</latest-review>
BELOW document('users.xml')/users/user[@id="217"]/LOs/
    LO[isbn=$delta/../../isbn]
AFTER isbn
```

Here, the system-defined `$delta` variable is bound to a newly inserted `review` node detected by the event part of the rule. The rule’s condition checks that the subject of the book in question is Computer Science. The rule’s action then deletes the existing latest review for this book within Johnny Mnemonic’s metadata (if there is one) and inserts the new review.

Suppose now that the following update occurs, appending a new review for the “Data On the Web” book:

```
INSERT <review>
    <reviewer>Neo Anderson</reviewer>
    <date>2003-04-29</date>
    <rating>9</rating>
    <description>
        Very clearly written and very well-organised.
        Describes in detail all the ...
    </description>
</review>
BELOW document('los.xml')/LOs/
    LO[isbn="1-55860-621-Y"]/annotations
AFTER TRUE
```

This update triggers the rule above, causing the replacement within Johnny Mnemonic’s personal metadata of the previous review submitted by John Smith by the new review submitted by Neo Anderson.

As another example rule, the following rule removes the current latest review (if there is one) of a Computer Science book in Johnny Mnemonic’s personal metadata if this review is removed from the list of reviews for this book (this rule assumes that each reviewer reviews a book only once):

```
ON DELETE document('los.xml')/LOs/LO/annotations/review
IF $delta/../../subject[.='Computer Science']
DO DELETE document('users.xml')/users/user[@id="217"]/LOs/
    LO[isbn=$delta/../../isbn]/
    latest-review[reviewer=$delta/reviewer]
```


We refer the reader to [7, 6] for a more detailed discussion of the syntax and semantics of our XML ECA rule language. Here, we next describe a prototype implementation.

2.2 A Prototype Implementation

Due to the current immaturity of existing XML repository products in supporting a sufficiently expressive update language, for this first prototype implementation we have used flat files and have exploited the functionality provided by the W3C DOM standard [29] for interacting with them. The architecture of our system is illustrated in Figure 1.

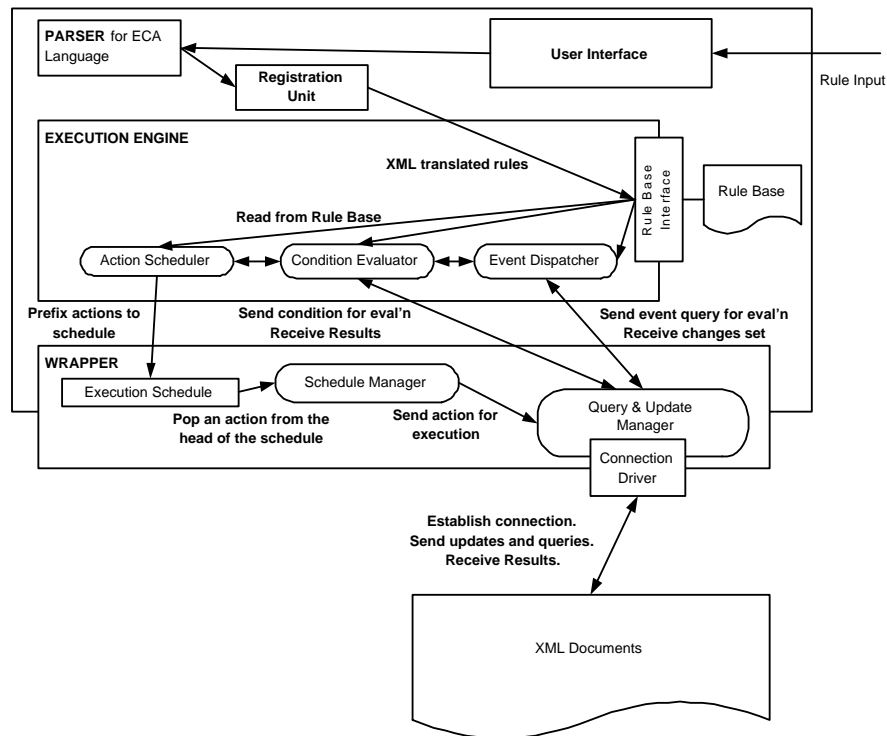


Fig. 1. ECA Engine Architecture

The *Parser* parses and checks the syntactic validity of a new rule. For construction of the parser, we have used the JavaCC lexer-parser generator. Valid rules are translated into an XML form and are added by the *Registration Unit* to the *Rule Base* (which is an XML file). Details about each rule are stored here, including its name, priority, and event, condition and action parts.

The *Execution Engine* encapsulates the rule processing functionality. In particular, the *Event Dispatcher*, *Condition Evaluator* and *Action Scheduler* implement these aspects of the rule processing, as we describe in more detail below. All of these components interface with the Wrapper in order to send and receive data to and from the underlying XML files.

The *Execution Schedule* contains a sequence of *updates* — these have the same syntax as rule actions except that they do not contain any `$delta` expressions within them. By “`$delta` expression” we mean an XPath expression (either stand-alone or possibly nested within another XPath expression) that starts with `$delta`. These portions of a rule’s action part are replaced by the result of evaluating the expressions on the current document — see below.

The *Wrapper* interfaces with the XML files on disk. All update and query requests from the upper levels of the system pass through this component, which coordinates them. It undertakes to open files, submit queries and updates, and receive back results from them. The Wrapper performs these services by using the functionality of the Apache Xalan API. All queries are performed directly by using XPath. For deletions, we identify the set of nodes that will be deleted by using the XPath expression within the DELETE part of the request, and we then remove all the subdocuments rooted at the nodes identified. For insertions, we identify the set of nodes that will be affected by using the XPath expression within the BELOW part of the request, and we then add the fragment specified within the INSERT part as a new child of each of the nodes identified, placing it relative to the existing children according to the AFTER or BEFORE qualifier.

Rule execution begins with a request from the *Schedule Manager* to the *Query & Update Manager* to execute the update currently at the head of the schedule. In case of an insertion, the Query & Update Manager executes the update and annotates the newly inserted nodes, while in the case of a deletion it annotates the nodes to be deleted without executing the deletion yet².

Following the execution of the update, control then passes to the *Event Dispatcher*. This requests the Query & Update Manager to evaluate the XPath query of the event part of each rule that may be triggered by the update that was just executed. For each rule whose query result set contains annotated nodes (either newly inserted or about-to-be deleted), the Event Dispatcher creates a **changes** set containing these annotated nodes, and the rule is triggered.

The *Condition Evaluator* then requests the Query & Update Manager to evaluate the condition part of each triggered rule on the affected document, using as the evaluation context either the root node if there are no occurrences of `$delta` within a query, or each instance of the **changes** set otherwise. The rule’s **delta** set is thus created, consisting of those members of its **changes** set for which the condition evaluates to true. If the **delta** set is non-empty, the rule fires and control is passed to the Action Scheduler to further process the rule. Otherwise, processing of this rule ends.

² The annotation of nodes is performed using non-DOM methods provided by Apache Xerces API that allow us to attach data to XML nodes without affecting the physical representation of the file.

The *Action Scheduler* reformulates a given rule's action(s) in order to eliminate any instances of `$delta` expressions within them. The reformulation algorithm performs the following steps for each node within the rule's `delta` set:

- Replaces the `$delta` variable in each of the `$delta` expressions by the current node of the `delta` set.
- Evaluates each of the modified `$delta` expressions with respect to the updated document.
- Replaces each `$delta` expression within the rule's action(s) by the corresponding result of the previous step.

The outcome of this reformulation is that one instance of the rule's action(s) is created for each node in the rule's `delta` set. These updates are now prefixed, in an arbitrary order, to the front of the schedule — this is known as Immediate scheduling, although other alternatives are also possible (see [20]). If multiple rules have fired as a result of the last update executed, then the updates that result from their actions are prefixed the schedule in order of the rules' specified priorities. Control then passes once more to the Schedule Manager and the cycle repeats. If the last update executed by the Query & Update Manager was a DELETE, then before control passes back to the Schedule Manager, the actual deletion of the annotated nodes is first performed.

2.3 Future Work

There is as yet no accepted standard update language for XML. If ECA rules are to be supported on XML repositories, then whatever standard eventually emerges, there is also the parallel issue of designing the event language to match up with this update language. Here we have seen how this was done in the context of our particular update language for XML. Elsewhere [6] it is shown how triggering and activation relationships can be detected for our particular XML ECA rules. In general, the ability to analyse and optimise ECA rules needs to be balanced against their complexity and expressiveness, and this issue also needs to be borne in mind in future developments in ECA rule languages for XML, and indeed for RDF.

It would be straightforward to extend our language to also support REPLACE events and actions, where the former would have the syntax

REPLACE *e*

and the latter the syntax

REPLACE *e* BY *r*

meaning that the set of nodes identified by *e* (and their subdocuments, if any) should be replaced by *r*. For example, the pair of actions in the first rule in Example 1 could be replaced by the single action

```
REPLACE document('users.xml')/users/user[@id="217"]/LOs/LO
      [isbn=$delta/../../isbn]/latest-review
BY    <latest-review>{$delta/*}</latest-review>
```

In general, our INSERT actions may result in non-determinism in the order in which a set of new fragments are inserted under a common parent, since the BEFORE and AFTER constructs only specify the ordering of new fragments with respect to the existing document content. It is an area of further work to extend our XML ECA language to capture ordering relationships between new fragments being inserted into a document.

At present we assume Immediate scheduling of rules that have fired, though it would be straightforward to also allow rules with other scheduling modes. However, the practical applicability and performance implications of these extensions is an area that requires further investigation.

Another important area is combining ECA rules with transactions and consistency maintenance in XML repositories.

3 ECA rules for RDF

The above language can be used for RDF which has been serialised as XML. However, we are also exploring ECA rule languages for RDF that will operate directly on a graph/triple representation. In our archetypal RDF ECA rule language, the event part of a rule is an expression of one of the following two forms:

```
INSERT e
DELETE e
```

where *e* is a path expression which again evaluates to a set of nodes.

The rule is triggered if this set includes any new node, in the case of an insertion, or any deleted node, in the case of a deletion. The system-defined variable `$delta` is again available for use within the condition and actions parts of the rule, and its set of instantiations is the set of new or deleted nodes returned by *e*.

The condition part of a rule is a query which may reference the `$delta` variable. Analogously to our XML ECA rule language, condition queries consist of conjunctions, disjunctions and negations of path expressions.

The actions part of a rule is a sequence of one or more actions, where each action is of one of the following two forms:

```
[let-expressions IN] INSERT triples
[let-expressions IN] DELETE triples
```

Here, *let-expressions* is an optional set of local variable definitions of the form `let variable = e`, where *e* is a path expression, and *triples* is a set of triples of the form (*subject, predicate, object*).

Example 2. Consider the two RDF graphs illustrated in Figures 2 and 3. Based on the application described in Example 1, the first shows the metadata relating to the “Data on the Web” book, while the second shows the personal metadata relating to user 128.

Suppose that user 128 wants to keep his set of reviews of Computer Science books up-to-date. If a new review of a Computer Science book is inserted, then the following ECA rule adds a new arc linking the new review into user 128’s personal metadata:

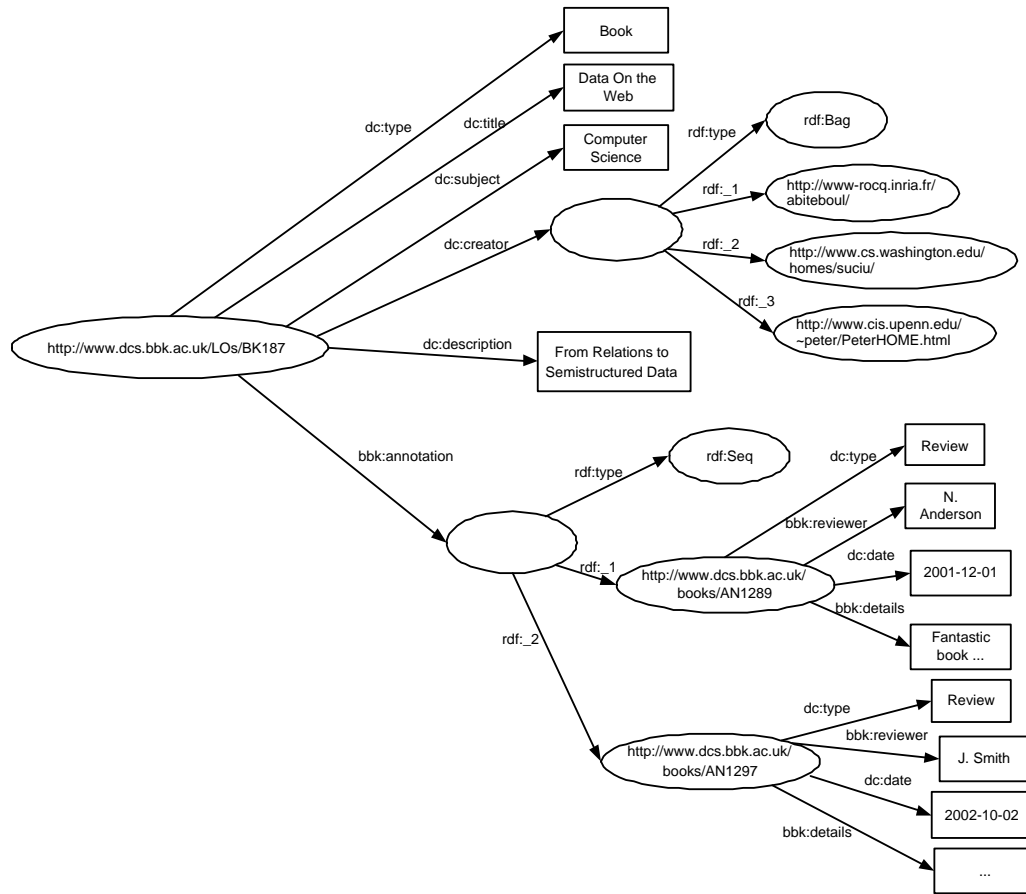


Fig. 2. Learning Object Metadata

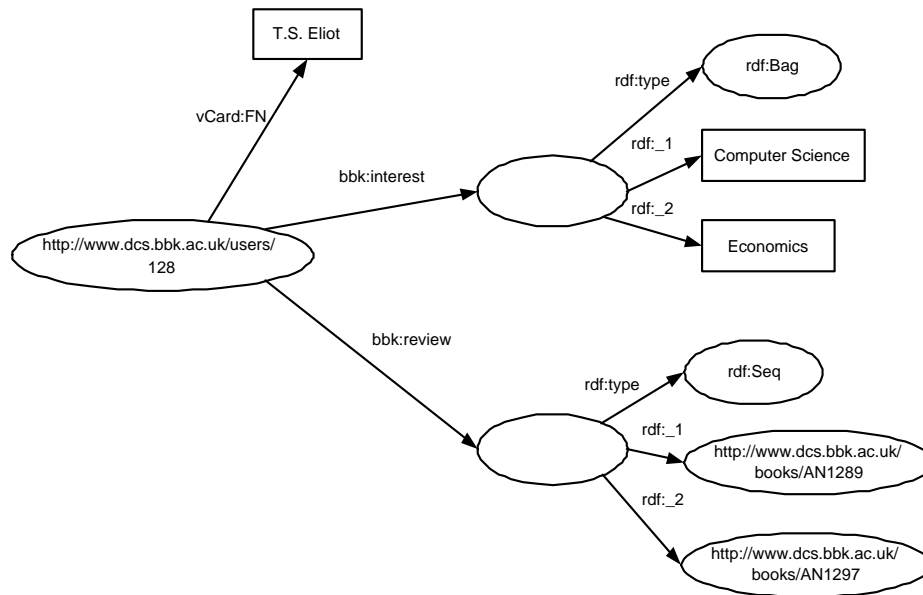


Fig. 3. User Metadata

```

ON INSERT resource() [child(dc:type)='Book']/child(bbk:annotation)/
    element() [child(dc:type)='Review']
IF $delta/parent()/parent() [child(dc:subject) = 'Computer Science']
DO LET $reviews = resource('http://www.dcs.bbk.ac.uk/users/128')
    /child(bbk:review) IN
    INSERT ($reviews,seq++, $delta)

```

Here, the event part of the rule checks whether a new review has been added for a book (expressed in the syntax of RDFPath [17]). The condition part checks if the new review is for a Computer Science book. If so, the action part inserts the new arc between user 128's reviews collection and the new review (we use the syntax `seq++` to indicate an increment in the collection's element count).

As another example, if user 128 removes one of his interests, then the following rule removes from his personal metadata all arcs to reviews of learning objects on that subject:

```

ON DELETE resource('http://www.dcs.bbk.ac.uk/users/128')
    /child(bbk:interest)/element()
IF TRUE
DO LET $reviews = resource('http://www.dcs.bbk.ac.uk/users/128')
    /child(bbk:review);
    $review = $reviews/element() [parent()/parent(bbk:annotation)
    [child(dc:subject) = $delta]] IN
DELETE ($reviews,seq?, $review)

```

Here, the event part checks if an interest of user 128 has been deleted. The condition part always holds. The LET part of the rule's action defines `$reviews` to be user 128's reviews collection and defines `$review` to be those reviews which relate to learning objects whose subject is the same as the deleted interest. Finally, the DELETE part will generate one triple to be deleted for each pair of distinct values of `$reviews` and `$review` (we use the syntax `seq?` to match any order of the review being deleted within the collection).

3.1 Future Work

There is as yet no standard query/update language for RDF and hence our RDF ECA language is even more prototypical than our XML ECA language. Some of the observations we made in Section 2.3 regarding the XML ECA language also apply here, namely the need to match up the event sub-language with the update sub-language, the need to balance expressiveness of ECA rules against the ability to analyze and optimize them, the possibility of a variety of scheduling modes beyond Immediate rule scheduling, and combining ECA rules with transactions and consistency maintenance in RDF repositories.

For the immediate future, we plan to:

- define formally the event, condition and action sub-languages of our RDF ECA rules;
- define the API requirements for the support of such rules over SeLeNe's RDF repository (likely to be FORTH's RDFSuite [5]);
- implement and experiment with the language, using as a testbed SeLeNe's educational metadata.

4 ECA Rules in a Distributed Environment

Beyond the centralized version of our system, we plan to develop a distributed version supporting ECA rules on distributed RDF repositories, as part of the ongoing SeLeNe project (<http://www.dcs.bbk.ac.uk/selene>). This project is investigating the technical requirements, and possible technical solutions, for 'self e-learning networks', where a self e-learning network is a distributed repository of metadata relating to learning objects (LOs) accessed by users wishing to publish or use such LOs. A self e-learning network (SeLeNe) will have a peer-to-peer topology, with facilities for peers to join or leave the network. Each peer will manage a fragment of the overall distributed metadata. This metadata will be expressed in RDF, and will contain information about learning objects and about the users of the SeLeNe (see [16]). Support of such networks will require:

- techniques for reconciliation and integration of heterogeneous metadata;
- definition of personalised views over this distributed metadata resource;
- detection, notification and propagation of changes to the metadata descriptions.

These requirements have a good fit with the functionality that could potentially be provided by ECA rules, and the architecture that we envisage is illustrated in Figure 4. Each ‘peer’ shown in that diagram is actually a ‘super-peer’ (SP) which may be coordinating a group of further peers (not shown in the figure).

At each SP there is installed one local ECA Engine, which has the same features and components as the centralized architecture discussed in Section 2.2 above and illustrated in Figure 1. One possibility is that each local ECA Engine will operate as a Web Service and that the communication between them can be via XML messages (e.g. SOAP).

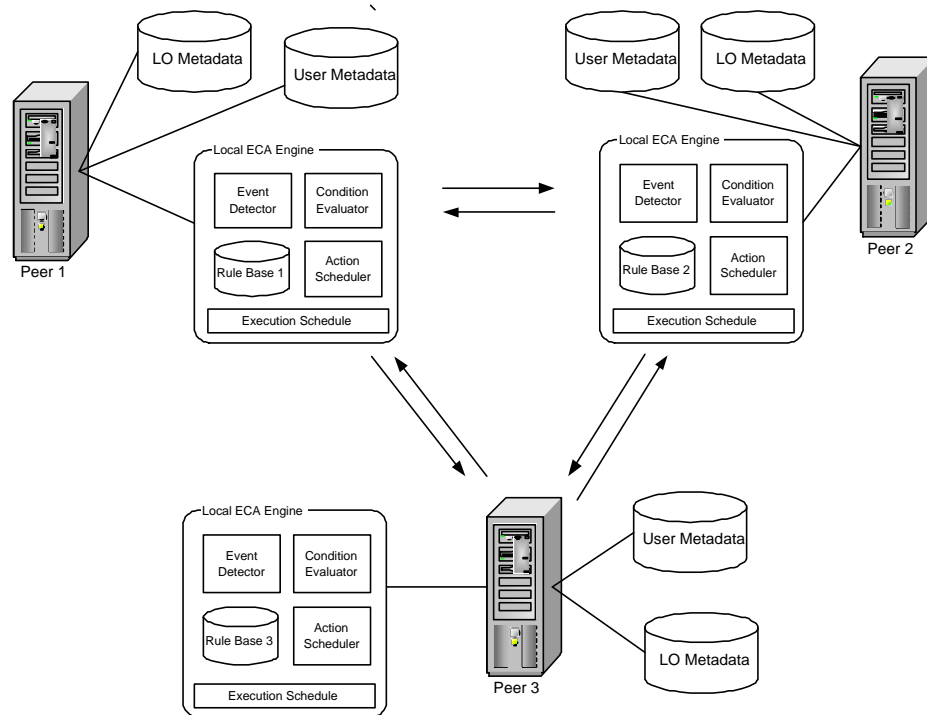


Fig. 4. Distributed System Architecture

Whenever a new ECA rule r is registered at a peer P , it will be sent to P 's SP for storage. As we will see below, from there r will also be sent to all other SPs, and a replica of it will be stored at those SPs that are *relevant* to r i.e. where an event may occur that may trigger r 's event part, or which may participate in evaluating r 's condition part, or where r 's actions may have to be scheduled for execution. At present, we assume that individual events and actions will occur at a single peer (which is likely to be the case in SeLeNe) although condition evaluation may be distributed.

Indexing at Peers and Super-Peers In order to determine whether an SP is relevant for a rule, an index can be kept at each peer and super-peer. There are a number of possibilities for doing this and we indicate here one solution:

As the RDF descriptions stored at each peer change over time, so each peer maintains an annotated copy of its local RDF Schema, which shows for each node in the schema whether or not there is RDF data of this type at this peer (a '0' or '1' bit).

This information is also propagated to the peer's coordinating SP. This SP maintains a combined RDF Schema which is annotated so that each node shows the set of peers in its own peer group that manage data of this type (a set of peer IDs), and also the remote SPs whose peer group manages such data (a set of SP IDs).

The latter information is gathered and maintained as follows: if a node in the RDF Schema of an SP changes from not having any data in this peer group to having data, or vice versa, this change is notified to all other SPs so that these can update the relevant annotation in their RDF Schemas. Note that in general the SPs may hold heterogeneous RDF Schemas, so there will need to be an RDF Schema translation service between SPs (as is indeed envisaged for SeLeNe).

Finally, as well as this annotated RDF Schema, each SP also keeps for each node annotated with a '1' in its RDF Schema a list of the RDF resources of this type that each peer in its peer group references — we call these lists of RDF resources *resource indexes*.

Comparison with related approaches: Querying and indexing data in a distributed RDF-based P2P network is more complex than for distributed structured databases. In the latter, the database servers and the database schema at each of them is known and fixed whereas in the former peers may dynamically join or leave the network and may manage data conforming to varying schema fragments. Schema-based routing indexes have been proposed to address this problem in Edutella [18]. Edutella uses two kinds of routing indexes: Super-Peer/Peer (SP/P RI) and Super-Peer/Super-Peer (SP/SP RI).

An SP/P RI stores information about metadata usage in each peer in its peer group. This includes information such as the schemas (e.g., `dc` or `lom`) or properties (e.g., `dc:subject`) used, as well as possibly conventional indexes on property values. When a peer registers with a SP, it provides the SP with its metadata usage, a process called advertisement. The peer undertakes to keep this advertisement up-to-date by informing its SP each time that a change affecting the advertised metadata takes place. At each super-peer, query fragments are matched against the SP/P RIs in order to determine peers that are relevant to this query (although this gives no guarantee that the returned result set from a peer is not empty). A similar approach is used in SP/SP RIs, but at a higher level of granularity and possibly only representing approximations of the information regarding their peers. A further difference to the SP/P RI is that an SP/SP RI contains information only about its neighbouring SPs in the SP topology. Update of SP/SP RIs is again based on broadcast messages sent between SPs.

For our purposes, we want to maintain more precise information about where various forms of metadata reside in the network and, as far as possible, do not want unnecessary routing of queries and updates to peers and super-peers that are not relevant. Hence, we have adopted the approach of using annotations on a full RDF schema and also resource indexes. The scalability of our proposal, however, still needs to be investigated.

Registering an ECA rule When a new rule is generated at a peer, it is sent to the peer's own SP for storage in its local rule base. The SP annotates the event, condition and action parts of the rule with the local peers that are relevant to each part (a list of peer IDs).

This can be determined by matching each part of the rule against the SP's annotated RDF Schema and/or its resource indexes — the former is useful if no resource is specified in this part of the rule and the latter is useful if a resource is specified. As the annotated RDF Schema and resource indexes at the SP evolve, so the annotations on the ECA rules can also be evolved to maintain consistency.

The rule is also sent to all other SPs that may be relevant to it — this is determined from the SP ID annotations on the originating SP's RDF Schema. These SPs repeat the above process of matching each part of the rule against their own annotated RDF Schema, and storing the resulting annotated rule in their own rule base if it is indeed relevant to any of their peer group. Note that, due to schema heterogeneity, the rule may first have to be translated so that its parts are expressed with respect to the local RDF Schema.

The final result is a replica of the rule at each SP which is relevant to the rule, annotated with local information about which peers may be affected by each part of the rule.

As the information at SPs changes with time, it may be that an ECA rule is no longer relevant to that SP, in which case the rule can be deleted from the SP's local rule base. Conversely, an ECA rule stored somewhere else may become relevant to an SP. This can be handled as follows:

Since all SPs know what kinds of data is stored at all other SPs, if any SP, SP1, is notified of a change in status of another SP, SP2, from not having data associated with a particular RDF Schema node to having such data, then SP1 will send SP2 a copy of any ECA rules that originated from SP1 and that may now have become relevant to SP2.

Rule triggering and execution At run-time, whenever an event E occurs at a peer P, it will notify its SP. This will determine whether E may trigger any ECA rule annotated with P's ID. If a rule r might have been triggered, the SP will send P r 's event query to evaluate.

If r has indeed been triggered, its condition will need to be evaluated, after generating an instantiation of it for each value of the `$delta` variable if this is present in the condition. The annotations on r can be used to determine to which local peers and other SPs sub-queries of the condition should be dispatched for evaluation. If the `$delta` variable is present in the condition, it will have

been instantiated and so we also consult the SPs' resource indexes for more precise information about which local peers are relevant to sub-queries of the instantiated condition.

If a condition evaluates to true, each corresponding rule action will be sent to, and scheduled, by the SPs that will execute it. Again this can be determined by the annotations on the rule action and consulting the SPs' resource indexes.

4.1 Future Work

There are several open issues remaining in realising the P2P ECA architecture we describe above:

- developing algorithms for matching rule event, condition and action parts with the schema-based indexes;
- defining the syntax of messages that will be passed between peers for distributed processing of ECA rules;
- defining the coordination with SeLeNe's distributed query processor for the evaluation of rule conditions;
- defining the coordination with SeLeNe's mediation functionality, for translating data and rules between heterogeneous schemas;
- more generally, mapping this distributed ECA functionality onto SeLeNe's service-based architecture;
- exploring distributed transactional aspects of the ECA rules (even though we assume that individual events and actions will occur at a single peer, the execution of an ECA rule may trigger another ECA rule and this whole cascade of rule firings may need to have the semantics of a single transaction).

5 Conclusions

In this paper we have discussed the provision of ECA rules for XML and RDF repositories, and have highlighted some of the new issues that arise in the context of such data. We have described a language for ECA rules on XML, and some preliminary ideas regarding a language for ECA rules on a graph/triple representation of RDF. We have described a prototype centralised implementation of the XML ECA rule language, and the architecture of a distributed implementation of the latter. For future work there are several directions to explore, as highlighted in Sections 2.3, 3.1 and 4.1 above.

An important issue is to evaluate the applicability and scalability of our languages, their execution models, and implementation. For this, we plan to deploy them for providing reactive functionality on distributed RDF repositories of educational metadata, as part of the ongoing SeLeNe project. This will also provide an opportunity to assess the impact of moving from a centralised to a distributed environment, with the additional challenges of network delay, network reliability, synchronisation of rule execution, maintaining consistency of the distributed metadata resource, tolerance of delays and failures etc.

References

1. S. Abiteboul, S. Cluet, G. Ferran, and M.-C. Rousset. The Xyleme project. *Computer Networks*, 39:225–238, 2002.
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L. Wiener. The Lorel query language for semistructured data. *VLDB Journal*, 1(1):68–88, 1997.
3. S. Abiteboul, V. Vianu, B. S. Fordham, and Y. Yesha. Relational transducers for electronic commerce. *JCSS*, 61(2):236–269, 2000.
4. A. Adi, D. Botzer, O. Etzion, and T. Yatzkar-Haham. Push technology personalization through event correlation. In *Proc 26th Int. Conf. on Very Large Databases*, pages 643–645, 2000.
5. S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *Proc. 2nd. Int. Workshop on the Semantic Web (SemWeb 2001)*, 2001.
6. J. Bailey, A. Poulouvasilis, and P.T. Wood. An Event-Condition-Action Language for XML. In *Proc. WWW'2002*, Hawaii, 2002.
7. J. Bailey, A. Poulouvasilis, and P.T. Wood. Analysis and optimisation for event-condition-action rules on XML. *Computer Networks*, 39:239–259, 2002.
8. A. Bonifati, D. Braga, A. Campi, and S. Ceri. Active XQuery. In *Proc. of the IEEE Conference on Data Engineering (ICDE)*, 2002.
9. A. Bonifati, S. Ceri, and S. Paraboschi. Active rules for XML: A new paradigm for e-services. *VLDB Journal*, 10(1):39–47, 2001.
10. A. Bonifati, S. Ceri, and S. Paraboschi. Pushing reactive services to XML repositories using active rules. In *WWW'01*, 2001.
11. S. Ceri and P. Fraternali. *Designing Database Applications with Objects and Rules: The IDEA Methodology*. Addison-Wesley, 1997.
12. S. Ceri, P. Fraternali, and S. Paraboschi. Data-driven one-to-one web site generation for data-intensive applications. In *Proc. 25th Int. Conf. on Very Large Databases*, pages 615–626, 1999.
13. E. Cho, I. Park, S. J. Hyum, and M. Kim. ARML: an active rule mark-up language for heterogeneous active information systems. In *Proc. RuleML 2002*, Sardinia, June 2002.
14. S. Cluet, P. Veltri, and D. Vodislav. Views in a large scale XML repository. In *Proc. 27th Int. Conf. on Very Large Databases*, pages 271–280, 2001.
15. H. Ishikawa and M. Ohta. An active web-based distributed database system for e-commerce. In *Proc. Web Dynamics Workshop, London*, 2001. <http://www.dcs.bbk.ac.uk/webDyn/>.
16. K. Keenoy *et al.* Self e-Learning Networks — Functionality, User Requirements and Exploitation Scenarios. See <http://www.dcs.bbk.ac.uk/selene/reports/Del122.pdf>, August 2003. SeLeNe Project Deliverable 2.2.
17. S. Kokkelink. Transforming RDF with RDFPath. See zoe.mathematik.uni-osnabrueck.de/QAT/Transform/RDFTransform.pdf, March 2001.
18. W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proc. WWW2003*, pages 536–543, 2003.
19. W. Nejdl *et al.* EDUTELLA: A P2P Networking Infrastructure Based on RDF. In *Proc. WWW'2002*, 2002.
20. N. Paton, editor. *Active Rules in Database Systems*. Springer-Verlag, 1999.

21. J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/subscribe systems. In *Proc 7th Int. Conf. on Cooperative Information Systems (CoopIS'2000)*, pages 162–173, 2000.
22. B. Simon *et al.* Smart space for learning: A mediation infrastructure for learning services. In *Proc. WWW'2003*, 2003.
23. I. Tatarinov, Z. G. Ives, A. Y. Halevy, and D. S. Weld. Updating XML. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 413–424, 2001.
24. G. Wagner. How to Design a General Rule Markup Language? In *Invited talk at the Workshop XML Technologien für das Semantic Web (XSW 2002)*, Berlin, June 2002.
25. J. Widom and S. Ceri. *Active Database Systems*. Morgan-Kaufmann, San Mateo, California, 1995.
26. World Wide Web Consortium. XML Path Language (XPath), Version 1.0. See <http://www.w3.org/TR/xpath>, November 1999. W3C Recommendation.
27. World Wide Web Consortium. XSL Transformations (XSLT), Version 1.0. See <http://www.w3.org/TR/xslt>, November 1999. W3C Recommendation.
28. World Wide Web Consortium. XQuery 1.0: An XML Query Language. See <http://www.w3.org/TR/xquery>, November 2002. W3C Working Draft.
29. World Wide Web Consortium. Document Object Model (DOM) Level 3 Core Specification. See <http://www.w3.org/TR/DOM-Level-3-Core/>, February 2003. W3C Working Draft.

Storing and Querying Ontologies in Logic Databases

Timo Weithöner¹, Thorsten Liebig², and Günther Specht¹

¹ Dept. of Databases and Information Systems
University of Ulm
D-89069 Ulm

{timo.weithoener|specht}@informatik.uni-ulm.de

² Dept. of Artificial Intelligence
University of Ulm
D-89069 Ulm

liebig@informatik.uni-ulm.de

Abstract. The intersection of Description Logic inspired ontology languages with Logic Programs has been recently analyzed in [GHVD03]. The resulting language, called Description Logic Programs, covers RDF Schema and a notable portion of OWL Lite. However, the proposed mapping in [GHVD03] from the corresponding OWL fragment into Logic Programs has shown scalability as well as representational deficits within our experiments and analysis. In this paper we propose an alternative mapping resulting in lower computational complexity and more representational flexibility. We also present benchmarking results for both mappings with ontologies of different size and complexity.

1 Introduction

Current research within the Semantic Web aims at combining knowledge representation methods with techniques of the Web. Such a combination would enable meaningful communication between people and heterogenous information processing systems for inter- and intranet applications. Ontologies play a pivotal role within such a framework by providing a shared and common understanding of a domain of interest. Formally, an ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i. e. its ontological commitment to a particular conceptualization of the world [Gua98].

Reasoning about logical theories requires logic-based inference systems which has been well studied within the field of knowledge representation in the AI community over the last decades. Description Logics (DL's) as a decidable fragment of first-order logic turned out to be an adequate formalism for representing and reasoning about expressive ontologies. As a consequence DL's form the formal foundation of W3C's Web Ontology Language (OWL), a proposed standard for a semantic markup language for publishing and sharing ontologies on the World Wide Web.

One of the key design goals for OWL was highest possible expressiveness [Hef03]. However, the more expressive a language is, the more difficult it will be to learn or to use this language. OWL has been criticized because of its high language complexity even by one of its language designers [vH02]. While analyzing online accessible ontologies the same language designer also noticed, that even experienced users exploit a very limited subset of the available language primitives in general. Beyond that, reasoning about ontologies with an expressiveness comparable to that of OWL requires sophisticated logical theorem provers, which are currently only available as research prototypes. Such systems have proven to be fast as well as reliable at least with most of the academic ontologies available so far. However, they are designed to deal with ontologies completely processable within a computers primary memory. In fact, we expect much larger ontologies for real world applications in the near future which very likely will not solely be loadable into (virtual) main memory. More concrete, realistic applications scenarios within the vision of the Semantic Web refer to (currently non-existent) ontologies with a limited set of language primitives but a very large set of individuals (flight schedules, phone books, etc.). Obviously, database technology will be necessary in order to be able to deal with ontologies of this size.

In this sense, as an alternative to tableaux-based DL theorem provers Groszof et. al. [GHVD03] recently suggested a mapping of a DL subset into Logic Programs (LP) suitable for evaluation with Prolog. This intersection of DL with LP called DLP completely covers RDF Schema and a fraction of OWL (notably most of OWL Lite extended with general concept inclusion). This approach is called the “Direct Mapping” approach in the following. Logical database systems seem most suitable to combine LP with efficient and persistent data storage. However, applying the Direct Mapping approach for loading, storing and evaluating ontologies in logical database systems has shown some significant scalability deficits as well as representational drawbacks. Therefore, we developed a new mapping without this limitations which we call the “Meta Mapping” approach below. This approach is meta in the sense that it maps the LP subset of OWL into a higher representational level resulting in lower computational complexity and more representational flexibility. For this reason the Meta Mapping approach is especially suitable for storing and processing ontologies within logical databases. In this paper we present our new Meta Approach together with some benchmarking results for both approaches.

The remainder of this paper is organized as follows. In the next section we will give an overview over logic based ontology languages currently proposed by the W3C and their relationship to Logic Programs as used in logic databases. Readers familiar with OWL and Logic Programs should skip Section 2. Section 3 explains the Direct Mapping approach from the DLP fragment of OWL to LP’s proposed by Groszof and Horrocks et. al. [GHVD03]. In Section 4 we present our Meta Mapping approach while making use of examples showing the conceptual differences between the two approaches. Section 5 contains benchmarking results for both mappings with ontologies of different size and complexity. We will end with a discussion about the pros and cons of the different mappings.

2 Preliminaries

This section will shortly introduce OWL. In particular, we will give syntax examples and their semantics in terms of corresponding First Order Logic (FOL) formulae. We then characterize the intersection of LP's with OWL. Reader familiar with OWL and LP's may skip this section.

2.1 Ontology Languages for the Web

The proposed mechanism for meaningful communication between people and / or machines within the World Wide Web is to add semantic markup to Web resources in order to explicitly describe their content. This semantic markup makes use of terms for which ontologies provide a concrete specification of their meaning.

The significant term structure of ontology languages currently under development for the Web consists of at least two elements (see also [Hef03]): *classes* and *relationships* (called properties) that can exist among classes.

RDF Schema. The two basic structuring elements from above are provided by the Resource Description Framework Schema (RDFS), the lowermost ontology language of the Semantic Web language layer architecture. As with the following ontology languages, RDFS usually is serialized as XML document in order to meet the syntactical requirements of today's Web communication protocols. RDFS can be considered as a very simple ontology language allowing the definition of class hierarchies via `subClassOf` statements. Exemplarily, a dog can be defined as some kind of mammal as follows:

```
<rdfs:Class rdf:ID="Dog"> [Ex. 1]
  <rdfs:subClassOf rdf:resource="#Mammal"/>
</rdfs:Class>
```

Semantically this can be expressed in FOL as an implication between two unary predicates: $\forall x : \text{Dog}(x) \Rightarrow \text{Mammal}(x)$ (DL abstract notation: $\text{Dog} \sqsubseteq \text{Mammal}$).

The possible combinations of classes and properties can be restricted by qualifying the domain and range of properties. An owner relationship for dogs, called `Dog-Owner`, is narrowed in its domain to the class `Human` and in its range to `Dog` in the following:

```
<rdf:Property rdf:ID="Dog-Owner"> [Ex. 2]
  <rdfs:domain rdf:resource="#Human"/>
  <rdfs:range rdf:resource="#Dog"/>
</rdf:Property>
```

The FOL correspondence to properties are binary predicates. According to that, the semantics of the property definition in example 2 is as follows: $\forall x, y : \text{Dog-Owner}(x, y) \Rightarrow \text{Human}(x)$ and $\forall x, y : \text{Dog-Owner}(x, y) \Rightarrow \text{Dog}(y)$ (DL: $\top \sqsubseteq \forall \text{Dog-Owner.Human}$, $\top \sqsubseteq \forall \text{Dog-Owner}^{\neg}.\text{Dog}$).

Property hierarchies can also be defined analogous to class hierarchies using `subPropertyOf` statements.

Web Ontology Language (OWL). OWL is developed as a vocabulary extension of RDF Schema¹ and is derived from the DAML+OIL Web ontology language. This extension covers class language constructs like conjunction, disjunction, negation, existential and universal qualified quantification and cardinality constraints of properties (plus some others). OWL itself provides three increasingly expressive sublanguages. The least expressive sublanguage is OWL Lite. With focus on the intersection of OWL with Logic Programs we will give a more detailed explanation for some of OWL Lite's language constructs here. For syntax and semantics of all constructs see [vHHH⁺03] resp. [PSHH03].

In OWL a class can be defined as conjunction of other classes or class descriptions using the `intersectionOf` statement. For example, it might be rational to define `Puppy` as the conjunction of the classes `Dog` and `Young-Animal`:²

```
<owl:Class rdf:ID="Puppy"> [Ex. 3]
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Dog"/>
        <owl:Class rdf:about="#Young-Animal"/>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

Semantically, this corresponds to logical conjunction in FOL: $\forall x : \text{Puppy}(x) \Rightarrow \text{Dog}(x) \wedge \text{Young-Animal}(x)$ (DL: `Puppy` \sqsubseteq `Dog` \sqcap `Young-Animal`).

So far, all class definitions result in logical implication with respect to their definition. They are therefore called necessary class definitions. In contrast, it is possible to give a necessary as well as sufficient definition for a class. In the following example it is a necessary as well as sufficient condition being a `Dog` and a `Rabbit-Animal` for being a `Rabbit-Dog`:

```
<owl:Class rdf:ID="Rabbit-Dog"> [Ex. 4]
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Dog"/>
    <owl:Class rdf:about="#Rabbit-Animal"/>
  </owl:intersectionOf>
</owl:Class>
```

Logically this corresponds to an equivalence between `Rabbit-Dog` and its defining conjunction: $\forall x : \text{Rabbit-Dog}(x) \Leftrightarrow \text{Dog}(x) \wedge \text{Rabbit-Animal}(x)$ (DL: `Rabbit-Dog` \equiv `Dog` \sqcap `Rabbit-Animal`).

Universal qualified quantification is a language construct for locally restricting the range of a given property within a class definition. E. g., a `Doghouse` is a `house` for which all fillers of the property `Occupants` are of type `Dog`:

¹ However, OWL does not include RDFS's recursive meta model property.

² Since OWL is layered on top of RDFS the examples from above are easily converted into OWL by changing `rdfs:Class` into `owl:Class` and `rdf:Property` into `owl:ObjectProperty`.

```

<owl:Class rdf:ID="Doghouse"> [Ex. 5]
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#House"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#Occupant"/>
          <owl:allValuesFrom rdf:resource="#Dog"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

```

The fragment from above has the following semantics in FOL: $\forall x : \text{Doghouse}(x) \Rightarrow \text{House}(x) \wedge (\forall y : \text{Occupants}(x, y) \Rightarrow \text{Dog}(y))$ (DL: $\text{Doghouse} \sqsubseteq \text{House} \sqcap \forall \text{Occupants.Dog}$).

Additionally, OWL Lite extends RDFS for transitive, inverse, and symmetric properties. A perfect example of a transitive property within our dogs world is the descendants relationship `Dog-Offspring`:

```

<owl:TransitiveProperty rdf:ID="Dog-Offspring"/> [Ex. 6]

```

Semantically, a transitive property enforce that: $\forall x, y, z : (\text{Dog-Offspring}(x, y) \wedge \text{Dog-Offspring}(y, z)) \Rightarrow \text{Dog-Offspring}(x, z)$ (DL: Dog-Offspring^+). An inverse property of a given property can be defined as follows:

```

<owl:ObjectProperty rdf:ID="Is-Dog-Of"> [Ex. 7]
  <owl:inverseOf rdf:resource="#Dog-Owner"/>
</owl:ObjectProperty>

```

In FOL this means: $\forall x, y : \text{Is-Dog-Of}(x, y) \Rightarrow \text{Dog-Owner}(y, x)$ and vice versa (DL: $\text{Is-Dog-Of} \equiv \text{Dog-Owner}^-$). A symmetric property can be introduced as follows:

```

<owl:SymmetricProperty rdf:ID="Friend-Of"/> [Ex. 8]

```

This definition imposes the following FOL semantics: $\forall x, y : \text{Friend-Of}(x, y) \Rightarrow \text{Friend-Of}(y, x)$.

Until now, we have only defined a formal vocabulary. In DL terminology such a vocabulary is called a TBox. In contrast, an ABox specifies concrete individuals with respect to a given TBox vocabulary. For example, `Fluffy` as a concrete dog can be defined by instantiating the class `Dog`:

```

<Dog rdf:ID="Fluffy"/> [Ex. 9]

```

Such a class instantiation corresponds to an unary predicate instantiation in FOL (and abstract DL): $\text{Dog}(\text{Fluffy})$. Let us assume we also want to assert a particular doghouse (with name `Fidos-Kennel`) occupying `Fido` an animal with rabies. FOL equivalent: $\text{Doghouse}(\text{Fidos-Kennel}), \text{Occupant}(\text{Fidos-Kennel}, \text{Fido})$ and $\text{Rabit-Animal}(\text{Fido})$:

```

<Doghouse rdf:ID="Fidos-Kennel">                                     [Ex. 10]
  <Occupant>
    <Rabbit-Animal rdf:ID="Fido"/>
  </Occupant>
</Doghouse>

```

The underlying semantics of our definitions allows to infer logically entailed knowledge implicitly encoded in our dogs world ontology. Usually, logical reasoning systems are used for making such knowledge explicitly available for users.

Concerning our example a simple inference chain could be the following. Since all occupants of a doghouse necessarily have to be dogs (due to the definition of `Doghouse` in Example 5) it follows, that `Fido` has to an instance of class `Dog`. As a consequence `Fido` is also classified as a `Rabbit-Dog` because being a dog and a rabid animal is sufficient for being a `Rabbit-Dog`.

2.2 Intersecting OWL with Logic Databases

Based on the analysis of the intersecting language of DL with LP in [GHVD03] we will shortly characterize the resulting language of the intersection of DL inspired OWL with LP in the following.

Logic Programs (LP) consist of a set of rules having the form:

$$A \leftarrow B_1 \wedge \dots \wedge B_n \quad \text{with } n \geq 0$$

where A, B_i are atomic formula of predicates of arbitrary arity. A is called the head of the rule and the conjunction of B_i 's is called the body. Atomic formula are allowed to be "negated". Note that this negation has to be interpreted as negation-as-failure. Negation-as-failure as well as other LP features (like procedural attachments) are not expressable in FOL and therefore also not expressable in logic based ontology languages. On the other hand predicates in LP are not restricted in their arity in contrast to DL. In addition DL restricts the usage of (implicit) free variables in quantifying expressions with guarding property predicates. As a result the intersection of DL with LP can be characterized as a definite (without negation-as-failure) equality-free Horn fragment of FOL, called Description Logic Programs (DLP) in [GHVD03].

Since OWL is a DL inspired language it covers DLP completely. More precise, DLP covers most of OWL Lite (the least expressive sublanguage of the OWL family) plus a portion of OWL DL, namely general concept inclusions (GCIs) with disjunction and qualified existential qualification on the l.h.s. However, GCIs are not very common in ontologies so far, we will focus on the intersection of OWL Lite with LP, which we will call OWLP Lite for the rest of the paper.

3 The Direct Mapping Approach

In the following we will provide two approaches of converting ontologies into logic programs. Starting with the previously published proposal in [GHVD03], the

direct mapping approach, we will suggest an alternative meta mapping approach in Section 4. After that we will provide an evaluation and comparison of both approaches, which makes it necessary to have a look at different aspects like the number of facts and rules contained in the resulting logic programs.

A straight forward approach to convert ontologies into a logic program is described in [GHVD03]. This approach maps every class or property definition contained in the ontology into a rule and every class-instance or instance-property-instance relationship into a fact. In the following we summarize this method of mapping into a description logic program.

3.1 The Mapping

Every concept instantiation is mapped to a unary relation with the concept name becoming the name of the relation and the individual name becoming the only argument. For example the statement that `Fluffy` is an instance of concept `Dog` (see Example 9) is mapped into the fact:

```
Dog("Fluffy"). [Ex. 11]
```

Every instance-property-instance relationship is mapped into a binary relation with the property's name becoming the name of the relation. The first argument is the name of the individual, the second argument is the property's value. Given a property `Occupant`, let us assume `Fidos-Kennel` being occupied by `Fido` (as defined in Example 10):

```
Occupant("Fidos-Kennel", "Fido"). [Ex. 12]
```

In addition, concept as well as property constructor statements are converted into a set of rules. In this step it gets obvious, why we called this approach the Direct Mapping approach. Every subclass relationship stated in the ontology is directly mapped into a corresponding logic program rule. As a consequence the OWL fragment defined in Example 1 is converted into:

```
Mammal(X) :- Dog(X). [Ex. 13]
```

For a mapping of the rest of the OWLP Lite language constructs see Table 1. Note, that the `EquivalentClasses` relationship is just a mutual `SubClassOf` relationship. An `EquivalentClasses` relationship can be simulated by two `SubClassOf` statements using both definition directions.

3.2 Size of the Resulting Program

Let us now have a look at the resulting logic program of a given ontology with a set \mathcal{C} of classes and a set \mathcal{P} of different properties. Let $\mathcal{R}_{\mathcal{C}}$ with $c \in \mathcal{C}$ be the set of rules required to express c . With \mathcal{I} being the set of class instantiations (see Example 11) and \mathcal{V} the set of property instantiations (see Example 12) defined within the ontology we get an total of

Table 1. Shows OWL statements and there representation in a logic program as suggested [GHVD03]. See Section 3.2 for definition of \mathcal{R}_c .

OWL Abstract Syntax [PSHH03] Definition of class c	DLP Statements Rule set \mathcal{R}_c	$ \mathcal{R}_c $
SubClassOf(c b)	$b(X) :- c(X).$	1
SubClassOf(unionOf($b_1 \dots b_n$) c)	$c(X) :- b_1(X).$...	n
SubClassOf(c intersectionOf($b_1 \dots b_n$))	$c(X) :- b_n(X).$	1
SubClassOf(intersectionOf($b_1 \dots b_n$) c)	$c(X) :- b_1(X), \dots, b_n(X).$ $b_1(X) :- c(X).$...	n
SubClassOf(c restriction(p allValuesFrom(b)))	$b_n(X) :- c(X).$ $c(X) :- p(X, b), \text{anonID}(X).$	1
SubClassOf(restriction(p someValuesFrom(b) c)	$\text{anonID}(X) :- p(X, b), c(X).$	1

$$\sum_{c \in \mathcal{C}} |\mathcal{R}_c| \quad (1)$$

rules and

$$S = |\mathcal{I}| + |\mathcal{V}| \quad (2)$$

facts, while the number of different predicates is

$$K = |\mathcal{C}| + |\mathcal{P}| \quad (3)$$

To understand of the above we will establish two criteria to compare different ontologies: Size and complexity of an ontology. Size means the number of concept and property instantiations included (see S in Equation 2) while complexity means the number of different concepts (see K in Equation 3). For example you would expect that an ontology made from data contained within a phonebook, would be of very limited complexity (containing only a very limited number of concepts like name, address and phone number) but of huge size (listing all inhabitants and thus having lots of individuals). Nevertheless the number of rules in the resulting logic program is growing linear with the number of concept and property definitions in the ontology.

3.3 A first assessment of the approach

Limitations. Looking at the logic program fragments as discussed above we soon realize that this approach has some significant weaknesses:

- First, the concept names cannot be accessed from within the logic program. For that reason it is virtually impossible to get an answer to the question “give me all classes the individual I is instance of”.

Reconsider Example 10 out of our dogs ontology. We are aware of the fact that `Fido` is a `Rabbit-Animal`. However we are not able to retrieve all classes `Fido` is an instance of unless we *manually* walk through every possible class and ask whether `Fido` is instance of this class. Which is not very efficient and would require to know all class names of the ontology.

- The transformation creates a limited number of facts. One for every class instantiation and one for every property instantiation. The number of different relations depends on the number of classes and properties defined in the ontology. One can expect to get a very limited number of facts per relation. While the number of facts remains manageable the number of different rules grows linear with the complexity of the ontology (the knowledge included within). See Table 1 for the number of rules needed to express a given OWL statement.
- The names of the relations used and the structure of the rules involved vary from ontology to ontology. Consequently precompilation or query optimization become an real issue in this approach.

Possible Improvements. Two things have to be done to overcome the limitations mentioned above.

1. The rules and facts have to be pushed to a meta level, where names of concepts and properties become arguments of “meta predicates”. As a result concept and property names can be reached from within the logic program easily.
2. We should try to get a constant set of rules valid for all ontologies accompanied by a set of fact predicates with constant names. Like this queries would look the same for every single ontology with the only difference in the amount of facts that have to be processed to get an answer.

The following section will suggest an approach which will produce a logic program following the above considerations.

4 The Meta Mapping Approach

This section will describe our approach of mapping an ontology into a logic program suitable for a deductive database. We will show that even if an ontology grows in complexity the resulting logic program will have a constant number of rules and predicates. And we will show how ABox as well as TBox reasoning become possible without the limitations from above.

4.1 The Basic Idea

Basically we convert the OWL statements contained in an ontology into a set of facts reflecting the content of the ontology. Coming from the Direct Mapping approach we basically push all facts defined there to a meta level comparable (at

least in parts) to the HiLog [CKW93] approach which has a higher-order syntax and allows terms to appear in places where predicates and atomic formulas occur in FOL. The meta mapping is realized by two new relations, which collect all facts defined in the various relations of the Direct Mapping approach.

Class Instantiation. Asserting instance *i* to be of class *C* results in instantiating the binary relation named `type` in the following way:

```
type("i", "C").
```

Property Instantiation. Likewise property instantiations are mapped into a relation named `propInst` with three arguments and constant name `propInst`. Arguments are the property name *P*, instance name *i* and property value *v*:

```
propInst("P", "i", "v").
```

Compared to the “Direct Mapping” Approach we avoid having one additional relation per property definition, by pushing the relation names into predefined “meta relations”. As a consequence concept and property names are now easily accessible from within the logic program. Table 2 compares the resulting logic program fragments from Examples 9 and 10 for both approaches.

Table 2. Individual and property definition in the different approaches

	Direct Approach suggested in [GHVD03]	Our Meta Approach
Fluffy is a Dog	Dog("Fluffy").	type("Fluffy", "Dog").
Fido is the Occupant of Fidos-Kennel	Occupant("Fidos-Kennel", "Fido")	propInst("Occupant", "Fidos-Kennel", "Fido").

Handling of Class Constructors. Let us once again have a look at the very basic OWL fragment stating that concept `Dog` is a subclass of concept `Mammal` as defined in Example 1. The Direct Mapping approach would create a rule which would say that every individual of type `Dog` is also an individual of type `Mammal` (See Example 13). Please note that the explicit knowledge of the subclass relationship gets lost. All we still know is that every `Dog` is also a `Mammal`. In the Meta Mapping approach subclass relationships or any other kind of constructors are not converted into a rule covering the meaning of the statement but into a fact which states, that the ontology defines such a relationship. Consequently the number of rules is not increased if a new class is constructed and no new relations are needed, as there is a fixed number of predefined relations, reflecting the vocabulary of OWLP Lite. The above example would consequently look like this in the Meta Mapping approach:

`isSub("Dog", "Mammal").` [Ex. 14]

In order to reflect the underlying semantic of the introduced meta relations, we will have to add some rules, which work on the given facts. The following rule defines that if an individual *I* is instance of concept *Y* and *Y* is subclass of concept *X*, *I* is also an individual of class *X*:

`type(I, X) :- isSub(Y, X), type(I, Y).`

As the rule can be used with any combination of bound and free variables, every kind of class-instance query (ABox reasoning) is possible (in contrast to the Direct Mapping approach). Additionally the transitivity of the subclass relationship is covered by the following rule:

`isSub(X, Y) :- isSub(X, Z), isSub(Z, Y).`

As you can see the above rules are completely independent of any entities defined in the ontology and can thus be used for every ontology. With the combination of the ontology specific facts and the general rule we can now perform all A- and TBox queries.

Table 3. Shows how A- and TBox reasoning is performed in the different approaches

Query	Meta Approach	Direct Approach
Is given individual <i>i</i> instance of given class <i>C</i> ?	<code>?type("i", "C").</code>	<code>?C("i").</code>
List all instances of given class <i>C</i> .	<code>?type(I, "C").</code>	<code>?C(I).</code>
List all classes given individual <i>i</i> is instance of.	<code>?type("i", C).</code>	Manually go through every known class <i>C</i> and check for <code>?C("i").</code>
Check if given class <i>C</i> is subclass of given class <i>D</i> .	<code>?isSub("C", "D").</code>	Create new instance <i>i_C</i> of class <i>C</i> and check whether <i>i_C</i> is also instance of class <i>D</i> .
List subclasses of given class <i>C</i> .	<code>?isSub(X, "C").</code>	Manually go through every known class <i>D</i> create new instance <i>i_D</i> from this class. Check whether <i>i_D</i> is also instance of <i>C</i> .

While the queries in our Meta Mapping approach only require basic knowledge of the entities defined in the ontology, Direct Mapping approach requires complete knowledge of all class names defined to be able to perform any class hierarchy query. This constitutes a big disadvantage for the user, as he either has to keep track of all class names and the results of his queries are always questionable or he has to provide and use additional predicates providing class (and property) names.

4.2 The Rule Set

In the above section we mentioned two rules which provide the logic behind the meta level relations we defined to store the ontologies knowledge. Depending on the number of different constructors used in the ontology this set of rules will vary in size. But the size of this rule set is not depending on the size or complexity of the ontology. E.g. if an ontology uses the `intersectionOf` statement the corresponding rules have to be added to the logic program. Any further use of the `intersectionOf` statement will not increase the rule set. Nevertheless for the following considerations we assume the rule set to be constant. This is achieved by working with the complete rule set, containing rules for every OWLP Lite statement no matter if they are used in the ontology or not. Table 4 shows some of the required facts for comparison with the Direct approach (see Table 1). Table 5 gives a nearly complete summary of the required rules and facts of our Meta Mapping approach.

Table 4. Shows OWL statements and there representation in a logic program

OWL Abstract Syntax Definition of class c	DLP Statements Fact set \mathcal{F}_c	$ \mathcal{F}_c $
<code>SubClassOf(c b)</code>	<code>isSub(c, b).</code>	1
<code>SubClassOf(c unionOf(b₁ ... b_n))</code>	<code>rhsUnionOf(c, {b₁, ..., b_n}).</code>	1
<code>SubClassOf(c intersectionOf(b₁ ... b_n))</code>	<code>rhsIntersectionOf(c, {b₁, ..., b_n}).</code>	1
<code>SubClassOf(intersectionOf(b₁ ... b_n) c)</code>	<code>lhsIntersectionOf(c, {b₁, ..., b_n}).</code>	1
<code>SubClassOf(c restriction(p allValuesFrom(b)))</code>	<code>rhsAllValuesFrom(c, p, b).</code>	1
<code>SubClassOf(restriction(p someValuesFrom(b)) c)</code>	<code>lhsSomeValuesFrom(c, p, b).</code>	1

4.3 Influence of the Ontology's Size

Again we have to consider the size of the resulting logic program in dependence of the size and complexity of the given ontology. With \mathcal{I} being the set of individuals, \mathcal{P} the set of properties defined in the ontology, \mathcal{V} the property values defined, \mathcal{C} being set of concepts defined within the ontology and finally \mathcal{F}_c with $c \in \mathcal{C}$ being the set of facts required to express concept c we get an total of

$$|\mathcal{I}| + |\mathcal{V}| + \sum_{c \in \mathcal{C}} |\mathcal{F}_c| \quad (4)$$

facts while the number of different predicates as well as the number of different rules remains constant independent of the ontology. Depending on the implementation – and the optimizations within – these numbers may vary but

Table 5. Shows a selection of the rules required in the Meta Mapping approach.

DL syntax	“Meta Mapping” representation
$i : C$	<pre>type("i", "C"). type(I, C) :- type(I, Z), isSub(Z, C).</pre> <p><i>If individual I is instance of Z and Z is subclass of class C, I is also instance of class C.</i></p>
$(i, v) : P$	<pre>propInst("P", "i", "v"). propInst(P, I, V) :- propInst(Q, I, V), subPropertyOf(Q, P).</pre> <p><i>If an instantiation I,V holds for property Q, it also holds for any property P that Q is subproperty of.</i></p>
$C \sqsubseteq D$	<pre>isSub("c", "d"). isSub(C, D) :- isSub(C, Z), isSub(Z, D).</pre> <p><i>The subclass relationship is transitive.</i></p>
$P \sqsubseteq Q$	<pre>subPropertyOf("P", "Q"). subPropertyOf(P, Q) :- subPropertyOf(P, Z), subPropertyOf(Z, Q).</pre> <p><i>The subproperty relationship is transitive.</i></p>
$\top \sqsubseteq \forall P.C$	<pre>domain("P", "C"). type(I, C) :- propInst(P, I, V), domain(P, C).</pre> <p><i>If instance I is in the domain of a relation P with a domain restriction on class C, then I is an instance of C.</i></p>
$\top \sqsubseteq \forall P^-.C$	<pre>range("P", "C"). type(V, C) :- propInst(P, I, V), range(P, C).</pre> <p><i>If instance V is in the range of a relation P with a range restriction on class C, then V is an instance of C.</i></p>
$C \sqsubseteq M_1 \sqcap \dots \sqcap M_n$	<pre>rhsIntersectionOf("C", {"M1", ..., "Mn"}). isSub(C, M) :- rhsIntersection(C, S), member(S, M).</pre> <p><i>Class C is subclass of any of the members of the intersection.</i></p>
$M_1 \sqcap \dots \sqcap M_n \sqsubseteq C$	<pre>lhsIntersectionOf("C", {"M1", ..., "Mn"}). oneOfOtherType(I,S) :- member(S,M), not type(I,M). type(I, C) :- lhsIntersectionOf(C, S), not oneOfOtherType(I, S).</pre> <p><i>An Individual I is an instance of C if there is no member of the intersection of which I is not an instance of.</i></p>
$C \sqsubseteq \forall P.D$	<pre>rhsAllValuesFrom("C", "P", "D"). type(I, D) :- rhsAllValuesFrom(C, P, D), type(X, C), propInst(P, X, I).</pre> <p><i>If instance X of class C is related to an individual I via a property P and the range of P is locally restricted to D in C, then I must be of type D.</i></p>
$P^+ \sqsubseteq P$	<pre>transitiveProp("P"). propInst(P, I, V) :- transitiveProp(P), propInst(P, I, Z), propInst(P, Z, V).</pre> <p><i>If P is a transitive property then for all property instantiations I,Z and Z,V also the instantiation I,V holds.</i></p>

either way, we are talking about a rule set, which should not exceed 20-40 rules and about the same quantity of different predicates.

More precisely linear increase of ontology size and/or complexity leads to a linear growth of facts but a constant set of relations and rules. In contrast applying the Direct Mapping approach would result in linear growth of relations, rules and facts. The following section provides a detailed performance analysis with respect to the different approaches in a logical database.

5 Evaluation

In the preceding sections of this paper we discussed two different approaches for mapping ontologies into description logic programs. While introducing the approaches, we came across a number of conceptual issues, which in some cases resulted in a loss of functionality. In other cases we presumed influence on the performance of the resulting logic program. Especially as ontologies are expected to rise in size (and complexity), we will have to think about reasoning systems, which are not entirely depend on main memory (like today's tableau reasoners or prolog) but are able to work on data in secondary storage structures, which immediately leads to database technology.

5.1 The Selected Database System

Deductive Database Systems. Due to the representation of the ontologies as logic programs it is obvious to choose a logic database or – to be more specific – a deductive database to store and query the knowledge contained in the ontologies. In the years 1988-1992 a number of deductive database systems like LDL, NAIL, LOLA and CORAL have been developed [Spe91]. Novel features of these systems have been:

1. Logic programs similar to Prolog replaced SQL as database definition and query language.
2. Arbitrary recursion (left/right hand side, quadratic) were introduced.
3. Deductive Databases abolished NF1 and allowed arbitrary (even recursive) term structures.

Basics of logic databases are a set at-a-time (not tuple at-a-time) bottom-up (not top-down) evaluation. To do so logic programs are converted into a relational algebra program internally, where each predicate symbol is represented by a relation, rules correspond to views and each fact can be understood as an entry in a base relation. Additionally in the rule body joins are created from conjunctions over predicates with equal variable names and instantiated terms are converted into selections. Negations become antisemijoins (a special kind of set difference) and recursion is evaluated using delta iteration or seminaive iteration respectively. Finally the transition from rule body to head becomes a projection.

Some of these features, particularly recursion and abolishment of NF1, have been re-adopted by the relational database technology. Consequently you will find some of the above in SQL:99, but there are still limitations like quadratic recursion (which is not used in either the Meta-³ nor the Direct Mapping approach).

Even though in the future a further mapping of the logic programs into SQL:2003 might be desirable, we base the following considerations upon a deductive database and thus stay with the logic programs as described in Sections 3 and 4. As both – relational and deductive – systems work with relational algebra programs internally, we assume only minor influence on the following discussion.

The Deductive Database System Coral. For the implementation of our test cases we have chosen the deductive database system CORAL [RSSS94], as it is a stable, well tested and easily available system. Special technical features of other deductive databases that would qualitatively lead to different results will also be addressed in the following.

5.2 Measurements

When measuring the performance of a deductive database processing a logic program two measures have to be considered. First we need to have a look at the time needed to read the logic program into the database and second we have to measure the time it takes to run the logic program (perform a given query). There are certain optimization steps like magic set transformation, which are performed at different times depending on the database system used. E.g. the deductive database system LOLA [FSS91] is performing this optimization task after the query is known, only considering those rules required for the specific query. In contrast the CORAL system is performing the magic set transformation (and other optimization like supplementary magic) when the logic program is initially loaded into the database. Thus *all* binding combinations⁴ are precompiled in CORAL. This gives us the opportunity to measure the time needed to perform this optimization tasks independently from the time needed to perform certain queries.

5.3 Testcase

We used a set of different ontologies for our performance tests. Even though these ontologies vary in size and complexity our main focus was the influence of the ontologies size.

³ In fact you will find an example of a quadratic recursion in Section 4.1. This recursion can easily be converted into a left/right hand side recursion in an optimized implementation.

⁴ arguments in the query may be bound (without variables) or free (variables or terms containing variables).

Testcase 1. The first set of ontologies, consisted of the a class hierarchy of 20 classes with a varying number of individuals. We used this test case to show the influence of the ontology’s size. We measured the time needed for preprocessing and time needed to process a class instance query.

Testcase 2. This test case consisted of a set of ontologies with a growing number of classes and a constant number of one instance per class. It was used to show the influence of the complexity of an ontology. We measured the time needed for preprocessing and one class subsumption query.

5.4 Test Environment

All tests were performed on a Pentium II (350 MHz) Linux system (kernel 2.4.19) with 250MB main memory running CORAL Version 1.5.2. Time Measurement was performed by the CORAL builtin commands `reset_timer()` and `display_timer()`, which were included in the logic programs. We present “CPU Time” measurements in this paper.

5.5 Results

Loading and Preprocessing the Logic Program. Figures 2 and 1 show that, our Meta Mapping approach outperformed the Direct Mapping approach when preprocessing the logic program. In fact preprocessing time grows linear with the number of instances and classes defined in the ontology. In contrast to the Direct Mapping approach, showing exponential rise (see Figure 2) of preprocessing time with a growing number of classes. In our test environment we very easily reached a point where CORAL was not able to load such a logic program within reasonable time⁵ while loading the same ontology converted into a Meta Mapping approach logic program took only seconds. This also meant that we were unable to perform a comparison of query time for really huge ontologies. Only the Meta Mapping approach is able to handle this case.

Querying the Logic Program. When comparing query times you have to keep in mind a number of issues which we will highlight in the following paragraphs.

Some queries can’t be performed in the Direct Mapping approach. For example you can’t get a list of all concepts defined in an ontology. Thus there is nothing to compare with the Meta Mapping approach.

Additionally there are a number of queries which cannot be performed in the logic program itself when using the Direct Mapping approach. Some queries would require some kind of manual interaction or scripting which – again – makes it useless to compare the runtime of queries.

CORAL as well as other deductive database systems work as described in Section 5.1 but in general do not use secondary storage structures, by now. It is

⁵ We aborted all tests after 30 minutes.

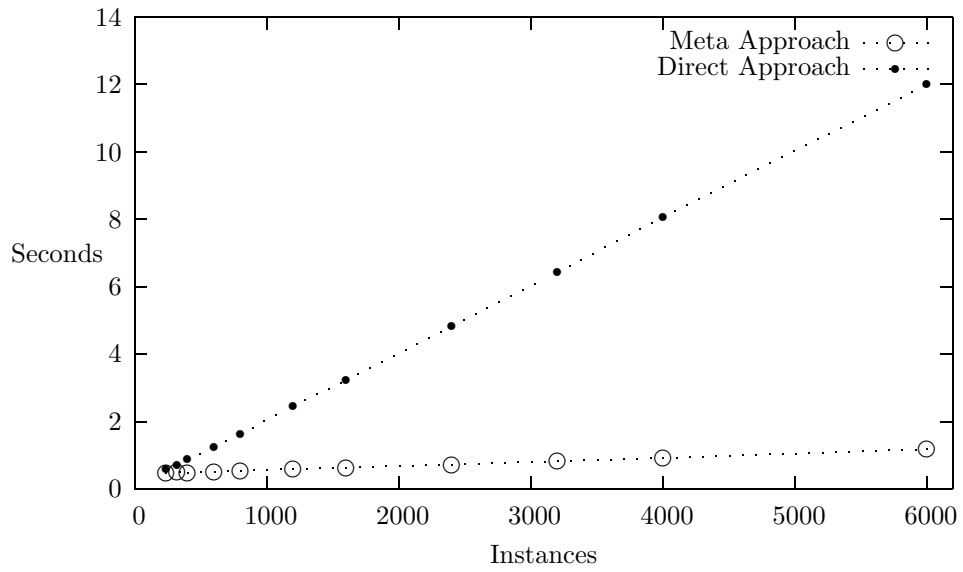


Fig. 1. Shows the time needed to preprocess the ontologies of Testcase 1 (see Section 5.3).

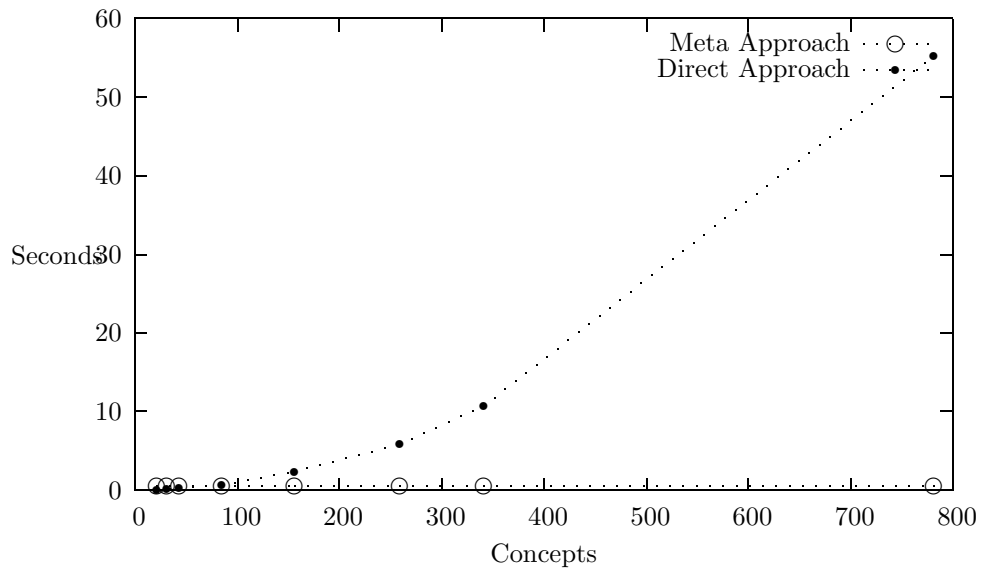


Fig. 2. Shows the time needed to preprocess the ontologies of Testcase 2 (see Section 5.3).

thus questionable whether efficient indexing mechanisms have been implemented. With a rising number of facts (tuples) such indexing mechanisms would be highly desirable. Indeed first tests indicate that CORAL lacks such mechanisms, which especially slows down the Meta Mapping approach as it has to perform a larger number of joins (because the average number of conjunctions per rule is higher) in comparison to the Direct Mapping approach. The Meta Mapping approach creates only few different relations with a comparatively large number of tuples. This is a procedure as meant for classical relational systems, which use indexing mechanisms like B-trees. In such systems we would thus expect only logarithmic increase in response time for the Meta Mapping approach. In fact we currently measure linear behavior for both approaches (as can be seen in Figure 3).

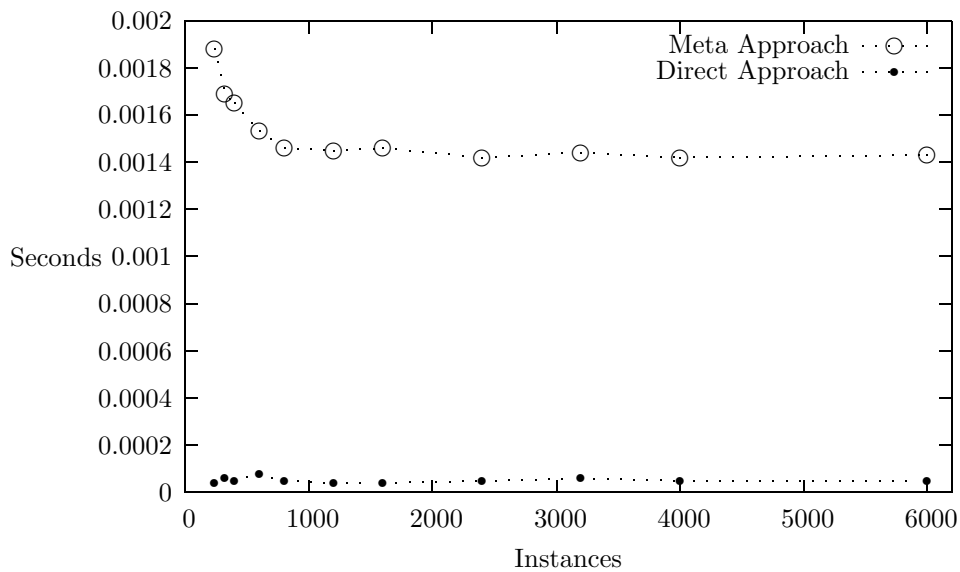


Fig. 3. Shows the linearity of the query time in Testcase 1 (See Section 5.3). The quotient of the number of instances and the query processing time is constant.

When having a look at current query time measurements we realize that the Direct Mapping approach answers faster (wherever it is able to deliver an answer at all) than the Meta Mapping approach. However we expect contrary results with a growing number of instances (which we cannot load into the database system using the Direct Mapping approach) and indexing performed on the relations.

In addition, CORAL performs the magic set transformation and other optimizations during the initial loading of the logic program. Other systems like the deductive database system LOLA perform these optimizations during query

processing. As you can see from the measurements performed this time is significantly higher (Figures 1 and 2) and growing exponentially (Figures 2) with the Direct Mapping approach. We can thus assume, that using the deductive database system LOLA (or any system working the same way) the query time would grow exponentially using the Direct Mapping approach but still remain linear using the Meta Mapping approach.

6 Summary

The Semantic Web aims at extending the current Web in a way such that content of Web pages will not exclusively be meaningful for humans. Here, ontologies play a key role by providing machine processable semantics. A recent W3C working draft proposes OWL as ontology language for the Web. However, modeling in OWL requires quite some formal logical skills as well as elaborated reasoners which are not available of the shelf so far. In addition, reasoning systems very likely have to cope with much larger ontologies consisting of a huge number of individuals in the near future. However, most of today's knowledge processing systems are not designed to use secondary storage mechanisms and will therefore not meet upcoming scalability requirements. In a first step, it therefore seems promising to focus on the intersection of OWL with Logic Programs suitable to adopt logical database technology. Logical databases provide a declarative representation and querying language as well as efficient and persistent data storage. A mapping of OWL into Logic Programs has recently been proposed by [GHVD03]. In this Direct Mapping every class or property definition maps into a rule and every class or property instantiation into a fact of the resulting logic program. However, this approach has some representational as well as practical drawbacks. A conceptual disadvantage with far reaching consequences is the fact, that classes as well as properties are not accessible unless their names are explicitly known to the user

We therefore proposed a new approach using a mapping into a logical representation on an appropriate abstract meta level similar to a subset of the HiLog [CKW93] higher-order syntax. We showed that this Meta Mapping overcomes this limitations. Classes as well as properties are accessible as first class entities within this mapping allowing comfortable query formulation.

In our approach an ontology will be mapped into a logic program consisting of a constant number of rules with a linear growing number of facts proportional to the number of classes and properties. In the Direct Mapping the number of relations in the resulting logic program grows linear with the number of class and property definitions of the original ontology. When benchmarking both mappings it turned out that even a linear growth in relations of the Direct Mapping results in fatal performance during preprocessing while loading the program into the CORAL deductive database. Our experiments also observed a linear growth of time for class instance and subclass querying with an increasing number of individuals for both approaches. However, in case of the Meta Mapping approach

better results are expected for systems with secondary storage indexing mechanisms used in commercial systems today.

In consideration of the above we favor the Meta Mapping approach because of its significant conceptual advantages, higher expressivity and better performance for storing and evaluation of large scale real world ontologies in logical databases.

References

- [CKW93] Weidong Chen, Michael Kifer, and David Scott Warren. HILOG: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, 1993.
- [FSS91] Burkhard Freitag, Heribert Schütz, and Günther Specht. LOLA - A Logic Language for Deductive Databases and its Implementation. In *Proc. 2nd International Symposium on Database Systems for Advanced Applications (DASFAA '91)*, pages 216 – 225, Tokyo, Japan, April 1991.
- [GHVD03] Benjamin N. Grosz, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programms: Combining Logic Programms with Description Logic. In *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary, May 2003.
- [Gua98] Nicola Guarino. Formal Ontology and Information Systems. In *Proceedings of FOIS'98*, pages 3 – 15, Trento, Italy, June 1998.
- [Hef03] Jeff Heflin. Web Ontology Language (OWL) Use Cases and Requirements. W3C Working Draft, March 2003.
- [PSHH03] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. W3C Working Draft, March 2003.
- [RSSS94] Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, and Praveen Sehadri. The CORAL Deductive System. *VLDB Journal: Very Large Data Bases*, 3(2):161 – 210, 1994.
- [Spe91] Günther Specht. LOLA, LDL, NAIL!, RDL, ADITI and STARDUST: A Comparison of Deductive Database Systems. Technical report, Institut für Informatik, TU München, 1991.
- [vH02] Frank van Harmelen. The Complexity of the Web Ontology Language. *IEEE Intelligent Systems*, 17(2):71 – 72, March/April 2002.
- [vHHH⁺03] Frank von Harmelen, Jim Hender, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn A. Stein. OWL Web Ontology Language Reference. W3C Working Draft, March 2003.

DESIGN REPOSITORIES ON THE SEMANTIC WEB WITH DESCRIPTION-LOGIC ENABLED SERVICES

Joseph B. Kopena* and William C. Regli**

Geometric and Intelligent Computing Laboratory
Department of Computer Science, College of Engineering
Drexel University
<http://gicl.cs.drexel.edu/>

Abstract. All engineering firms maintain archives of previously designed artifacts, often in the form of databases of computer aided design (CAD) data. Design repositories are an evolution of such databases to include more heterogenous information and to provide enhanced capabilities through the application of knowledge representation techniques. This paper introduces on-going work on applying description logic and the Semantic Web to constructing such design repositories.

1 Introduction

Engineers spend large portions of their time searching through vast amounts of corporate legacy data and catalogs searching for existing solutions which can be modified to solve new problems or to be assembled into a new device [1]. This requires utilizing databases or online listings of text, images, and computer aided design (CAD) data. Browsing and navigating such collections are based on manually-constructed categorizations which are error prone, difficult to maintain, and often based on an insufficiently dense hierarchy. Search functionality is limited to inadequate keyword scanning or matching on overly simplistic attributes.

Design repositories [2, 3] are an evolution of traditional design databases. They aim to overcome existing limitations by applying knowledge representation techniques to storing and working with artifacts in the collection. Function, behavior, rationale, and other aspects of the designs are captured and reasoned on to enable search, categorization, and other tasks in support the engineer, similar to case-based reasoning [4]. Figure 1 depicts this process. Design repositories typically also extend design databases by including more heterogenous information, incorporating such items as CAD data, documentation, simulations, animations, and analyses. Many also have the explicit goal of providing support throughout the lifecycle of a design, as opposed to focusing solely on detailed design geometry suitable for manufacturing.

This paper introduces and briefly overviews work on applying description logic and the Semantic Web to constructing design repositories. An ontology of electromechanical devices based on function and flow is used in representing, classifying, and comparing such devices. In particular, the application of least-common-subsumer and most-

* At the National Institute of Standards and Technology during preparation of this document.

** Also of the Department of Mechanical Engineering; Email: regli@drexel.edu

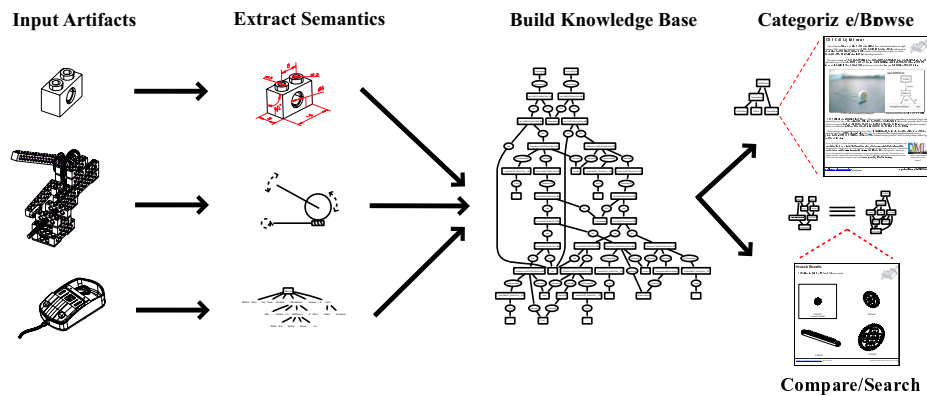


Fig. 1. The design repository process.

specific-class induction to providing design repository inferences not possible with traditional database implementations is introduced. The role of the Semantic Web in creating a new kind of public, distributed, multi-source design repository is also discussed.

The organization of this paper is as follows: Section 2 compares this effort to related work in assembly representation and design repositories. Description logic-based knowledge representation and reasoning for design repositories is described in Section 3. Section 4 discusses design repositories on the Semantic Web. Finally, Section 5 presents some closing conclusions.

2 Related Work

This section analyses work on assembly representation and repository reasoning.

Most familiar to engineers are the representations of devices used in Computer Aided Design (CAD) packages. At the core of these are the solid models of the device's components. Upon these are placed constraints in the form of analytic geometry equations describing the motions present in the mechanism [5–7]. However, such equations provide little basis for reasoning beyond simulation through constraint solving. In addition, current CAD does not typically capture abstract information such as function in any form suitable for automated reasoning or even efficient human use, nor information across multiple domains (e.g. electrical and mechanical).

Also familiar to engineering design students are notations of function as in [8]. Many representations explicitly capture the functions present in an assembly [9–11]. These typically capture function information at variable levels of abstraction and across multiple domains but lack the formal framework to support automated reasoning. Repositories using such representations often rely on simple keyword and attribute matching.

Qualitative physics and logic-based representations [12–16] attempt to define the semantics of assemblies in more abstract manners than the geometric equations employed in CAD in order to provide for richer inferences. However, these systems typically do not address many types of inference associated with design repositories, such as deter-

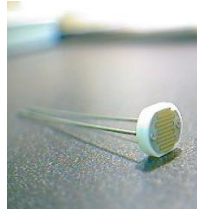


Fig. 2. Typical CDS Cell.

mining similarities between devices. In addition, the expressiveness of the languages used often incurs significant cost in terms of computability and tractability.

3 Representation and Reasoning

Description logic has been chosen as the formal framework for this work because of its favorable complexity results and inference capabilities. In order to promote efficient reasoning, careful attention has been paid to the class expression constructs used. The central ontology is simple enough to be expressed in $\mathcal{AL}\mathcal{EN}^1$. For simplicity, in this paper \mathcal{EL} is used in the examples while in practice $\mathcal{AL}\mathcal{EN}$ is also used for these.

For reasons of space², the ontology defining the representation is not given in this paper but is instead shown through examples. The core of the ontology defines a simple representation of artifacts based on function and flow [9–11]. Extensions to the core ontology define extensive taxonomies of functions and flows derived from those presented in [9] and [10], which were developed from large surveys of engineers.

Figure 2 depicts a typical artifact, a cadmium sulfide cell. These cells are used as light sensors. They are photoresistors; the presence of light decreases the resistance encountered by electricity flowing across the cell. This information can be represented as in Figure 3(a) by a *function and flow diagram* similar to those presented in [9] but not identical. By interpreting the diagrams as a set of objects and relations, they can be represented in a description logic model as in Figure 3(b).

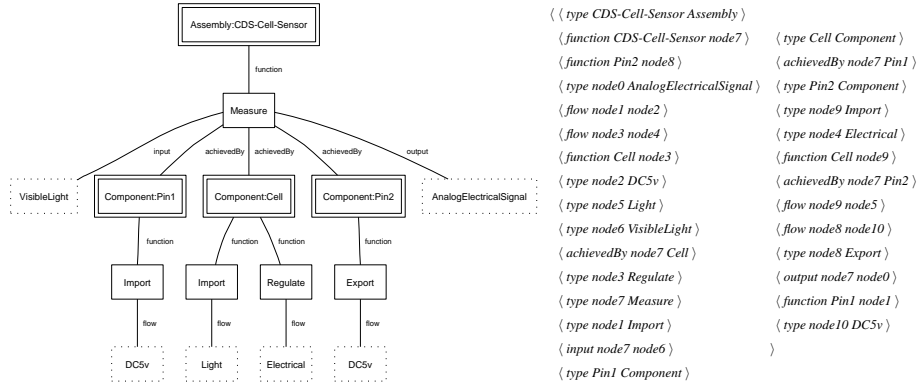
This representation does not rigorously and unambiguously capture the semantics of mechanisms. Instead, it provides a language expressive enough to describe and distinguish devices while maintaining efficiency and computability. It is neither so formal as to prevent practical computing, nor so informal as to prohibit automated reasoning.

Classification of objects is a standard description logic inference. It can be used to search for devices meeting specific criterion as well as to apply a manually developed categorization scheme. For example, the class of electronic sensors could be defined as those objects which measure some input and return an electrical signal:

$$\text{Electronic-Sensor} \equiv \text{Artifact} \sqcap \exists \text{function}. [\text{Measure} \sqcap \exists \text{input}. \text{Flow} \sqcap \exists \text{output}. [\text{Electrical} \sqcap \text{Signal}]].$$

¹ A good reference for description logic expressivity notations is [17].

² (and the lead author breaking an arm in five places shortly before the submission deadline...)



(a) Simplified function and flow diagram.

(b) Figure 3(a) as model.

(c) Function and flow diagram as class description.

Fig. 3. Function and flow modeling of a Cadmium Sulfide (CDS) Cell, a common photoresistor.

The representation in Figure 3 matches this definition, provided that $Assembly \subseteq Artifact$, $VisibleLight \subseteq Flow$, and $AnalogElectricalSignal \subseteq Electrical \sqcap Signal$.

Determination of *subsumption* relations between classes, another standard description logic inference, can be used to organize the knowledge base for more efficient classification. A related but less common inference is induction of the *least common subsumer* of sets of classes [18–20]. The least common subsumer is the class of which each class in the given set is a subclass and for which there exists no subclass of which each class in the given set is a subclass. This is often used in conjunction with the ability to develop the *most specific class* of an object. The most specific class defines the necessary and sufficient conditions for membership in the class consisting of the given object—it is a mapping of all the properties of the given object into a class definition.

Least common subsumer inference can be used in a design repository in several ways. It can facilitate the construction of a categorization scheme in a bottom-up fashion from very specific classes; these tend to be more intuitive for users not trained in knowledge representation to create [21]. It can also be combined with most specific class induction to derive a categorization from given devices, as opposed to developing one *a priori*. The novelty of this capability in this domain is the ability to provide automatically generated, sophisticated categories for browsing/manual search, knowledge discovery of similarities, and a form of searching where the query is placed into

the context of the hierarchy and the user can browse through relationships to existing models, rather than being presented solely with items which match exactly.

Figure 4(c) shows the most specific class for the cadmium sulfide cell representation in \mathcal{EL} , which consists of conjunctions and existential restrictions³. Figures 4(a) through 4(d) depict other devices and their function and flow diagrams. Most specific classes for these are shown in Figure 4(e). By inferring least common subsumers for these classes, it is possible to automatically construct a hierarchy of induced class descriptions. Classes in the hierarchy generated from these devices include at the top level $\exists function.Measure$ and $\exists function.Produce$, provided that $Detect \subset Measure$, partitioning the devices into sensors and effectors. At the other end of the hierarchy, the most specific classes of *CDS-Cell-Sensor* and *Phototransistor* will have been inferred to be subclasses of an induced class identical to the two classes, a result of their identical models and denoting a great deal of similarity between the two devices.

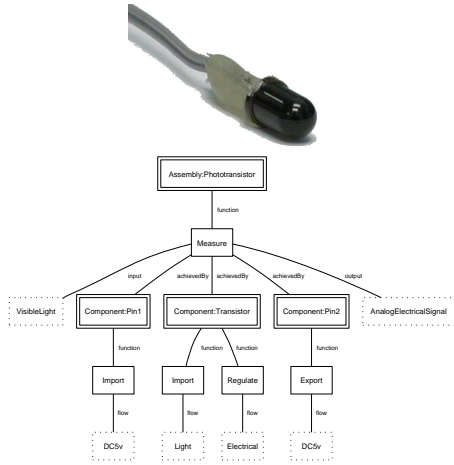
4 The Role of the Semantic Web

Utilizing current Semantic Web technology, design representations can be embedded inside Web resources. Figure 5(a) gives the source for Figure 3(a), encoded using the Resource Description Framework (RDF) [22], DARPA Agent Markup Language (DAML) [23], and the DAML version of the ontology described in the previous section. In this form it can be cleanly incorporated into webpages such as in Figure 5(b).

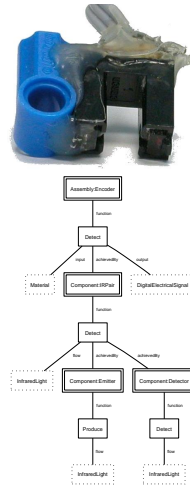
One feature of this capability is that the data sources are readily available for consumption by a wide variety of Web users. Existing search engines and directory services may index and retrieve them using standard techniques. Human users may come upon them through links from other webpages, with or without such markup. Importantly, they will be accessible to many different kinds of repositories distributed across the Internet. The information present in the markup can be easily read into a database-based system, which may offer the best performance for classification and applying *a priori* categorizations, as well as into description logic-based repositories or other systems with an enhanced set of services, such as automatically inducing categorizations.

Another feature is that the data sources can be easily published by any organization or individual with an interest in making their designs available for human Web traffic and multiple search engines, databases, and design repositories. This may enable: improved communication and resource utilization between potentially geographically distributed design groups on a company intranet; enhanced search and navigation capabilities of catalogs and listings for consumers looking for products, possibly through third-party portal sites leveraging the semantic markup to achieve increased effectiveness and coverage; design repository support for a wide range of individuals such as hobbyists and students with an interest in maintaining a collection of their own designs or those produced within an entire community of users.

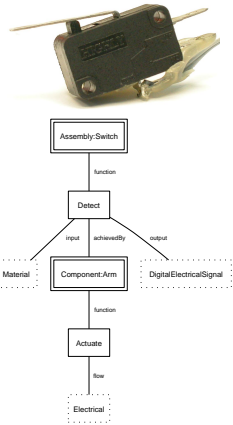
³ In more expressive description logics it might not be.



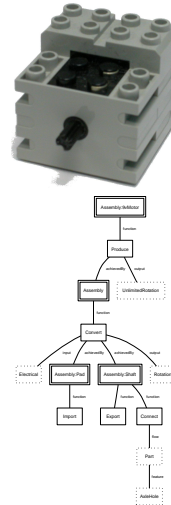
(a) Phototransistor.



(b) Break-Beam.



(c) Switch.



(d) 9V Motor.

$Phototransistor \equiv Assembly \sqcap \exists function. [Measure \sqcap \exists input. VisibleLight \sqcap \exists achievedBy. [Component \sqcap \exists function. [Import \sqcap \exists flow. DC5v]] \sqcap \exists achievedBy. [Component \sqcap \exists function. [Import \sqcap \exists flow. Light] \sqcap \exists function. [Regulate \sqcap \exists flow. Electrical]] \sqcap \exists achievedBy. [Component \sqcap \exists function. [Export \sqcap \exists flow. DC5v]] \sqcap \exists output. AnalogElectricalSignal].$
 $Encoder \equiv Assembly \sqcap \exists function. [Detect \sqcap \exists input. Material \sqcap \exists achievedBy. [Component \sqcap \exists function. [Detect \sqcap \exists flow. InfraredLight] \sqcap \exists achievedBy. [Component \sqcap \exists function. [Produce \sqcap \exists flow. InfraredLight] \sqcap \exists achievedBy. [Component \sqcap \exists function. [Detect \sqcap \exists flow. InfraredLight]]] \sqcap \exists output. DigitalElectricalSignal].$
 $Switch \equiv Assembly \sqcap \exists function. [Detect \sqcap \exists input. Material \sqcap \exists achievedBy. [Component \sqcap \exists function. [Actuate \sqcap \exists flow. Electrical]]] \sqcap \exists output. DigitalElectricalSignal].$
 $9vMotor \equiv Assembly \sqcap \exists function. [Produce \sqcap \exists achievedBy. [Assembly \sqcap \exists function. [Convert \sqcap \exists input. Electrical \sqcap \exists achievedBy. [Assembly \sqcap \exists function. [Import] \sqcap \exists achievedBy. [Assembly \sqcap \exists function. [Export] \sqcap \exists function. [Connect \sqcap \exists flow. [Part \sqcap \exists feature. AxleHole]]] \sqcap \exists output. Rotation] \sqcap \exists output. UnlimitedRotation]].$

(e) Function and flow diagrams as class descriptions.

Fig. 4. Function and flow diagrams for several other devices.


```

<rdf:RDF
  xmlns:rdf = "rdf:#" xmlns:rdfs = "rdfs:#" xmlns:daml = "daml:#"
  xmlns:eng = "eng:#" xmlns:func = "func:#" xmlns:flow = "flow:#"
  xmlns =
  "http://edge.ncs.drexel.edu/assembly/tests/na103/cds-cell.daml#"
  >
  <daml:Ontology rdf:about=""#>
    <daml:imports rdf:resource="eng:" />
    <daml:imports rdf:resource="func:" />
    <daml:imports rdf:resource="flow:" />
  </daml:Ontology>
  <eng:Assembly rdf:about=""#CDS-Cell-Sensor">
    <eng:function><func:Measure>
      <eng:input><flow:VisibleLight /></eng:input>
      <eng:achievedBy><eng:Component rdf:about=""#Pin1">
        <eng:function><func:Import>
          <eng:flow><flow:IC5V /></eng:flow>
        </func:Import></eng:function>
      </eng:Component></eng:achievedBy>
      <eng:achievedBy><eng:Component rdf:about=""#Cell">
        <eng:function><func:Import>
          <eng:flow><flow:Light /></eng:flow>
        </func:Import></eng:function>
      </eng:Component></eng:achievedBy>
      <eng:achievedBy><eng:Component rdf:about=""#Pin2">
        <eng:function><func:Regulate>
          <eng:flow><flow:Electrical /></eng:flow>
        </func:Regulate></eng:function>
      </eng:Component></eng:achievedBy>
      <eng:achievedBy><eng:Component rdf:about=""#Pin3">
        <eng:function><func:Export>
          <eng:flow><flow:IC5V /></eng:flow>
        </func:Export></eng:function>
      </eng:Component></eng:achievedBy>
      <eng:output><flow:AnalogElectricalSignal /></eng:output>
    </func:Measure></eng:function>
  </eng:Assembly>
</rdf:RDF>

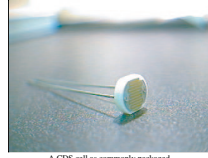
```

(a) RDF/DAML source for diagram in Figure 3(a).

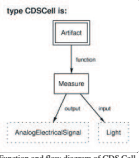
CDS Cell Light Sensor

One of the most common items used on small hobby or educational robots are light sensors. With even the simplest of uses, they enable the robot to perform tasks such as navigating towards a light, hiding in dark corners, following other robots, etc. More advanced uses permit following lines and detecting obstacles.

The most common of such sensors fall into two categories: photoresistors and phototransistors. One more particular type of the former are made of Cadmium Sulfide cells. A picture is presented below. These are commonly available from Radio Shack or from any electronic components catalog pretty cheaply.



type CDSCell is:



Function and flow diagram of CDS Cell.

A CDS cell as commonly packaged.

CDS Cells are photoresistive light sensors. When no light is present their impedance is extremely high, and conversely very low when no light is present. In contrast to phototransistors, these cells generally don't seem to have as large a range of values between the two extremes of light and dark. These cells also have a much slower reaction time in response to changes in light as they have a large memory effect.

These sensors are straightforward to wire. They're bidirectional, so simply connect one leg to your sensor input pin and the other to ground. Follow this [link](#) for a discussion on connecting phototransistors to a HandyBoard. These CDS Cells connect in the same fashion except you don't have to worry about which leg goes to which pin.

[Note: This page is a small demonstration of marking up pages about robots and their components. You can click on the "DAML Backer" link to see the marking, which corresponds to the function and flow diagram above. The diagram isn't truly necessary as it currently has no bearing on the text. I note that the markup in a word form—a class-oriented view—which I think will be useful in practice. Instead, the device will probably be modeled as an instance in any actual system we develop. The markup's very simple currently. Some of the interesting things to add is the additional information present on the text, such as the note about its response time and notes on connecting it to other devices.]

<http://www.ncs.drexel.edu/~joe/kepen> updated: Thursday, 19-02-2003 19:28

(b) Webpage with Figure 5(a) embedded in the source.

Fig. 5. Function and flow model incorporated into Web resource for cadmium sulfide cell.

5 Conclusions

This paper has briefly overviewed work on constructing design repositories for the Semantic Web. Applications of description logic inferences to achieve novel repository capabilities have been introduced. The motivations for placing such repositories on the Semantic Web have also been described. A planned future testbed of these ideas is implementation in support of a university course on small mobile robotics [24]. Current work includes the addition of non-standard description logic inferences to DAML-JessKB, a DAML-native description logic reasoner developed for this project [25], to demonstrate the utility of such inferences as applied to design repositories.

Acknowledgements: This work supported in part by National Science Foundation (NSF) CAREER Award CISE/IIS-9733545 and Office of Naval Research (ONR) Grant N00014-01-1-0618. Additional support by Honeywell FM&T, AT&T Labs, Bentley Systems and Lockheed Martin, Naval Electronics and Surveillance Systems. All opinions, findings, and conclusions expressed in this material are those of the author(s) and not necessarily those of the supporting organizations.

References

1. Ullman, D.G.: The Mechanical Design Process. McGraw-Hill, Inc. (1997)
2. Szykman, S., Bochenek, C., Racz, J.W., Senfaute, J., Sriram, R.D.: Design repositories: Engineering design's new knowledge base. IEEE Intelligent Systems **15** (2000) 48–55

3. Szykman, S., Sriram, R.D., Regli, W.C.: The role of knowledge in next-generation product development systems. *Journal of Comp. and Inf. Science in Engineering* **1** (2001) 3–11
4. Fowler, J.E.: Variant design for mechanical artifacts: A state-of-the-art survey. *Engineering with Computers* **12** (1996) 1–15
5. Lee, K., Andrews, G.: Inference of the positions of components in an assembly: part 2. *Computer Aided Design* (1985) 20–24
6. Hoffmann, C.M., Joan-Arinyo, R.: Symbolic constraints in constructive geometric constraint solving. In: *Journal of Symbolic Computation*. (1997) 287–300
7. Anantha, R., Kramer, G.A., Crawford, R.H.: Assembly modelling by geometric constraint satisfaction. *Computer-Aided Design* **28** (1996) 707–722
8. Pahl, G., Beitz, W.: *Eng. Design—A Systematic Approach*. 2nd edn. Springer (1996)
9. Szykman, S., Racz, J.W., Sriram, R.D.: The representation of function in computer-based design. In: *ASME Design Engineering Technical Conferences, 11th International Conference on Design Theory and Methodology*, New York, NY, USA, ASME, ASME Press (1999)
10. Hirtz, J., Stone, R., McAdams, D., S, S., Wood, K.: Evolving a functional basis for engineering design. In: *ASME Design Engineering Technical Conferences, 13th conference on Design Theory and Methodology*, New York, NY, USA, ASME, ASME Press (2001)
11. Kirschman, C., Fadel, G., Jara-Almonte, C.: Classifying functions for mechanical design. In: *ASME Design Engineering Technical Conferences, 8th conference on Design Theory and Methodology*, New York, NY, USA, ASME, ASME Press (1996)
12. Faltings, B.: Qualitative kinematics in mechanisms. *A.I Journal* **44** (1990) 89–119
13. Kuipers, B.: Commonsense reasoning about causality: Deriving behavior from structure. *Artificial Intelligence Journal* **24** (1984) 169–204
14. Forbus, K.: Qualitative process theory. *Artificial Intelligence Journal* **24** (1984) 85–168
15. Subramanian, D., Wang, C.: Kinematic synthesis with configuration spaces. In: *Proceedings of Qualitative Reasoning 93*,. (1993) 228–239
16. Joskowicz, L., Neville, D.: A representation language for mechanical behavior. *Artificial Intelligence in Engineering* (1996) 109–116
17. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: *The Description Logic Handbook*. Cambridge University Press (2002)
18. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In Rosenbloom, P., Szolovits, P., eds.: *Proceedings of the Tenth National Conference on Artificial Intelligence*, Menlo Park, California, AAAI Press (1993) 754–761
19. Baader, F., Küsters, R., Molitor, R.: Computing least common subsumers in description logics with existential restrictions. In Dean, T., ed.: *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, Morgan Kaufmann (1999) 96–101
20. Küsters, R., Molitor, R.: Computing least common subsumers in $\mathcal{AL}\mathcal{EN}$. In Nebel, B., ed.: *Seventeenth IJCAI*, Morgan Kaufmann (2001) 219–224
21. Brandt, S., Turhan, A.Y.: Using non-standard inferences in description logics - what does it buy me? In: *KI Workshop on Applications of Description Logics*. (2001)
22. W3C: Resource Description Framework (RDF) model and syntax specification. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/> (1999)
23. DARPA: DAML march 2001 specifications (DAML+OIL). <http://www.daml.org/2001/03/daml+oil-index> (2001)
24. Greenwald, L.G., Kopena, J.B.: On achieving educational and research goals with small, low-cost mobile robot platforms. *IEEE Robots and Automation, Special Issue on Robotics in Education* (To appear 2003)
25. Kopena, J.B., Regli, W.C.: DAMLJessKB: A tool for reasoning with the semantic web. In: *International Semantic Web Conference*. (To appear 2003)

Mediation of XML Data through Entity Relationship Models

Irini Fundulaki^{1*} and Maarten Marx^{2**}

¹ Bell Laboratories, Lucent Technologies, USA and
INRIA-Rocquencourt, France.

`fundulaki@research.bell-labs.com`

² Institute for Logic Language and Computation,
University of Amsterdam, The Netherlands.
`marx@science.uva.nl`

Abstract. This paper describes an approach for the querying of heterogeneous XML resources using an ontology-based mediator. Here an ontology is an Entity-Relationship schema defined independently of the schemas of the data sources. The sources are described to the mediator by means of mapping rules as in the Local-As-View approach to data integration. User queries are conjunctive queries formulated in terms of the ontology, and answers to these queries are obtained by rewriting them to XQuery expressions and evaluating these on the data sources. A formal semantics for queries is defined by interpreting XML sources into ER models. As there can be many such interpretations, a certain answer to a query is one which is true in all of them. We describe the rewriting algorithm and we show its completeness and correctness with respect to the given semantics. We also give an algorithm for producing a canonical model of the ontology and the interpreted data sources. It is shown that the certain answers can also be obtained by evaluating the query to just this one model.

1 Introduction

XML [1] is becoming the de-facto standard for data representation and exchange of Web data and plays an essential role in the deployment of the *Semantic Web* whose goal is “to develop enabling technologies and standards to support richer discovery, data integration, navigation and automation of tasks” [31]. In such an environment where the sources are autonomous and heterogeneous, data integration is a critical issue. The goal there is to enable users to query the data of heterogeneous sources as if it resides in a single source, which is exactly what a data integration system does. The fact that the sources concern a *restricted domain of interest* is crucial for the successful deployment of data integration systems. Their backbone is a *global schema* which describes the basic notions in

* This work is funded by a scholarship from the French National Institute for Research in Computer Science and Control (INRIA).

** Research funded by NWO grant 612.000.106.

the domain. Appropriate *mappings* between the global schema and the schemas of the sources are defined to describe the latter to the former. User queries are formulated in terms of the global schema and the answers are obtained by accessing the source data.

There are several questions that need to be considered when one is constructing a data integration system: (i) is the global schema defined out of the sources schemas (e.g. federated architecture [22]), or is independent thereof (e.g. defined by an authority in the domain); (ii) is query answering done *on the fly* by translating the user query into queries expressed in the sources' schemas (query mediation paradigm [34, 26]), or by evaluating it against the database which has been constructed out of the source data (e.g. data warehouse paradigm [33]). In an evolving environment like the Web, defining the global schema out of the sources schemas may lead to significant overload of schema maintenance: every insertion/deletion of a new source or modification of an already integrated source's schema, leads to modification(s) of the global schema. Besides that, the global schema can become quite unintelligible, due to the idiosyncrasies present in the different local schemas. An independently defined global schema (an ontology) seems most appropriate for the integration of web data, and this line is followed in the paper. In an environment where data is changing rapidly, keeping the data warehouse "up-to-date" is not the easiest of tasks, so we choose to study mediation.

Another dimension of a data integration system is the approach used to define the mappings between the global schema and the sources schemas. There are two prevailing approaches: Global-As-View (GAV) and Local-As-View (LAV) [23]. In the former, the global schema relations are defined as views of the sources relations. The mappings are conjunctive queries³ which specify how to obtain the tuples for the former: their head involves one global schema relation, and their body involves a conjunction of source relations. Queries are formulated in terms of the global schema relations and their translation is done by substituting the global schema relations by their definitions⁴. In the latter, a source is described as a (materialized) view of the global schema relations [24]. The mappings (also conjunctive queries) have the inverse direction of those in the GAV approach. Query translation in this context is known as *querying rewriting using views* [24]. The drawback of the first approach is that the global schema depends on the sources schemas, while in the second, it is defined independently of them. But, query translation in LAV is more complicated than in GAV. In the simple case of conjunctive queries, it has been proved NP-hard [24].

A number of algorithms have been proposed in the literature for the latter approach. Authors in [25, 29, 28, 15, 17, 21] consider query rewriting for conjunctive queries and relational views. Algorithms in [25, 29] are characterized as *bucket-based* where the idea is to match each query subgoal with the body of the view definitions. Authors in [15, 17] follow a different approach where they produce

³ We use the term conjunctive query to refer to *rule-based conjunctive query* as defined in [2].

⁴ This substitution is called query unfolding.

a set of *rewriting rules* out of the view definitions. Query rewriting is done by matching the body of the rules with the body of the query. Rewriting in a different framework is done in [8, 20, 7] where *regular path* queries and views are considered.

In this paper we consider query rewriting in the framework proposed in [4, 18]. A mediator-based architecture for the integration of XML resources is proposed where the LAV approach is used to describe the sources to the global schema, the latter defined independently of the schemas of the former. The global schema is an *ontology* which incorporates the basic features of the *Entity Relationship (ER)* model [9] (which can be easily represented as an object-oriented or as an RDF schema [30]). The principal contributions of this work in the domain of XML data integration are:

- the identification of a reasonable subset of the web data integration problem in which mediation is a feasible alternative to data warehousing;
- the use of a rich conceptual schema which considers inverse roles and global keys, instead of a relational one as the global schema of the mediator;
- an expressive mapping language to describe a source’s schema to the global schema. All previous approaches consider that a source is described as a conjunctive query over the global schema relations. In [4] *mapping* rules which associate XML fragments (expressed by XPath [10] expressions) to ontology paths are used.

This approach has several advantages. First ontologies are more expressive than XML DTDs or Schemas due to the presence of (i) subsumption relations and (ii) typed binary relationships between entities. In XML neither of them exist, and the only type of relationships between nodes is the parent/child and attribute relationships. The ID/IDREF attribute mechanism can be used to relate two nodes but these relationships are untyped. Second, the use of XPath in the mapping language allows one to describe arbitrary XML structures, and in addition to associate specific semantics (expressed in the ontology) to the relationships between XML nodes. We will not discuss here the benefits of using an ER schema as the global schema. Detailed discussion on this issue can be found in [3].

In this context, the result of the rewriting for a query q and for a set of sources S is the union of all the rewritings of q for each source s . The idea behind rewriting q for s is simply to match the query variable binding paths to the ontology paths of the rules. The proposed framework along with the query rewriting algorithms has been implemented in the *ST_YX* prototype [19].

A similar but simpler approach has been undertaken in [13]. The global schema is a node-labeled tree (called abstract DTD) hence there is no notion of subsumption relationship and parent/child is the only relationship between nodes. XML sources are described by concrete DTDs (also node-labeled trees). The mapping language is much simpler than in [4]: only the `child` axis of XPath is used where absolute abstract paths are mapped to absolute concrete paths⁵.

⁵ Paths are called absolute since they are specified *only* from the root node of the abstract/concrete DTDs.

Given a user query expressed in terms of the abstract DTD, their query rewriting algorithm tries to find the concrete paths that match the abstract paths.

In this paper we examine the completeness and soundness of the rewriting algorithm in [4] for a query q and a source s following a different approach for the manipulation of ontology paths in the models of the ontology. In this new setting we started by examining the proposed framework and in order to achieve our goal we had to consider the following restrictions: (i) the ontology contains no inverse roles, (ii) no global keys are considered and (iii) only the `child` and `attribute` axis of XPath are used in the mapping rules. Hence, we had to contemplate with a poorer mapping language than the original one to prove completeness. The conclusion that we can draw is that if one wants to go for expressive power, then one must pay the price of incompleteness.

The paper is organized as follows: in Section 2 we present the approach followed in [4] using a cultural example. Section 3 gives the formal definitions for the ontology, the mapping rules, queries and answers. In Section 4 we show how a model for the ontology is built in which the XML sources are interpreted using a Tableau System. We also show how this model can be used to yield all certain answers to queries. Section 5 presents the query rewriting algorithm and proves its completeness and correctness. Conclusions are given in Section 6. Proofs of all statements are provided in the Appendix.

2 Overview of the approach through a cultural example

We give in this section an overview of the approach in [4] using an example from the cultural domain. An XML resource is considered to contain a number of XML documents which conform to a unique XML DTD⁶. The upper side of Fig. 1 shows an XML DTD describing painters and their paintings for source <http://www.paintings.com>. Each painter (element `Painter`, line 2) is associated with one or more paintings (element `Painting`). A painter has a name (attribute `name`, line 3). A painting has a title (attribute `title`, line 5). Fig. 1 presents an XML document that conforms to the DTD.

The backbone of the mediator is an *ontology* which incorporates the basic features of the *Entity Relationship (ER)* model [9]. The terms *concept* and *role* are used to name entities and relationships respectively. Concepts are related to each other with the *isa* subsumption relation. We use the same ontology as the one used in [4] as a reference example, which is inspired from the ICOM/CIDOC Reference Model [14]⁷.

The ontology is depicted in the graphical notation of ER schemas, with the addition of *isa* arcs and is shown in Fig. 2. It describes concepts such as `Actor`,

⁶ Although this may look like a simplification from the setting with a number of heterogeneous sources, on our level of description it is not: using appropriate renaming every subset of XML documents with their own DTD can be equivalently transformed into one source with one DTD.

⁷ The ICOM/CIDOC model has been defined and used for the documentation of cultural information. Several cultural authorities have participated in this effort whose basic purpose was to provide a single schema to exchange their data.

```

1. <!ELEMENT Collection (Painter+)>
2. <!ELEMENT Painter (Painting+)>
3. <!ATTLIST Painter name CDATA #REQUIRED>
4. <!ELEMENT Painting EMPTY>
5. <!ATTLIST Painting title CDATA #REQUIRED>

<Collection>
  <Painter name='Rembrandt'>
    <Painting title='de Nachtwacht' />
    <Painting title='de Staalmeesters' />
  </Painter>
  <Painter name='Vermeer'>
    <Painting title='de Brief' />
    <Painting title='het Melkmeisje' />
  </Painter>
</Collection>

```

Fig. 1. XML DTD and document for source <http://www.paintings.com>.

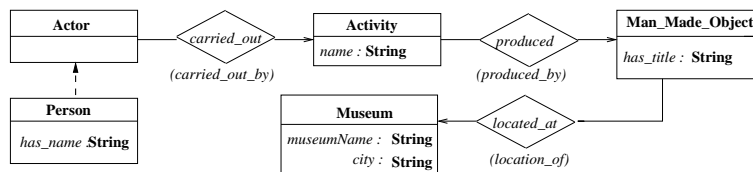


Fig. 2. An Ontology for Cultural Artifacts.

R_1 : <code>http://www.paintings.com/Collection/Painter</code>	as u_1	\rightarrow Person
R_2 : <code>u_1/@name</code>		\rightarrow has_name
R_3 : <code>u_1/Painting</code>	as u_3	\rightarrow carried_out/produced
R_4 : <code>u_3/@title</code>		\rightarrow has_title

Fig. 3. Set of Mapping Rules for source <http://www.paintings.com>.

its subconcept Person, Activity, Man_Made_Object, and Museum. The concepts are connected using binary roles such as `carried_out`, `produced` and `located_at`. The fact that an actor (instance of concept Actor) performs an activity (instance of concept Activity) to produce a man made object (instance of concept Man_Made_Object) is represented by roles `carried_out` and `produced`. Concepts may also have attributes: attribute `has_name` is defined in concept Person and takes its values in the atomic type String. In the ontology, each role has an inverse depicted within parenthesis.

To describe an XML resource to the ontology a set of *mapping rules* is defined. Each rule associates an XPath location path [10] to concepts, attributes and (composite) roles in the ontology (we use `'/'` to denote the composition of two roles).

The set of rules illustrated in Fig. 3 map fragments of XML documents which conform to the DTD of Fig. 1 into the ontology of Fig. 2. Each rule has a left

hand side and a right hand side. The right hand side is a concept, attribute or (composite) ontology role (referred to as *ontology paths* in the sequel). The left hand side is composed of (i) an XPath location path evaluated on some URL or on some variable and (ii) an optional variable declaration.

The first rule states that all elements returned when evaluating the XPath expression `Collection/Painter` on the root nodes of documents in URL `http://www.paintings.com` are instances of concept `Person` and this set of elements is bound to variable u_1 . The statement `'as u1'` is a variable declaration. The second rule states that values of the attribute `has_name` of `Person` are obtained from evaluating the XPath expression `@name` on instances of variable u_1 (`Painter` elements). The third rule states that x `carried_out/produced` y holds whenever x is a `Painter` element (element of the variable u_1) and y is a `Painting` element, child of x . The obtained `Painting` elements are bound to variable u_3 which is used in the last rule to obtain the values for attribute `has_title`.

The mapping rules allow one to define the semantics of XML fragments and their structural relationships in terms of the ontology. Thus, R_1 defines a subset of the extension of concept `Person`, while rule R_3 relates elements in this subset by the composite role `carried_out/produced` to a subset of the extension of `Man_Made_Object`.

Queries are conjunctive queries expressed in the vocabulary of the ontology [2]. Such queries can be presented in several formats. In more theoretical work, conjunctive queries are presented as Datalog rules. Commercial systems often employ a **select/from/where** formulation (as in SQL).

The query “Which objects are created by Vermeer” is expressed in the OQL syntax [11] and as a conjunctive query in Table 1. These two formulations differ only with respect to their notion of an answer. An answer to the conjunctive query is an instantiation of x_3 which makes the body of the rule true. The answer to the OQL query is the set of all these instantiations. In the sequel we use answer in the former sense.

Each variable in the query is bound to some ontology path (called in the sequel its *binding path*). For example, for the OQL query Q_1 , `Person` is the binding path of x_1 , `has_name` is the binding path of x_2 and finally `carried_out/produced` is x_3 's binding path.

Given a query, the rewriting algorithm proposed in [4] works as follows: given that each variable in the query is bound to some ontology path, the algorithm searches for mapping rules or concatenations thereof, that can be used to translate these paths to XPath location paths. This is done by *matching* the binding paths of the query variables against the ontology paths of the mapping rules. For example for Q_1 and for the set of mapping rules depicted in Fig. 3, we see that instances for variable x_1 are found by rule R_1 , for variable x_2 by R_2 and for variable x_3 by rule R_3 . By substituting the binding path of a query variable with the location path of the corresponding rule, we obtain query $Q_1(a)$ which can be easily translated into the XQuery [12] expression $Q_1(b)$ both shown in Table 2. For the document of Fig. 1, there are two answers to this query: `<Painting title = " de Brief"/>` and `<Painting title = " het Melkmeisje"/>`.

```

Q1: select x3
from Person x1, x1.has_name x2,
      x1.carried_out/produced x3
where x2 = "Vermeer"

```

$Q_1(x_3) :- \text{Person}(x_1), \text{has_name}(x_1) = x_2, \text{carried_out/produced}(x_1, x_3), x_2 = \text{"Vermeer"}$.

Table 1. Query Q_1 .

```

Q1(a): select x3
from http://www.paintings.com/Collection/Painter x1,
      x1./@name x2, x1./Painting x3,
where x2 = "Vermeer"
Q1(b): FOR      $x1 IN document('http://www.paintings.com')/Collection/Painter,
                  $x2 IN $x1/@name,
                  $x3 IN $x1/Painting
WHERE $x2 = "Vermeer"
RETURN $x3

```

Table 2. Queries $Q_1(a)$ and $Q_1(b)$.

3 Mapping XML resources to Ontologies

In the setting proposed in [4] several XML sources are integrated into one ER model. Whence, it is not evident what constitutes a correct answer to a query. In this section we develop a theory for interpreting XML sources into ER models and use that to define the notion of a *certain answer*. This notion originated in the context of incomplete databases and has been used for query answering in data integration [23].

Ontologies As aforementioned, ontologies incorporate the basic features of the Entity Relationship (ER) model [9]. The model considered in [4] differs from standard ER models in three respects: (i) only binary relationships (called roles) between entities are allowed, (ii) only two roles (called *source* and *target* assigning the domain and range to each relationship) are used, and (iii) attributes are *partial* rather than *total* functions.

Formally, an ontology is an 8-tuple $O = (C, R, S, A, source, target, isa, key)$, where: (i) C is a set of *concept* symbols, (ii) R is a set of *role* symbols, (iii) S is the set of *atomic types* defined in the XML Schema Datatypes document [5], (iv) A is a set of *attribute* symbols, (v) *source* is a function that assigns to roles and attributes their domain in C , (vi) *target* is a function that assigns to roles their range in C and to attributes their range in S , (vii) *isa* is a subsumption relation between concepts in C , (viii) *key*(\cdot) is a function from C to $\mathcal{P}(A)$, assigning to every concept a (possibly empty) set of its attributes. *key*(\cdot) is such that for each pair of concepts c_1, c_2 with $c_1 isa c_2$ it holds that $key(c_1) \subseteq key(c_2)$ ⁸. We use isa^* to denote the reflexive and transitive closure of the *isa* relation.

⁸ In contrast to [4] we only deal with single valued keys and with at most one key per concept. In general, concepts may have several multivalued keys as in [6]. The general

$\text{Actor}^{\mathfrak{B}} = \{p_1, p_2\}$ $\text{Person}^{\mathfrak{B}} = \{p_1, p_2\}$
 $\text{Activity}^{\mathfrak{B}} = \{a_1, a_2, a_3\}$ $\text{Man_Made_Object}^{\mathfrak{B}} = \{o_1, o_2, o_3, o_4\}$ $\text{Museum}^{\mathfrak{B}} = \emptyset$
 $\text{carried_out}^{\mathfrak{B}} = \{(p_1, a_1), (p_1, a_2), (p_2, a_3)\}$
 $\text{produced}^{\mathfrak{B}} = \{(a_1, o_1), (a_2, o_2), (a_3, o_3), (a_3, o_4)\}$ $\text{located_at}^{\mathfrak{B}} = \emptyset$
 $\text{has_title}(o_1) = \text{“de Nachtwacht”}$ $\text{has_name}(p_1) = \text{“Rembrandt”}$
 $\text{has_title}(o_2) = \text{“de Staalmeesters”}$ $\text{has_name}(p_2) = \text{“Vermeer”}$
 $\text{has_title}(o_3) = \text{“de Brief”}$
 $\text{has_title}(o_4) = \text{“het Melkmeisje”}$

Table 3. A model \mathfrak{B} for the ontology O .

The semantics of an ontology can be given by specifying which database states are consistent with the information structure represented by the ontology. Formally, a database state \mathfrak{B} for an ontology O consists of a nonempty finite set $\mathfrak{D}^{\mathfrak{B}}$ (which is disjoint from all basic domains) and a function $(\cdot)^{\mathfrak{B}}$ which interprets the symbols of the ontology: concepts c as subsets $c^{\mathfrak{B}}$ of $\mathfrak{D}^{\mathfrak{B}}$; roles r as subsets $r^{\mathfrak{B}}$ of $\mathfrak{D}^{\mathfrak{B}} \times \mathfrak{D}^{\mathfrak{B}}$; attributes a as partial functions $a^{\mathfrak{B}}$ from $\mathfrak{D}^{\mathfrak{B}}$ to the union of the basic domains; atomic types S as the corresponding basic domains $S^{\mathfrak{B}}$.

A database state \mathfrak{B} is a model for the ontology O , if (i) \mathfrak{B} interprets the concept, role, attribute and atomic type symbols of the ontology, (ii) for each pair of concepts c_1, c_2 with c_1 *isa* c_2 it holds that $c_1^{\mathfrak{B}} \subseteq c_2^{\mathfrak{B}}$, (iii) for each role r , the *domain* and *range*⁹ of r are subsets of $\text{source}(r)^{\mathfrak{B}}$ and $\text{target}(r)^{\mathfrak{B}}$, respectively, (iv) for each attribute a , for each $e \in \mathfrak{D}^{\mathfrak{B}}$, if $a(e)$ is defined then $e \in \text{source}(a)^{\mathfrak{B}}$ and $a(e) \in \text{target}(a)^{\mathfrak{B}}$ and (v) for all $x, y \in \mathfrak{D}^{\mathfrak{B}}$, whenever $x, y \in c^{\mathfrak{B}}$, for some concept c and $\{a_1, \dots, a_n\}$ is the key for c it holds that: $x = y$ iff $a_1^{\mathfrak{B}}(x) = a_1^{\mathfrak{B}}(y)$ and \dots and $a_n^{\mathfrak{B}}(x) = a_n^{\mathfrak{B}}(y)$ (i.e. \mathfrak{B} does not contain different objects with the same value for a key).

A model \mathfrak{B} for the ICOM/CIDOC ontology is shown in Table 3. The domain $\mathfrak{D}^{\mathfrak{B}}$ of \mathfrak{B} consists of the following set of elements: $\{p_1, p_2, a_1, a_2, a_3, o_1, o_2, o_3, o_4\}$. There is only one basic domain of type **String**.

Mapping rule R_3 in Fig. 3 uses the composite role **carried_out/produced**. Composite roles have proven to be very useful for mapping XML elements into the ontology [3] and can be seen as describing paths in the ontology. A role path (*rolepath*) is a composition of roles, a concept path (*conceptpath*) is a composition of a concept and a role path and finally an ontology path (*ontologypath*) is either a concept path, a role path, an attribute or a composition of a conceptpath and an attribute (the latter called *attribute path*).

rolepath	::= $r \mid \text{rolepath}/r$	for $r \in R$
conceptpath	::= $c \mid c/\text{rolepath}$	for $c \in C$
ontologypath	::= conceptpath \mid rolepath \mid $a \mid$ conceptpath/ a for $a \in A$	

case leads to no new conceptual difficulties; the only difference is that determining identity of objects is a longer process.

⁹ The domain of a role r is the set $\{x \mid \exists y.(x, y) \in r^{\mathfrak{B}}\}$. Its range is the set $\{x \mid \exists y.(y, x) \in r^{\mathfrak{B}}\}$.

The source of a role path is the source of its first element; the target of a role path is the target of its last element. A composition of two roles, or of a role and an attribute r/s is *safe* if $target(r) isa^* source(s)$. The composition of a concept and a role or attribute c/r is *safe* if $c isa^* source(r)$. An ontology path is safe if all its compositions are safe.

Examples of (safe) role paths in the ICOM ontology are `carried_out` and `carried_out/produced`. `Person/carried_out/produced` is a safe concept path.

The interpretation of ontology paths in a model of an ontology is defined as follows: For $p = r_1/\dots/r_n$ a role path, $(r_1/\dots/r_n)^{\mathfrak{B}} = r_1^{\mathfrak{B}} \circ \dots \circ r_n^{\mathfrak{B}}$.¹⁰ For $p = c/r$ a concept path, $(c/r)^{\mathfrak{B}} = \{x \mid \exists y \in c^{\mathfrak{B}} \text{ and } (y, x) \in r^{\mathfrak{B}}\}$. For $p = c/a$ with c a concept path and a an attribute, $(c/a)^{\mathfrak{B}} = \{(x, v) \mid x \in c^{\mathfrak{B}} \text{ and } a(x) = v\}$.

Mapping Rules As presented in Section 2, an XML resource is described to the ontology by means of mapping rules which associate XPath location paths to ontology paths. A *mapping rule* is an expression of the form $R : u/q \text{ as } v \rightarrow p$, or $R : u/q \rightarrow p$ where: (i) R is the rule's *label*, (ii) u is either a variable or a URL, (iii) q is an XPath location path, (iv) p is an ontology path, restricted as follows: p is an attribute only if q is of the form $@q'$ where q' is an XML attribute and u is a variable; p is a role path only if u is a variable, and a concept path only if u is a url (iv) *as v*, is a variable declaration, present only if p is a role path or a concept path. A set of mapping rules is called a *mapping*. Rules are distinguished between *relative* and *absolute*: a rule R is called absolute if it starts with a URL, relative otherwise. Concatenation of mapping rules is defined as follows: for $R_1 : a/q_1 \text{ as } v_1 \rightarrow p_1$, $R_2 : v_1/q_2 \text{ as } v_2 \rightarrow p_2$ two rules in a mapping M , their *concatenation* is the new rule $R_1/R_2 : a/q_1/q_2 \text{ as } v_2 \rightarrow p_1/p_2$. Given a mapping M , its *closure*, denoted by M^* , is the set of all rules that can be obtained from M by repeated concatenation. Its *expansion*, denoted \hat{M} , is the set of absolute rules in M^* .

Definition 1 (Well Presented Mapping). *We call a mapping well presented if it satisfies the following two conditions: (i) all ontology paths of rules in \hat{M} are safe, and (ii) if $R_1 : u/q \text{ as } v_1 \rightarrow p_1$, $R_2 : u/q \text{ as } v_2 \rightarrow p_2$ are both in \hat{M} , then $v_1 = v_2$.*

It is straightforward to check that the mapping rules in Fig. 3 are well presented.

From this point on, and w.l.g. we assume that a source is an XML document specified by a URL u and a mapping M in which u appears as the only URL.

As described in Section 2, a mapping M from an XML source s to an ontology O populates the concepts, roles and attributes of O . This is done formally by an *interpretation* function which maps the elements and values in an XML document to a model \mathfrak{B} of the ontology.

Definition 2 (Interpretation Function). *Let (u, M) be a source, d the XML document specified by u and \mathfrak{B} a model for an ontology O . An interpretation of*

¹⁰ $r_1^{\mathfrak{B}} \circ r_2^{\mathfrak{B}}$ is defined as follows: $r_1^{\mathfrak{B}} \circ r_2^{\mathfrak{B}} = \{(x, y) \mid \exists z. (x, z) \in r_1^{\mathfrak{B}} \text{ and } (z, y) \in r_2^{\mathfrak{B}}\}$.

(u, M) in \mathfrak{B} is a function f which: (i) sends elements of d to $\mathfrak{D}^{\mathfrak{B}}$; (ii) sends attribute values of d to the basic domains of \mathfrak{B} of the same type; and (iii) respects all rules in the expansion of M in the following sense¹¹: Let $R_1 : u/q$ as $v \rightarrow p$ be an absolute and $R_2 : v/q'$ as $v' \rightarrow r$ a relative mapping rule. We say that f respects R_1 if $\{f(e) \mid e \in u/q\} \subseteq p^{\mathfrak{B}}$. We say that f respects R_1/R_2 if f respects R_1 and for all $e \in u/q$, for all $e' \in e/q'$, $(f(e), f(e')) \in r^{\mathfrak{B}}$.

In the sequel it will be convenient to assume that the database state in which a source is interpreted has the set of elements of the XML document as constant symbols. Then we can use the XML element e to refer to the database object $f(e)$. So given an interpretation f of a source (u, M) in a database state \mathfrak{B} , we assume that \mathfrak{B} contains the set of elements of the XML document specified by u as constant symbols and the interpretation is such that $f(e) = e^{\mathfrak{B}}$. Often we just equate e with $e^{\mathfrak{B}}$.

Example 1. We will interpret the XML document of Fig. 1 into the model of the ICOM ontology given in Table 3, in such a way that it respects the mapping rules R_1 – R_4 . The function f is shown in Fig. 4. The XML document has two painter elements, one for *Rembrandt*, and one for *Vermeer*. f maps the first to p_1 , and the second to p_2 . The four painting elements are mapped to the four objects o_1 – o_4 . The values of the XML attributes `name` and `title` are mapped to the identical strings in the basic domain of `String`. It is not hard to check that f is indeed an interpretation.

```
f(<Painter name='Rembrandt'>
  <Painting title='de Nachtwacht'>/>
  <Painting title='de Staalmeesters'>/>
</Painter>)=p1
f(<Painter name='Vermeer'>
  <Painting title='de Brief'>/>
  <Painting title='het Melkmeisje'>/>
</Painter>)=p2

f(<Painting title='de Nachtwacht'>/>)=o1
f(<Painting title='de Staalmeesters'>/>)=o2
f(<Painting title='de Brief'>/>)=o3
f(<Painting title='het Melkmeisje'>/>)=o4
```

Fig. 4. Interpreting the XML document of Fig. 1 into the ontology model of Table 3.

Queries and Answers Queries are conjunctive queries expressed in the vocabulary of the ontology [2] extended with the ontology paths and some fixed set

¹¹ In this paper we use the expression $e' \in e/q'$ to denote that element e' is obtained when evaluating XPath location path q' on element e . We use this expression instead of $e' \in \llbracket q' \rrbracket(e)$ as defined in [32].

of comparison predicates defined over the basic domains. Formally, given an ontology O , a conjunctive query is an expression of the form:

$$q(\bar{x}) : - \bigwedge c(x), \bigwedge r(x, x'), \bigwedge a(x) = x', \bigwedge x\theta n.$$

where \bar{x} is a sequence of variables all occurring in the body of the query, every c is a concept path, every r a role path, every a an attribute, and every $x\theta n$ a comparison predicate where x is a variable, n is a value and θ is one of $\{=, <, >, \leq, \geq\}$.

Conjunctive queries in which no composite concepts or roles occur, are called *atomic*. Conjunctive queries can be easily transformed into queries in an OQL-like syntax¹². We recall here the definition of an answer to a conjunctive query to a model $\mathfrak{D}^{\mathfrak{B}}$ [2]. Let $q(x_1, \dots, x_n)$ be a conjunctive query with variables x_1, \dots, x_n and y_1, \dots, y_m . Then a_1, \dots, a_n is an answer to $q(x_1, \dots, x_n)$ if $\mathfrak{D}^{\mathfrak{B}} \models \exists y_1 \dots \exists y_m q(a_1, \dots, a_n)$, where \models is the standard first order satisfaction relation.

Finally we come to the central definition in this paper. Let O be an ontology, (u, M) a source mapping elements of the documents in u into O , and $q(\bar{x})$ a conjunctive query expressed in the vocabulary of the ontology. We want to define an intuitively reasonable notion of a correct answer to $q(\bar{x})$ given O and (u, M) . This notion then will determine whether an algorithm which produces the answers is both correct and complete. The definition below captures the idea that something is an answer to a query if there is no countermodel. In the literature on data integration, these are called the *certain answers* [23]. More formally, a list of XML elements and attribute values \bar{a} of u is an answer to the query $q(\bar{x})$ if there is no model \mathfrak{B} for O in which the source is interpreted and where $\mathfrak{B} \not\models q(\bar{a})$. This is usually formulated positively as follows:

Definition 3 (Certain answer). *Let (u, M) be a source, O an ontology and $q(\bar{x})$ a conjunctive query in the language of O . M maps the elements of u into O . Let \bar{a} be a list of XML elements and attribute values coming from the source. We say that \bar{a} is a certain answer to the query $q(\bar{x})$ posed to (u, M) , if for each model \mathfrak{B} for O , for each interpretation f of (u, M) into \mathfrak{B} it holds that $\mathfrak{B} \models q(f(\bar{a}))$.*

4 Data Warehousing

Given an ontology O and a source (u, M) which is mapped to O we can construct a model of O using a tableau system. This model can be seen as a data warehouse. In the previous section we have seen that there are many models of O in which a source can be interpreted. A certain answer was an answer which is true in all these interpretations. Rather surprisingly now, all certain answers to a

¹² Let $q(\bar{x})$ be a conjunctive query. Query $q(\bar{x})$ can be transformed into query q' in an OQL-like syntax as follows: q' 's **select** clause contains the variables in \bar{x} ; q' 's **from** clause contains $\bigwedge c(x), \bigwedge r(x, y), \bigwedge a(x, y)$ ($a(x, y)$ for the expression of the form $a(x) = y$) and q' 's **where** clause contains the conjunction of the comparison predicates.

conjunctive query can be obtained by evaluating the query to just this one constructed model (Theorem 3). In the context of data exchange [16] such a model has been called the *universal canonical solution*. We use the same terminology. To have a constructive way of producing all certain answers turns out very useful in the next section: it will be straightforward to establish completeness of the rewriting algorithm on the basis of the tableau system.

The tableau rules are given in Table 4¹³. They can be grouped according to their function. The mapping rule rules implement the meaning of the mapping rules. The domain, target, isa and global key rules derive facts from the structure of the ontology. The composition, inverse and equality rules specify the logic behind these operations. The comparison rule just lists facts true of the basic domains. Given that a source is finite, the tableau construction reaches a fixed point in which application of each rule only duplicates lines already on the tableau¹⁴. Thus

Theorem 1. *Constructing a tableau with the tableau rules in Table 4 terminates.*

If we apply the tableau rules until no rule yields new information, we end up with a complete description of a model of the ontology in which the source is interpreted (Theorem 2). In the description of the model we use the XML elements of the source and parameters to denote elements of the domain of the model.

Theorem 2. *Let T be a complete tableau expansion for the ontology O and the source (u, M) mapped to O . Then there exists a model \mathfrak{B}_T such that: (i) \mathfrak{B}_T is a model for O , (ii) (u, M) is interpreted in \mathfrak{B}_T , (iii) the domain of \mathfrak{B}_T consists of the set of parameters and XML elements occurring in T factored by the relation $\{(x, y) \mid x = y \text{ occurs in } T\}$, (iv) for all concepts C , relations R and attributes a ,*

$$\mathfrak{B}_T \models C(\bar{e}_1) \text{ iff } C(e_1) \text{ on the tableau} \quad (1)$$

$$\mathfrak{B}_T \models \bar{e}_1 R \bar{e}_2 \text{ iff } e_1 R e_2 \text{ on the tableau} \quad (2)$$

$$\mathfrak{B}_T \models a(\bar{e}_1) = n \text{ iff } a(e_1) = n \text{ on the tableau,} \quad (3)$$

where \bar{e} denotes the equivalence class of e .

Proofs of all results are provided in the Appendix. The tableau rules not only give us a model, we can also use them as an algorithm to yield answers to queries. Here we present a primitive way of doing that. A more efficient calculus following the GAV query unfolding approach is described in the full version of this paper

¹³ When constructing this model, the order of XML elements in the actual XML document is lost.

¹⁴ One has to restrict the application of the concept path and composition rules to exactly one time for each occurrence of the enumerator. Because the parameter is new, applying these rules more often yields no extra information.

Absolute Mapping rule	$\frac{R_i : u/q \text{ as } v_i \rightarrow C \text{ is an absolute rule in } \hat{M}}{C(e) \text{ for all } e \in u/q}$
Relative Mapping rule I	$\frac{R_i : u/q \text{ as } v_i \rightarrow C \text{ is an absolute rule in } \hat{M} \quad R_j : v_i/p \text{ as } v_j \rightarrow R \text{ is a relative rule in } M^*}{eRe' \text{ for all } e \in u/q \text{ and } e' \in e/p}$
Relative Mapping rule II	$\frac{R_i : u/q \text{ as } v_i \rightarrow C \text{ is an absolute rule in } \hat{M} \quad R_j : v_i/@n \rightarrow a \text{ (for } a \text{ an attribute) is a relative rule in } M}{a(e) = e/@n \text{ for all } e \in u/q}$
Isa rule	$\frac{C(e) \quad C \text{ isa } D}{D(e)}$
Domain rules	$\frac{eRe' \quad \text{source}(R) = C \quad a(e) \text{ is defined} \quad \text{source}(a) = C}{C(e)} \quad \frac{a(e) \text{ is defined} \quad \text{source}(a) = C}{C(e)}$
Target rules	$\frac{eRe' \quad \text{target}(R) = C}{C(e')} \quad \frac{a(e) \text{ is defined} \quad a(e) \notin \text{target}(a)}{\mathbf{ABORT CONSTRUCTION}}$
Concept path rule	$\frac{C/R(e)}{C(n)} \quad n \text{ is a new parameter in the tableau expansion.}$ nRe
Composition rule	$\frac{eR/Se'}{eRn} \quad n \text{ is a new parameter in the tableau expansion.}$ nSe'
Inverse rule	$\frac{eR^{-1}e'}{e'Re}$
Global key rule	$\frac{\text{key}(C) = \{a_1, \dots, a_n\}, C(e), C(e'),}{a_1(e) = a_1(e'), \dots, a_n(e) = a_n(e')}{e = e'}$
Reflexivity rule	$\frac{}{e = e}$
Replacement rule	$\frac{\phi(e) \quad e = e'}{\phi(e')}$
Comparison predicate rule	$\frac{}{n\theta m} \quad \text{for every true statement } n\theta m \text{ (} n, m \text{ elements of some basic domain occurring on the tableau.}$

Table 4. Tableau rules.

[27]. The following completeness theorem however is very convenient for proving completeness results later on.

Let $\phi(\bar{x})$ be an atomic query. We say that the tableau system gives \bar{e}_1 as an answer to $\phi(\bar{x})$ asked to ontology O and source (u, M) if there exists parameters and XML elements \bar{e}_2 such that every conjunct $\phi_i(\bar{x}, \bar{y})$ occurs in the tableau with \bar{e}_1 and \bar{e}_2 substituted for \bar{x} and \bar{y} , respectively.

Theorem 3 (Soundness and completeness of the tableau system). *Given an ontology O , source (u, M) and an atomic conjunctive query $\phi(\bar{x})$, the tableau system gives \bar{e} as an answer to $\phi(\bar{x})$ if and only if \bar{e} is a certain answer to $\phi(\bar{x})$.*

We end with two technical lemmas which will be used in the next section. The first lemma can be seen as a reason for asking that the paths in the mapping rules are safe. In that case the domain rule of the tableau system becomes redundant.

Lemma 1. *Let T be a tableau for the ontology O and the source (u, M) mapped to O . Let the paths in the mapping rules of \hat{M} be safe. Then every occurrence $C(e)$ in T can be obtained without an application of the domain rule.*

The next lemma specifies a situation in which tableau proofs are particularly simple, and in which the universal canonical solution has the convenient property of being a tree.

Lemma 2. *Let O be an ontology without inverse roles and without global keys. Let (u, M) be a source in which the paths in the mapping rules of \hat{M} are safe. Then*

- (i) *for each conjunctive query $q(\bar{x})$, \bar{e} is a certain answer to $q(\bar{x})$ if and only if there is a tableau proof for $q(\bar{e})$ which only uses the mapping rule rules, isa, target, concept path and composition rules;*
- (ii) *the universal canonical solution is a tree.*

5 Lav Approach : Query Rewriting

In this section we discuss the completeness of the rewriting algorithm in [4] in our context. As in [4], we consider the rewriting of *tree queries* without joins between variables and we use an OQL-like syntax to express them¹⁵.

The algorithm in [4] needs only rewrite the conjuncts in the **from** clause of the query. We demonstrate the idea using query Q_1 shown in Table 1 and the mapping rules of Fig. 3.

In short, the algorithm examines the query variables and finds the mapping rules which provide answers for them. Query variables are arranged in preorder, and the algorithm considers a variable, after it has examined its parent. The root variable x_1 is examined first. The algorithm looks for absolute mapping rules

¹⁵ A tree query is a conjunctive query $q(\bar{x}) : -c(x_0), \bigwedge r(x, y), \bigwedge a(x) = y, \bigwedge x\theta n$ satisfying the following restrictions: (i) x_0 is the *root variable* of the query, (ii) if $r(x, y)$ or $a(x) = y$ occurs in the query, x is called the *parent* of y and the variables with the parenthood relation form a tree with x_0 as the root.

1. input: tree query
2. repeat until no further reduction is possible
3. do
4. for each $x.R y$ with y a leaf and y not selected, replace $x.R y$ by $x.R/* y$.
5. for each $x.R y$ with x not selected and y the only child of x then
6. if x is the root variable of the query and $C x$ appears in the query
7. replace $C x, x.R y$ by $C/R y$ (making y the new root).
8. else if $z.S x$ is present in the query, then replace $z.S x, x.R y$ by $z.S/R y$
9. od
10. output: tree query

Fig. 5. Query preprocessing algorithm

which return instances of concept `Person`. Such a rule is R_1 which states that the elements obtained by evaluating XPath expression `Collection/Painter` on the documents in URL `http://www.paintings.com` are instances of concept `Person`. These are bound in variable u_1 . We create the binding $\{(x_1, u_1)\}$ which states that instances of x_1 are the instances of u_1 . In this case the expression `Person` x_1 in Q_1 is replaced by `http://www.paintings.com/Collection/Painter` x_1 . Then, variable x_2 is examined. The algorithm looks for a rule which (i) can be evaluated on instances of x_1 (i.e. instances of u_1) and (ii) whose ontology path is `has_name` (i.e. the binding path of x_2). Such a rule is R_2 since its root variable is u_1 and its ontology path is `has_name`. The expression $x_1.has_name$ x_2 in Q_1 is replaced by $x_1./@name$ x_2 . In a similar manner, expression $x_1.carried_out/produced$ x_3 is replaced by $x_1./Painting$ x_3 using rule R_3 and the binding is extended to $\{(x_1, u_1), (x_3, u_3)\}$. When all the variables have been considered, the obtained query is $Q_1(a)$ illustrated in Table 2. With a simple syntactical transformation¹⁶, it is transformed into the XQuery expression $Q_1(b)$ shown in Table 2.

There are cases where the algorithm briefly described above, does not return a rewriting even if one exists. The reason is the presence of composite concepts and roles in the query and the mapping rules.

To overcome this problem, we need to do a preprocessing of the query. This step is necessary since the algorithm tries to match *all* the query variables with some mapping rule. Here we distinguish between *necessary* and *unnecessary* variables. The latter appear neither in the **select**, nor in the **where** clause of the query. The former must be *always* instantiated by instances from the sources, the latter can be bound to objects in the model of the ontology which do not exist in the sources. Such objects appear in models of the ontology because of

¹⁶ The transformation of query $Q_1(a)$ to the XQuery expression $Q_1(b)$ is done as follows: $Q_1(b)$'s **FOR** clause is obtained by replacing each expression of the form $x_i./q$ x_j in the **from** clause of $Q_1(a)$ by (i) x_j **IN** $x_i./q$ if x_i is a variable and (ii) x_j **IN** `document("u")/q` if x_i is of the form u/q where u is a URL. $Q_1(b)$'s **WHERE** clause contains all conditions in the **where** clause of $Q_1(a)$. Finally, the **RETURN** clause of $Q_1(b)$ contains all variables in the **select** clause of $Q_1(a)$.

role paths and concept paths in the ontology paths of the mapping rules (this is clearly visible from the tableau composition rules). The function of the query preprocessing algorithm is to remove all unnecessary variables by rewriting the initial query into an equivalent one. The query preprocessing algorithm is given in Fig. 5. The special predicate $x.R/*y$ is an abbreviation of the disjunction $x.R'y$ for which there is a mapping rule v_i/q as $v_j \rightarrow R'$ and R is a prefix of R' .

Lemma 3. (i) *The algorithm in Fig. 5 yields an equivalent query.*

(ii) *Let O be an ontology without inverse roles. Let $\exists\bar{y}Q(\bar{x},\bar{y})$ be the output from the algorithm in Fig. 5. Then the following are equivalent:*

- \bar{a} is a certain answer to $\exists\bar{y}Q(\bar{x},\bar{y})$.
- $\bar{a}\bar{b}$ is a certain answer to $Q(\bar{x},\bar{y})$, for some \bar{b} .

5.1 Query Rewriting Algorithm

Our aim is to create a simple, efficient and complete rewriting algorithm. For this, we restrict the queries to tree queries and we do not allow inverse roles, attribute paths in the mapping rules and the queries and global keys in the ontology. The mapping rules must be well presented and obey that all XPath expressions A, B in the mapping rules have the property that if there is a document on which A and B have a non empty intersection, then A and B are syntactically the same. This can be implemented by allowing only the `child` and `attribute` axes in the XPath expressions of the mapping rules. An elaborate motivation (apart from their use in the completeness proof) of these restrictions is provided in the full version of the paper [27].

In this section we present the query rewriting algorithm `oquery2xquery` illustrated in Fig. 6 which is a variation of the query rewriting algorithm in [4]. It accepts as input an OQL tree query (preprocessed by the algorithm shown in Fig. 5) and a mapping M . As output it gives an OQL like “XQuery” expression which has the same **select** and **where** clause as the tree query but whose **from** part consists of the set `XPool` in the algorithm.

The algorithm uses the following three sets: `OntPool` which contains the expressions in the query **from** clause, `XPool` which contains expressions of the form $x_i./XExp\ x_j$ and $XExp\ x_0$ where $XExp$ is an XPath expression, and the x_i 's are query variables and finally `Bindings` that contains pairs of the form (x_i, Var) where x_i is a query variable and Var is a mapping rule variable. It makes use of the (non-deterministic) DATALOG procedure `concept2x` shown in Fig. 7. Given a concept c from the ontology and a mapping M , `concept2x` returns (i) the XPath location path and (ii) the bound variable of an absolute mapping rule R in \hat{M} which returns elements that are (according to the ontology path of R) instances of c .

The algorithm examines first the query root variable x_0 . Let $C_0\ x_0$ be the expression in `OntPool` for x_0 , where C_0 is the binding path of x_0 . Procedure `concept2x()` is called with C_0 and returns the location path $XExp$ and the bound variable Var of an absolute mapping rule which returns instances of C_0 or of subconcepts thereof. $\{XExp\ x_0\}$ is then added in `XPool` and (x_0, Var) in the set

```

begin
  OntPool := the from part of the ontology query;
  let  $C_0$   $x_0 \in$  OntPool
    if concept2x( $C_0$ , XExp, Var)
    then XPool := {XExp  $x_0$ }; Bindings := {( $x_0$ , Var)};
      OntPool := OntPool  $\setminus$  { $C_0$   $x_0$ };
    else fail;
  do OntPool  $\neq \emptyset \longrightarrow$ 
    let  $x_i$ .Exp  $x_j \in$  OntPool
      if ( $x_i, v_i$ )  $\in$  Bindings and  $v_i$ /XExp as Var  $\rightarrow$  Exp  $\in M^*$ ;
      then XPool := XPool  $\cup$  { $x_i$ ./XExp  $x_j$ }; OntPool := OntPool  $\setminus$  { $x_i$ .Exp  $x_j$ };
        Bindings := Bindings  $\cup$  {( $x_j$ , Var)};
      else fail;
    let  $x_i$ .Exp/*  $x_j \in$  OntPool
      if ( $x_i, v_i$ )  $\in$  Bindings and  $v_i$ /XExp as Var  $\rightarrow$  Exp'  $\in M^*$ ;
      where Exp is a prefix of Exp'
      then XPool := XPool  $\cup$  { $x_i$ ./XExp  $x_j$ }; OntPool := OntPool  $\setminus$  { $x_i$ .Exp/*  $x_j$ };
        Bindings := Bindings  $\cup$  {( $x_j$ , Var)};
      else fail;
    let  $a(x_i, x_j) \in$  OntPool
      if ( $x_i, v_i$ )  $\in$  Bindings and  $v_i$ /@XExp  $\rightarrow a \in M^*$ ;
      then XPool := XPool  $\cup$  { $x_i$ ./@XExp  $x_j$ }; OntPool := OntPool  $\setminus$  { $a(x_i, x_j)$ };
      else fail
    od
  end

```

Fig. 6. Query rewriting algorithm oquery2xquery.

```

/* for concepts C and C/Rolepath */
/* if Rolepath is the empty string, "/Rolepath" stands for the empty string */

concept2x(C/Rolepath, U/X, Var):-
  P isa* C,
  U/X as Var  $\rightarrow$  P/Rolepath  $\in \mathbb{M}$ .

concept2x(C/Rolepath, U/X, Var):-
  target(Path) isa* C,
  U/X as Var  $\rightarrow$  Path/Rolepath  $\in \mathbb{M}$ .

```

Fig. 7. Procedure **concept2x**.

Bindings. Expression $C_0 x_0$ is removed from `OntPool`. The algorithm then examines the query variables in preorder by considering a variable after its parent. Let x_j be the variable considered and let $x_i.\text{Exp } x_j$ be an expression in `OntPool`. Let x_i be bound to variable v_i (i.e. $(x_i, v_i) \in \text{Bindings}$). The algorithm will then look for a relative mapping rule R starting with v_i . As far as `Exp` is concerned, we distinguish between two cases:

- If `Exp` is of the form S , R is selected if its ontology path is equal to S ;
- if `Exp` is of the form $S/*$ where S is a role path, then R is selected if S is a prefix of R 's ontology path;

Let the relative rule R be of the form $R : v_i/\text{XExp as Var} \rightarrow \text{Exp}$. Then expression $x_i./\text{XExp } x_j$ is added to `XPool`, $x_i.\text{Exp } x_j$ is removed from `OntPool` and (x_j, Var) is added to `Bindings`. If the algorithm examines an expression of the form $a(x_i, x_j)$ where a is an attribute in the ontology, and x_i is bound to variable v_i , then the algorithm looks for a relative mapping rule of the form $v_i/@\text{XExp} \rightarrow a$. As previously, $x_i./@\text{XExp } x_j$ is added in `XPool` and $a(x_i, x_j)$ is removed from `OntPool`.

We arrived at the central result of the paper:

Theorem 4. *The query rewriting algorithm presented in Fig. 6 is correct and complete. That is, \bar{a} is a certain answer to the query $q(\bar{x})$ posed to the source (u, M) published in O if and only if there exists an “Xquery” $p(\bar{x})$ such that $\text{query2xquery}(q(\bar{x}), p(\bar{x}))$ is true and \bar{a} is an element of the answer set of $p(\bar{x})$ evaluated on source u .*

6 Conclusions

In this paper we have shown completeness and correctness of the query rewriting algorithm proposed in [4] in a restricted setting. That work proposed a novel framework for query mediation over a set of XML resources using an ontology-based mediator. The ontology is an ER schema with subsumption relations between concepts. The sources are mapped to this ontology by means of mapping rules which associate XPath location paths to ontology paths. The main problem for a query rewriting algorithm in this setting is to map node labeled trees (XML documents) to edge labeled graphs with inverse roles and global keys. Leaving out these last two features from the ontology makes that the universal canonical solution has the shape of a tree, a much simpler structure to work with. The mapping language proposed in [4] is very rich: all XPath 1.0 location paths are allowed. This introduces problems for the proposed algorithm which binds query variables to mapping rule variables representing sets of XML elements. Such an algorithm can only be complete if for each XML element there is a unique location path leading to it in the mapping rules. This explains our restriction to location paths using only the `child` and `attribute` XPath axes in the mapping rules. The restrictions we had to pose seem huge but are still reasonable for practical applications (cf [13] for an argument). Our main contribution thus is that mediation in a setting with two very different data formats is feasible in

a simplified but non-trivial case. The main challenge for future work is to cope with inverse roles and global keys in a computationally attractive way as well as to raise the limitations of using only the previously mentioned XPath axes in the mapping rules. Especially efficient algorithms which reason about equalities between data from different XML documents will be needed. Moreover, another axis of research is to introduce XML Schemas in the place of XML DTDs. The other contributions of the paper are a side effect of our effort to prove completeness of the algorithm. They are: (i) the definition of an interpretation function which maps fragments of XML sources into models of the ontology (ii) the notion of certain answer in this context, (iii) a tableau calculus for creating a canonical model (a data warehouse) for the source data and (iv) using this tableau calculus to design other algorithms for query answering.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data On the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, October 1999.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-based Integration of XML Resources. In *Proc. of the First Int'l Conf. on the Semantic Web (ISWC)*, Sardinia, Italy, June 2002.
4. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Querying XML sources using an Ontology-based Mediator. In *Proc. of the Int'l Conf. on Cooperative Information Systems (CoopIS)*, Irvine, California, USA, November 2002.
5. P. Biron and A. Malhotra (eds.). XML Schema Part 2: Datatypes. W3C Recommendation, May 2001. <http://www.w3.org/TR/xmlschema-2/>.
6. P. Buneman, S. B. Davidson, W. Fan, C. S. Hara, and W. C. Tan. Keys for XML. In *Proc. of the Int'l Conf. on the World Wide Web (WWW)*, pages 201–210, 2001.
7. D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. View Based Query Answering and Query Containment over Semi-Structured Data. In *Proc. of DBPL*, pages 40–61, Rome, Italy, September 2001.
8. D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pages 194–204, Philadelphia, Pennsylvania, USA, 1999.
9. P. P. Chen. The Entity Relationship model : Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.
10. J. Clark and S. DeRose (eds.). XML Path Language (XPath) Version 1.0. W3C Recommendation, November 1999. <http://www.w3c.org/TR/xpath>.
11. S. Cluet. Designing OQL: Allowing Objects to be Queried. *Information Systems*, 23(5), 1998.
12. World-Wide Web Consortium. XQuery 1.0: A Query Language for XML. <http://www.w3.org/TR/xquery/>.
13. C. Delobel and M-C. Rousset. A Uniform Approach for Querying Large Tree Structured Data through a Mediated Schema. In *Proc. of Foundations of Models for Information Integration Workshop (FMII-2001)*, Rome, Italy, September 2001.
14. M. Doerr and N. Crofts. Electronic organization on diverse data - the role of an object oriented reference model. In *Proc. of 1998 CIDOC Conf.*, Melbourne, Australia, October 1998.

15. O. M. Duschka and M. R. Genesereth. Answering Recursive queries using Views. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, Tuscon, Arizona, USA, 1997.
16. R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. 2003 Int'l Conf. on Database Theory (ICDT '03)*, pages 207–224, 2003.
17. M. Friedman, A. Levy, and T. Millstein. Navigational Plans for Data Integration. In *Proc. AAAI/IAAI*, 1999.
18. I. Fundulaki. *Integration and Querying of XML resources for Web Communities*. PhD thesis, Conservatoire National des Arts et Métiers, Paris, France, January 2003.
19. I. Fundulaki, B. Amann, C. Beeri, M. Scholl, and A. Vercoustre. STYX : Connecting the XML World to the World of Semantics. In *Proc. of Conf. on Extending Database Technology*, Prague, Czech Republic, March 2002. Demo Presentation.
20. C. Grahne and A. Thomo. New Rewritings and Optimizations for Regular Path Queries. In *Proc. of the Int'l Conf. on Database Theory (ICDT)*, Siena, Italy, January 2003.
21. G. Grahne and A. Mendelzon. Tableau Techniques for Querying Information Sources through Global Schemas. In *Proc. of the 7th Int'l Conf. on Database Theory (ICDT)*, Jerusalem, Israel, January 1999.
22. D. Heimbigner and D. McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3):253–278, July 1985.
23. M. Lenzerini. Data Integration : A Theoretical Perspective. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, Madison, Winsconsin, USA, June 2002.
24. A. Levy. Answering queries using views: a survey. *VLDB Journal*, 2001.
25. A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB), September 3–6, 1996, Mumbai (Bombay), India*, 1996.
26. W. Litwin, I. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.
27. M. Marx and I. Fundulaki. XML Data Mediation. Technical report, University of Amsterdam, 2003.
28. P. Mitra. An Algorithm for answering queries efficiently using views. In *Proc. of the 12th Australasian Conf. on Database Technologies*, Queensland, Australia, 1999.
29. R. Pottinger and A. Levy. A Scalable Algorithm for Answering Queries using Views. *VLDB Journal*, 2001.
30. W3C Technology and Society Domain : Resource Description Framework (RDF). <http://www.w3.org/RDF/>.
31. World Wide Web Consortium. W3C Semantic Web Activity. <http://www.w3.org/2001/semweb-fin/w3c>.
32. P. Wadler. A formal semantics of patterns in XSLT, 1999. URL: cite-seer.nj.nec.com/wadler99formal.html.
33. J. Widom. Research Problems in Data Warehousing. In *Proc. of the 4th Int'l Conf. on Information and Knowledge Management (CIKM)*, pages 25–30, Baltimore, Maryland, November 1995. ACM.
34. G. Wiederhold. Intelligent integration of information. *Journal of Intelligent Information Systems*, 6(2):281–291, May 1996.

A Proofs

Proof (Theorem 2). Construct \mathfrak{B}_T as specified in the theorem. By replacement, $=$ is a congruence, whence the valuation of the concepts, roles and attributes is well defined. \mathfrak{B}_T is a model for O by the isa, source, target and global key rules.

Now let f be the mapping from u to the domains of \mathfrak{B}_T defined by $f(e) = \bar{e}$ for elements and $f(n) = n$ for attribute values.

We now show that f respects the rules of M^* . This is immediate for atomic concepts, roles and attributes by the tableau rules. For the paths, we need the following result for all concept paths C and role paths R : (i) if $C(e_1)$ on the tableau, then $\mathfrak{B}_T \models C(\bar{e}_1)$; (ii) if $e_1 R e_2$ on the tableau, then $\mathfrak{B}_T \models \bar{e}_1 R \bar{e}_2$, which is proved by a simple induction, using the concept path and composition rules.

We note that the other direction only holds in the following sense: $\mathfrak{B}_T \models \bar{e}_1 R_1 \dots R_n \bar{e}_2$ only if there exists a_1, \dots, a_{n-1} such that all of $e_1 R_1 a_1, \dots, a_{i-1} R_i a_i, a_{n-1} R_n e_2$ are on the tableau, and similarly for the concept paths.

Proof (Theorem 3). Suppose \bar{e}_1 is a certain answer to $\phi(\bar{x})$. Then in any model \mathfrak{B} , $\mathfrak{B} \models \phi(\bar{e}_1)$. In particular, $\mathfrak{B}_T \models \phi(\bar{e}_1)$. By assumption, all conjuncts of ϕ are atomic. But then by Theorem 2, all these conjuncts appear on the tableau. But then the tableau procedure yields \bar{e}_1 as an answer.

For the other direction, we need some more work. It is easy to see that all tableau rules are sound: that is, if the condition of the rule is true for a model of O in which (u, M) is interpreted, then the conclusion as well. To continue, we say that the tableau system gives \bar{e}_1 as an answer to $\phi(\bar{x})$ asked to ontology O and source (u, M) if the tableau system with the additional following query rule closes: (here $\phi_1 \dots \phi_n$ are all atomic conjuncts of ϕ)

$$\phi(\bar{x}) \text{ query rule } \frac{\bar{e}_2 \text{ is any sequence of parameters and XML elements}}{\neg\phi_1(\bar{e}_1, \bar{e}_2) \mid \dots \mid \neg\phi_n(\bar{e}_1, \bar{e}_2)},$$

The vertical bars in the tableau rule indicate a branching of the tableau. A branch of a tableau closes if it contains a formula and its negation. A tableau closes if all its branches are closed.

Clearly if the tableau yields \bar{e}_1 as an answer without the new rule, \bar{e}_1 is also an answer with the added rule. Now suppose that \bar{e}_1 is not a certain answer to $\phi(\bar{x}, \bar{y})$. Then for any choice \bar{e}_2 for \bar{y} , there is a model in which one of the conjuncts $\phi_i(\bar{e}_1, \bar{e}_2)$ is false. So, applying the query rule yields a satisfiable situation. But since all rules, preserve satisfiability, there must be an open branch, whence the tableau system does not yield \bar{e}_1 as an answer.

Proof (Lemma 1). Suppose T contains an application of the domain rule

$$\frac{e R e' \quad \text{source}(R) = C}{C(e)}.$$

Then $e R e'$ must have come from an application of (a) the concept path rule, (b) the relative mapping rule or (c) the composition rule. In case (a) we had $\frac{D/R(e')}{D(e), e R e'}$ for some concept D . But by assumption, the path D/R is safe, whence $D \text{ isa source}(R)$ holds. But then an application of the *isa* rule yields the desired conclusion $C(e)$.

In case (b) there are two rules $R_i : u/q \text{ as } v_i \rightarrow D$ and $R_j : v_i/q' \text{ as } v_j \rightarrow R$. As the paths in \hat{M} are safe, it must be that $D \text{ isa } source(R)$. So to derive $source(R)(e)$ apply the absolute mapping rule with rule R_i to derive $D(e)$ and then the isa rule to derive $source(R)(e)$.

In case (c) eRe' came from an application of the composition rule. If the relation R is always the first on a path eR/Pn then this path must have originated from the concept path rule or the relative mapping rule, and we can reason as before. Otherwise, we had the following application of the composition rule to get eRe' :

$$\frac{nP/Re'}{nPe, eRe'}$$

It is easy to show that all paths occurring in the tableau must be a subpath of a path which occurs in the conclusion of some rule in \hat{M} . Whence by assumption then $target(P) \text{ isa } source(R)$. But now we derive $target(P)(e)$ from nPe by the target rule, and from that the desired $source(R)(e)$ with the isa rule.

Proof (Lemma 2). For (i), assume the hypothesis of the lemma. By the completeness theorem $q(\bar{a})$ is an answer iff there exists a tableau proof for $q(\bar{a})$. By Lemma 1, the domain rule is not needed in this proof. Obviously the inverse and the global key rules are not needed in a tableau proof as their antecedent is never true. The two rules for reasoning about equality are not needed because 1) in queries = cannot be used to relate domain elements, and 2) without global keys no statement of the form $e = e'$ is produced in a tableau.

(ii) is immediate by (i), the fact that the composition and concept path rules use *new* parameters and Theorem 2.

Proof (Lemma 3). (i) Immediate by the meaning definition. (ii) By an inspection of the tableau rules yielding the universal canonical solution and Lemma 1.

Proof (Theorem 4). For ease of reference we list once more the restrictions imposed:

- **C1:** Ontologies contain neither inverse roles nor global keys;
- **C2:** All mapping rules are well presented;
- **C3:** The location paths of the mapping rules use only the `child` and `attribute` XPath axes.

Assume a source (u, M) and an ontology O are fixed. Throughout we assume that (u, M) is published in O and that it satisfies conditions **C1–C3**.

First define a partial function b from the elements of u to the set of variables occurring in M as follows:

$$b(e) = v \text{ iff there exists a } X \text{ and } C \text{ such that } u/X \text{ as } v \rightarrow C \in \hat{M} \text{ and } e \in X.$$

To see that b is a function assume that there are X, X', C, C' such that $u/X \text{ as } v \rightarrow C, u/X' \text{ as } v' \rightarrow C' \in \hat{M}$ and $e \in X \cap X'$. Then by the constraints mentioned above $X = X'$, whence by C2 $v = v'$. In the sequel, as usual, if $b(e)$ occurs in a statement it is assumed that b is defined on e .

The next lemma proves the theorem for the four different kinds of conjuncts occurring in queries.

Lemma 4. 1. For any role path P , the following are equivalent

- (i) e_i, e_j is a certain answer to $x_i.P x_j$;
- (ii) for some X , there is a rule $b(e_i)/X \text{ as } b(e_j) \rightarrow P$ and $e_j \in e_i/X$.

2. For any role path $P/*$, the following are equivalent
 - (i) e_i, e_j is a certain answer to $x_i.P/* x_j$;
 - (ii) for some X and role path R' , there is a rule $b(e_i)/X$ as $b(e_j) \rightarrow R'$ and $e_j \in e_i/X$ and P is a prefix of R' .
3. For any concept path C/R (where R can be the empty path), the following are equivalent
 - (i) e is a certain answer to $C/R(x)$;
 - (ii) for some XPath expression u/X the procedure `concept2x(C/R, u/X, b(e))` succeeds and $e \in u/X$.
4. For any attribute a , the following are equivalent:
 - (i) e is a certain answer to $a(x) = n$;
 - (ii) for some XML attribute X there exists a rule $b(e)/@X \rightarrow a \in M^*$ and $n \in e/@X$.

Proof (Lemma 4). In all proofs we use the fact that the certain answers can be computed by the tableau algorithm using only the mapping rule rules, isa, target, composition and concept path rules (Theorem 3 and Lemma 2).

1. The direction from (ii) to (i) is straightforward: since $b(e_i)$ is defined there exists an absolute rule u/q as $b(e_i) \rightarrow C$. This together with the rule $b(e_i)/X$ as $b(e_j) \rightarrow P$ and the fact that $e_j \in e_i/X$ yields e_iPe_j by the relative mapping rule. For the other direction we proceed by induction on the length of P . Let $|P| = 1$. The relative mapping rule is the only rule which produces statements of the form aRb for a, b elements in the source. Whence e_iPe_j must have been produced by an application of this rule. Then there are rules u/Y as $v \rightarrow C$ and $e_i \in u/Y$ and v/X as $w \rightarrow P$ and $e_j \in e_i/X$. But then $v = b(e_i)$ and $w = b(e_j)$, by definition of b . For the induction step, let $|P| = n + 1$. If the tableau proves e_iPe_j then it is obtained by a direct application of the relative mapping rule. In this case we have the desired result, as before. Otherwise the tableau proves e_iP_1n and nP_2e_j for n an element and $P_1/P_2 = P$. By inductive hypothesis then the following two rules are in M^* :

$$\begin{aligned} b(e_i)/X_1 \text{ as } b(n) &\rightarrow P_1 \\ b(n)/X_2 \text{ as } b(e_j) &\rightarrow P_2, \end{aligned}$$

and $n \in e_i/X_1$ and $e_j \in n/X_2$. But then also $b(e_i)/X_1/X_2$ as $b(e_j) \rightarrow P_1/P_2 \in M^*$, and $e_j \in e_i/X_1/X_2$ as desired.

2. By the previous item and the fact that $x.R/* y$ is an abbreviation of the disjunction $x.R' y$ for which there is a mapping rule v_i/q as $v_j \rightarrow R'$ and R is a prefix of R' .
3. It is easy to see that (ii) is equivalent to
 - (iii) for some XPath expression u/X , there exists an absolute rule u/X as $b(e) \rightarrow C'/R$ in \hat{M} and $C' \text{ isa}^* C$ and $e \in u/X$, or there exists a concept path C' and an absolute rule u/X as $b(e) \rightarrow /R$ in \hat{M} and $\text{target}(C') \text{ isa}^* C$ and $e \in u/X$.
 We now prove that (iii) is equivalent to (i). The direction from (iii) to (i) is immediate by an inspection of the tableau rules. The other direction is again by an induction on the length of the role path R . For $|R| = 0$, the statement follows directly from an inspection of the tableau rules and Lemma 2. The inductive case is proved using the first item of this lemma by a similar argument as used in the proof of that item.
4. The statement for attributes is immediate from the tableau rules.

Now we are ready to prove the theorem. By Lemma 3 we may assume without loss of generality that there are no quantified variables in the query. First we prove correctness. Let $p(\bar{x})$ be the “XQuery” and $s(\bar{x})$ the answer. We show that $s(\bar{x})$ is an answer to $q(\bar{x})$ as well. Let B be the set of bindings produced by the algorithm. By construction of the algorithm, for all $(x_i, v_i) \in B$, $b \circ s(x_i) = v_i$. The query consists of four kinds of conjuncts. Here we show for the relational expressions that $s(\bar{x})$ is an answer to those. The other expressions have a similar proof. Let $x_i./Xexp\ x_j$ be a conjunct in $p(\bar{x})$. Thus $s(x_j) \in s(x_i)/Xexp$. This holds if $x_i.Exp\ x_j$ in the ontology query $q(\bar{x})$ and there is a mapping rule $v_i/Xexp\ as\ v_j \rightarrow Exp$ in M^* . Whence $b \circ s(x_i)/Xexp\ as\ b \circ s(x_j) \rightarrow Exp$ in M^* . Moreover $s(x_j) \in s(x_i)/Xexp$ by assumption. Whence by the previous Lemma, $s(x_i)Exps(x_j)$ holds in the universal canonical solution.

Conversely, let $s(\bar{x})$ be a certain answer to the ontology query $q(\bar{x})$. We show that the algorithm produces a rewriting which has $s(\bar{x})$ in its answer set. Let the set of bindings B be $\{(x_i, v_i) \mid b \circ s(x_i) = v_i\}$. By Lemma 4, for every conjunct of $q(\bar{x})$, there exists a corresponding mapping rule. For instance, if $x_i.Exp\ x_j$ is a conjunct, then there is a mapping rule $b \circ s(x_i)/Xexp\ as\ b \circ s(x_j) \rightarrow Exp$ in M^* and $s(x_j) \in s(x_i)/Xexp$. Choose for each conjunct a mapping rule. With this choice the algorithm succeeds with the defined bindings and produces an “XQuery” $p(\bar{x})$ with $s(\bar{x})$ in its answer set.

The ICS-FORTH SWIM: A Powerful Semantic Web Integration Middleware^{*}

V. Christophides¹, G. Karvounarakis¹, I. Koffina¹, G. Kokkinidis¹, A. Magkanaraki¹, D. Plexousakis¹, G. Serfiotis¹, and V. Tannen^{2**}

¹ Institute of Computer Science - FORTH
Vassilika Vouton, PO Box 1385, 71110, Heraklion, Greece
{christop, gregkar, koffina, kokkinid, aimilia, dp,
serfioti}@ics.forth.gr

² Department of Computer and Information Science, University of Pennsylvania
200 South 33rd Street, Philadelphia, PA 19104-6389, USA
val@cis.upenn.edu

Abstract. Semantic Web (SW) technology aims to facilitate the integration of legacy data sources spread worldwide. Despite the plethora of SW languages (e.g., RDF/S, DAML+OIL, OWL) recently proposed for supporting large scale information interoperation, the vast majority of legacy sources still rely on relational databases (RDB) published on the Web or corporate intranets as *virtual* XML. In this paper, we advocate a Datalog framework for mediating high-level queries to relational and/or XML sources using community ontologies expressed in a SW language such as RDF/S. We describe the architecture and the reasoning services of our SW integration middleware, called SWIM, and we present the main design choices and techniques for supporting powerful mappings between different data models, as well as, reformulation and optimization of queries expressed against mediation schemas and views.

1 Introduction

A cornerstone issue in the realization of the Semantic Web (SW) vision is the achievement of semantic interoperability among legacy data sources spread worldwide. In order to capture information semantics in a machine processable way, various ontology-based formalisms have been recently proposed (e.g., RDF/S [21, 5], DAML+OIL [29], OWL [10]). However, the vast majority of existing legacy data is not yet in RDF/S or any other SW language [24, 26]. As a matter of fact, most of the data is physically stored in relational database (RDB) systems and are actually published on the Web or corporate intranets as *virtual* XML.

SW applications, however, require to view data as *virtual* RDF, valid instance of a domain or application specific RDF/S schema, and to be able to manipulate them with high-level query languages, such as RQL [18] or RVL [25]. Therefore,

^{*} This work was partially supported by the EU project SeLeNe (IST-2001-39045).

^{**} Work performed during the visit of the author at ICS-FORTH.

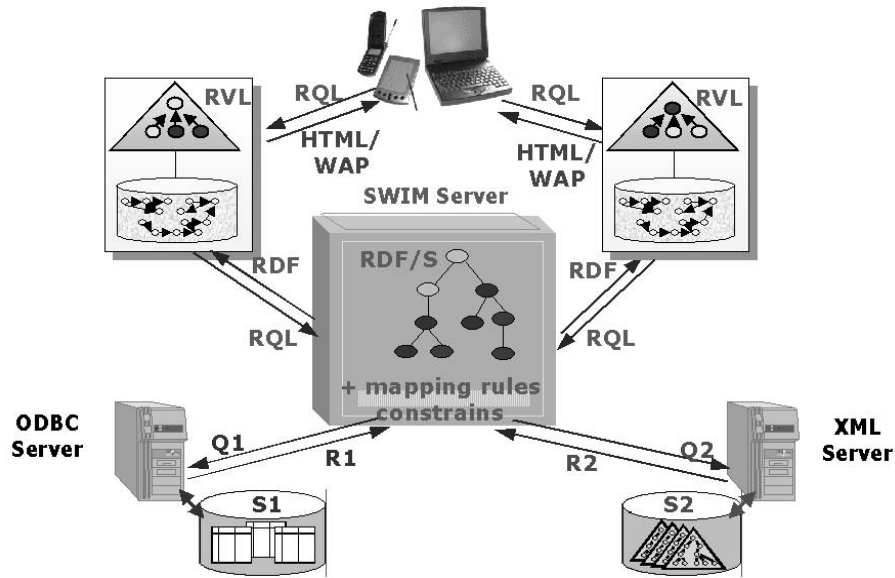


Fig. 1. SWIM Architecture

we need middleware systems that can either republish XML as RDF, or publish RDB data directly as RDF, or - even better - be capable of doing both. Sometimes the practical solution will be to rely just on the virtual XML schema and XML query interface of an existing XML publishing system. At other times, the SW publishing middleware will be built as an alternative to the XML publishing system, taking advantage of direct access to the underlying RDB management system (RDBMS). It is also possible that the SW middleware will have to integrate data in some RDBMS with data in native XML storage.

We need to deal flexibly with all these situations in a uniform framework. A decade of experience with information integration architectures based on mediators [9, 30, 28, 22] suggests that it is highly beneficial to (semi)automatically generate such systems from succinct formal specifications, rather than programming their semantics into low-level code. This greatly enhances the maintainability and reliability of the systems in an environment of often revised and shifting requirements.

This paper presents the fundamental ideas for devising a comprehensive framework that allows user communities to

1. specify XML \rightarrow RDF and RDB \rightarrow RDF mappings;
2. verify that these mappings conform to the semantics of the employed SW ontologies;
3. compose RQL queries with these mappings and produce XML or RDB queries (a.k.a query reformulation);
4. specify further levels of abstraction as RDF \rightarrow RDF views;
5. compose RQL queries with such views;
6. perform query optimizations.

The last requirement is extremely important in such systems. Queries written by humans will rarely have blatant redundancies but queries resulting from automated manipulation/generation are often very "dumb". Minimization techniques, sometimes taking advantage of data semantics provided by ontologies expressed in a SW language, can transform such queries into more efficient ones.

Figure 1 sketches the architecture of a SW integration middleware system that we are building, called SWIM. The lower part of the figure depicts data sources, that could be XML repositories or RDBMS. On top of these sources, we have a domain or application ontology for a particular community, expressed, for instance, in RDF/S. Mapping rules can then be used for the integration, i.e., to translate back and forth from RDF/S to the source data models. As a result, through a SWIM server we can view the underlying sources as virtual RDF repositories and use RQL to query these sources as RDF data or even define personalized views on top using RVL. In this context, the main challenge is to choose an expressive, but still tractable logical framework in which the above functionality (1-6) can be effectively supported by appropriate SW (reasoning) services.

This paper only presents our preliminary design for the SWIM framework. We expect to report on many of the technical challenges and engineering decisions in future publications.

Related Work : Previous projects sharing similar motivations are described in [2, 3], [27] and [16]. Our approach is closest to that of [2, 3], while using a more expressive language for the specification of mappings and a different ontology query language. The papers [23, 6] present formal specifications of mappings from less structured schemas such as XML and relational to more structured schemas of the same level of complexity as RDF. Languages similar to our Datalog with XPath atoms are also used, for example, in [8, 20]. Finally, compared to the Datalog framework for RDF/S-based query mediation of [27], SWIM ensures the compositionality of queries with views and mappings, as well as, supports advanced optimization and verification services.

The remainder of the paper is organized as follows. Section 2 presents a motivating example for cultural data available in RDB or XML sources which can be integrated through an appropriate RDF/S schema. Section 3 presents the internal logical framework of SWIM and its use in the translation and composition of RQL queries. Section 4 touches upon the issue of query optimization by minimization using dependencies while Section 5 addresses the issue of view reformulation. Section 6 examines mapping consistency issues and finally, Section 7 presents our conclusions and an outlook for further research.

2 Motivating Example and SWIM Mapping Rules

Let us assume an XML repository with cultural data, a sample of which appears in the left part of Figure 2. This data could be queried using an XML query

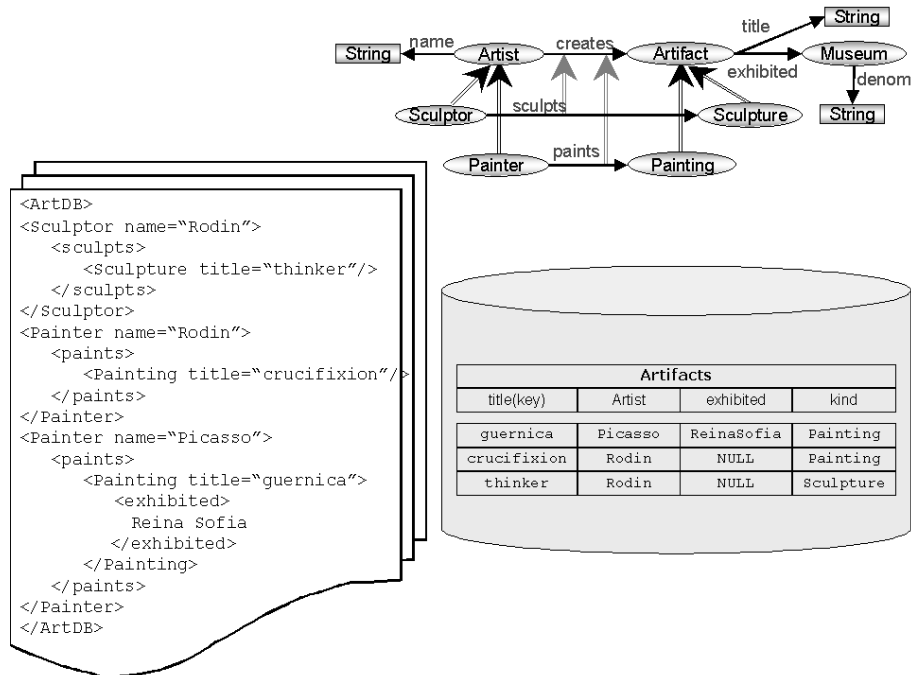


Fig. 2. Example of XML/RDF sources and Mediation RDF/S schema

language, such as XQuery [7]. But now, suppose we add a SWIM server on top of this XML data. For this purpose we design - or import from some community standardization body - an RDF/S cultural schema, as the one depicted in the top part of Figure 2. Now we can formulate queries using an RDF query language by employing only few abstract classes and properties from our mediation RDF/S schema. For example, the following RQL query returns the names of the artists (sculptors or painters) whose work is exhibited in the “Reina Sofia” museum:

```

SELECT Z
FROM {X}creates.exhibited.denom{Y}, {X}name{Z}
WHERE Y = "Reina Sofia"

```

We can observe that the RDF/S layer is completely virtual. The actual data can only be queried using an XML language. Hence, the RQL query we saw needs to be reformulated by the middleware into an XML query. This reformulation should be guided by a formal description of the relationship between the XML and the RDF data, for example a *mapping* from XML to RDF. The question that normally arises is: *how do we express formally such mappings?*

The rich theory developed in the relational case has identified classes of queries and mappings (views) that can be manipulated formally such that various problems like query containment, composing queries with views and rewriting queries with views are algorithmically solvable [1, 15]. These problems can also

be solved in the presence of certain classes of relational constraints [1, 11, 14]. We shall try to rely as much as possible on a well-known and robust formalism: conjunctive queries and views, and embedded implicational dependencies [1]. The results about queries and views are easily extended with union, therefore dealing with the positive existential first-order queries also known as non-recursive Datalog. The dependencies can easily be extended with disjunction [12].

To define XML→RDF mappings we will use an analog to the relational queries just mentioned. We use the same logical shape as that of Datalog rules, but instead of relational atoms, we use XPath atoms in the bodies (this is similar to the XBind queries of [14]). For example, the XPath atom `./Painting(X,Y)` is satisfied by any valuation that maps `X` and `Y` to element nodes in the XML document, such that `Y` has tag `Painting` and is a descendant of `X`. The heads of the rules define RDF instances in the style of the `VIEW` clause employed by the RDF/S view definition language RVL [25]. So, as part of the mapping we can use rules, such as:

```
Painter(X) :- (//Painter) (X)      Sculptor(X) :- (//Sculptor) (X)
```

to define the (direct) extent (i.e., the set of direct instances) of the classes `Painter` and `Sculptor` in the virtual RDF layer. Property extents can be also defined in the same style:

```
paints(X,Y) :- (//Painter) (X), (./Painting) (X,Y)
```

Note that this mapping is not always straightforward, since there usually exist schematic and semantic discrepancies between the source and the middleware schema. For example, class inheritance is not expressed in the XML document. Moreover, properties (let alone property inheritance) `creates`, `paints` and `sculpts` are not used explicitly in the XML document.

We expect SWIM to be able to take the RQL query and the XML→RDF mapping given above and produce an XML query (e.g., an XQuery). We will discuss in Section 3 how this reformulation can be done.

In addition of being available in XML, the cultural data may be available through an RDBMS, for instance in a table as illustrated in the right part of Figure 2. As for XML, there is an RDB→RDF mapping which is also expressed in a mixed language, where instead of XPath atoms we can use standard Datalog atoms:

```
Painter(X) :- Artifacts(_, X, _, "Painting")
paints(X,Y):- Artifacts(Y, X, _, "Painting")
name(X,Y)  :- Artifact(_, X, _, "Painting"), Y=X
name(X,Y)  :- Artifact(_, X, _, "Sculpture"), Y=X
```

As in the case of XML, there may also be discrepancies in the RDB→RDF mapping. For instance, in our example, the classification of an Artist to `Painter` or `Sculptor` is determined by the value of the attribute `kind`, i.e., schema information is “encoded” inside data values.

Again, the SW middleware should be able to automatically reformulate the RQL query, using this mapping, into a relational query, presumably SQL [17].

3 Query Mediation in SWIM

We need an internal logical framework that captures RDF/S semantics, as well as queries, so that we can "virtually populate" given RDF/S schemas. It should also capture - to any needed extent - the XML and RDB semantics. As we showed in the previous section, Datalog-like rules are very convenient for expressing mappings, even across data models, such as XML \rightarrow RDF. Based on the experience of [13, 11] of performing XML query reformulation via translation in a first-order, relational framework, we propose to follow the same approach for RDF, in order to translate both queries and mappings into this framework.

3.1 SWIM Internal Logical Framework

The SWIM internal logic framework employs first-order relations together with some first-order constraints to model RDF/S. It is convenient to use a signature with three sorts: Resource, Property, Class³. The relations used have the following meaning:

- C_EXT(c, x) iff the resource x is in the *proper extent* (i.e., it is a direct instance) of class c . In RDF class extents can overlap, due to multiple classification of resources.
- C_SUB(c, d) iff c is a (not necessarily direct) subclass of d .
- PROP(c, p, d) iff class c is the domain and class d is the range of property p .
- P_EXT(x, p, y) iff (x, y) is in the *proper extent* (i.e., it is a direct instance) of property p . In our model instances of properties are represented as ordered pairs of the resources they connect.
- P_SUB(p, q) iff p is a (not necessarily direct) subproperty of q .

The relations must satisfy some *built-in* RDF/S constraints which are considered by RQL. In particular, the domain and range of a property must be unique, while the subclass and subproperty relations must be reflexive, transitive and satisfy the following *subproperty/subclass compatibility* constraint:

$$\forall a, p, b, c, q, d \text{ P_SUB}(q, p) \wedge \text{PROP}(a, p, b) \wedge \text{PROP}(c, q, d) \\ \longrightarrow \text{C_SUB}(c, a) \wedge \text{C_SUB}(d, b)$$

This means that if q is a subproperty of p , the domain and range of q are subclasses of the domain and range of p , respectively.

Finally, we have the *property-class extent compatibility* constraint, i.e., any instance of a property p connects a pair of instances of some subclasses of the domain and range of p , respectively:

$$\forall a, p, b, x, y \text{ PROP}(a, p, b) \wedge \text{P_EXT}(x, p, y) \\ \longrightarrow \exists c, d \text{ C_SUB}(c, a) \wedge \text{C_SUB}(d, b) \wedge \text{C_EXT}(c, x) \wedge \text{C_EXT}(d, y)$$

Let Δ_{RDF} be the set of dependencies (constraints) used to axiomatize the internal RDF/S model.

³ For simplicity reasons, we ignore metaclasses and metaproperties in this discussion but they can be handled easily in the same way.

Theorem 1. *It is decidable whether $\Delta_{\text{RDF}} \models d$ and whether $\Delta_{\text{RDF}} \models Q_1 \sqsubseteq Q_2$, where d is an embedded implicational dependency, Q_1, Q_2 are conjunctive queries and \sqsubseteq is query containment.*

Translation of RDF/S schemas: It is straightforward to translate the information of an RDF/S schema to the SWIM internal framework as a set of relational facts (in Datalog parlance—an extensional database), involving the relations C_SUB, PROP, P_SUB as well as the names of classes and properties in the schema as constants. Some of the facts obtained from the schema in Figure 2:

C_SUB(Painting, Artifact) PROP(Artist, name, String) P_SUB(sculpts, creates)

Note that this set of facts will include all C_SUB and P_SUB reflexivity instances and will be “closed” under transitivity and under subproperty/subclass compatibility.

3.2 Translation of RQL Queries

RQL is a powerful language for querying smoothly both RDF/S schemas and their instances. An RQL *conjunctive* query has the form $ans(\bar{X}) : - C_1, \dots, C_n$ where C_i 's are either RQL class or property patterns (as they appear in the RQL FROM clause) or equalities involving variables and/or constants and \bar{X} is a tuple of variables or constants (range restrictions [1] are also required). Many RQL queries are in fact conjunctive queries, e.g., the query given in Section 2 can be written:

```
ans(Z):- {X}creates{V}, {V}exhibited{W}, {W}denom{Y},
         {X}name{Z}, Y="Reina Sofia"
```

Conjunctive RQL queries can then be translated into relational conjunctive queries in the SWIM internal logical framework. Indeed, according to the declarative semantics in [18], RQL patterns have the same meaning as conjunctions of relational atoms. For example:

<i>RQL Pattern</i>	<i>Internal SWIM Translation</i>
$\{X; \$C\}@P\{Y; \$D\}$	PROP(a, p, b), P_SUB(q, p), P_EXT(x, q, y), C_SUB(c, a), C_SUB(d, b), C_EXT(c, x), C_EXT(d, y)
$\{X\}@P\{Y\}$	P_SUB(q, p), P_EXT(x, q, y)

In the above RQL patterns, X, Y are resource variables, $\$C, \D are class variables (and can be replaced with constant class names), and $@P$ is a property variable (that also can be replaced by a constant property name). Using these

patterns, the RQL conjunctive query above translates internally to the following Datalog rule:

$$\begin{aligned} ans(z) : - & P_SUB(q_1, creates), P_EXT(x, q_1, v), \\ & P_SUB(q_2, exhibited), P_EXT(v, q_2, w), \\ & P_SUB(q_3, denom), P_EXT(w, q_3, "Reina Sofia"), \\ & P_SUB(q_4, name), P_EXT(x, q_4, z) \end{aligned}$$

3.3 Composing Queries with Mappings

Starting with the internal translation of the query, we perform an interesting *partial evaluation* using the RDF schema information, i.e., we evaluate first the schema-part of the query, namely the P_SUB expressions. This is related to partial evaluation of Datalog programs [4]. Because some atoms (e.g., P_SUB(q_1 , creates)) match more than one fact in the schema, what was a single conjunctive query now becomes a (non-recursive) Datalog program. Here is one of the rules in our example (the other two feature `sculpts` and `creates`):

$$\begin{aligned} ans(z) : - & P_EXT(x, paints, v), P_EXT(v, exhibited, w), \\ & P_EXT(w, denom, "Reina Sofia"), P_EXT(x, name, z) \end{aligned}$$

The next step is to translate into the SWIM internal framework the heads of the rules that define the mappings. For example, a rule defining the extent of the class `Painter` has the head `Painter(X)`. We translate this into `C_EXT(Painter, x)`. In the same style we can translate the rule defining the extent of the property `paints(X, Y)` into `P_EXT(x, paints, y)`. Thus, the mapping becomes a (non-recursive) Datalog-like program with XPath atoms for the XML→RDF case and a plain non-recursive Datalog program for the RDB→RDF case. The composition of the query and the mapping is now simply the composition of two Datalog programs.

To finish the reformulation, we must still eliminate the intermediate predicates `C_EXT`, `P_EXT` because they are not part of the data sources. This is done with standard matching/substitution but it may increase (square, in fact) the number of rules. In the examples we have looked at so far, however, the resulting union of conjunctive queries can be minimized significantly because many of the rules are unsatisfiable and hence can be discarded (see next section).

4 RQL Query Reformulation and Optimization

Continuing the example from Section 3.3, we compose the query with the mapping for the RDB→RDF case. After eliminating the intermediate predicates `C_EXT` and `P_EXT` we obtain a Datalog program with eight rules. Six of these rules, however, are unsatisfiable because their bodies equate distinct constants. Moreover, standard conjunctive query minimization [1] applies to the remaining

two rules. The final reformulated query, after optimizations, for the RDB→RDF case is the following union of conjunctive query (a non-recursive Datalog program with two rules):

```
ans(z) :- Artifacts(x, z, "Reina Sofia", "Painting")
ans(z) :- Artifacts(x, z, "Reina Sofia", "Sculpture")
```

Similar transformations are performed in the case of the XML→RDF mapping. We also encounter six unsatisfiable rules: for example in a rule containing both (`//Sculpture`) (y) and (`./Painting`) (x, y) there is no valuation for y since an XML element cannot have two different tags (i.e., `Sculpture` and `Painting`). The reformulated query for the XML→RDF case is given below:

```
ans(z) :- (//Painter)(x), (./@name)(x, z),
           (//Painter)(x), (./Painting)(x, y),
           (//Painting)(y), (./@exhibited)(y, "Reina Sofia")

ans(z) :- (//Sculptor)(x), (./@name)(x, z),
           (//Sculptor)(x), (./Sculpture)(x, y),
           (//Sculpture)(y), (./@exhibited)(y, "Reina Sofia")
```

However, the problem of deciding satisfiability of rules with XPath atoms seems more complicated to cope with. We expect that the techniques developed in [14] will help with this problem and more generally with the minimization of such queries.

The optimizations we have seen so far do not take into account the specifics of the RDF/S semantics considered by RQL. However, once we have encoded this semantics into the relational dependencies Δ_{RDF} (see Section 3.1) we can use Δ_{RDF} in minimizing queries. For example, by translating into the internal model and by using minimization under dependencies done with the Chase&Backchase algorithm [11] it is possible to show that the conjunctive RQL queries of the form

```
ans(X,@P,Y) :- {X;$C}@P{Y;$D}, rest(X,@P,Y)
```

minimize to (the internal translation of):

```
ans(X,@P,Y) :- {X}@P{Y}, rest(X,@P,Y)
```

thus eliminating several redundant scans over the class variables $\$C$ and $\$D$ (`rest(X,@P,Y)` stands for a boolean predicate whose variables are X , $@P$ and Y only). It should be stressed that if we just translate these queries into SWIM internal conjunctive queries, the results are not equivalent in the absence of Δ_{RDF} . The examples we saw in this section serve as a guide for design decisions regarding what kind of optimization facilities need to be incorporated into SWIM.

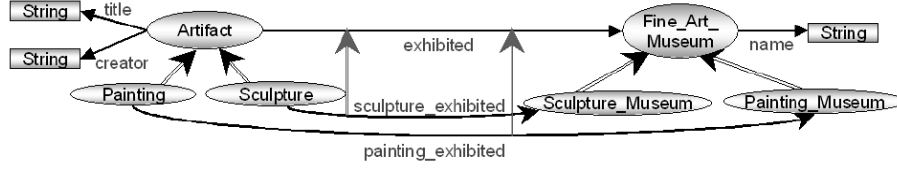


Fig. 3. A Virtual RDF/S Schema on cultural data

5 Composing RQL Queries with RVL views

In order to favor personalization, virtual RDF/S schemas can be also specified on top of the mediator schema, as for instance the RVL schema shown in Figure 3. If we restrict our attention to "conjunctive" RVL definitions, virtual classes' and properties' extents can also be written as rules of the following form:

```

painting_exhibited(X,Y) :- {X;Painting}exhibited{Y}
name(Y,W) :- {X;Painting}exhibited{Y}, {Y}denom{W}
name(Y,W) :- {X;Sculpture}exhibited{Y}, {Y}denom{W}

```

Then, these rules can be employed by SWIM in order to translate RQL queries expressed in terms of a virtual RDF/S schema into the mediator RDF/S schema and back to the source schemas as well. Consider for example the following query, which retrieves the exhibits of the *Reina Sofia* museum:

$$ans(x) :- \{X\}painting_exhibited\{Y\}, \{Y\}name\{Z\}, Z = "Reina Sofia"$$

which translates to:

$$ans(x) :- P_SUB_V(q', painting_exhibited), P_EXT_V(x, q', y), \\ P_SUB_V(q'', name), P_EXT_V(y, q'', "Reina Sofia")$$

The SWIM internal framework is equipped in this case with similar relations as those presented in Section 3.1 in order to capture virtual classes and properties, as well as their virtual subsumption relationships as defined in RVL, namely C_EXT_V, P_EXT_V, C_SUB_V, P_SUB_V, respectively. Since P_SUB_V(q', painting_exhibited) matches only the reflexivity instance P_SUB_V(painting_exhibited, painting_exhibited) (similarly for P_SUB_V(q'', name)), we obtain the following queries (called in order Q₁ and Q₂) against the mediator schema:

$$ans(x) :- PROP(a, exhibited, b), P_SUB(q, exhibited), P_EXT(x, q, y), \\ C_SUB(Paintings, a), C_EXT(Paintings, x), \\ P_SUB(q_2, denom), P_EXT(y, q_2, "Reina Sofia")$$

$$ans(x) :- PROP(a, exhibited, b), P_SUB(q, exhibited), P_EXT(x, q, y), \\ C_SUB(Paintings, a), C_EXT(Paintings, x), \\ C_SUB(Sculpture, a), C_EXT(Sculpture, x), \\ P_SUB(q_2, denom), P_EXT(y, q_2, "Reina Sofia")$$

As we can observe, Q_1 is a subquery of Q_2 . Hence, the result of Q_2 is subsumed by the result of Q_1 ($Q_2 \sqsubseteq Q_1$) and the original query against the view is reformulated to Q_1 .

6 Consistency of Mappings

When a mapping $RDB \rightarrow RDF$, $XML \rightarrow RDF$, or even $RDF \rightarrow RDF$ (that is an RVL view) is specified by a user, its output (if materialized) may not be a valid RDF instance, that is, it may not satisfy the built-in constraints Δ_{RDF} of Section 3.1. For example, suppose, in the context of our example from Section 2, that we define the extent of the property `name` in an $RDB \rightarrow RDF$ mapping by

```
name(X,V) :- Artifacts(Y,X,Z,U), V=X
```

(instead of the correct rules given in Section 2). With this, the mapped data will not satisfy the *property-class extent compatibility* constraint (unless the relation "Artifacts" contains only "Painting" or "Sculpture" as kinds).

Can such an error be detected automatically? That is, given an $RDB \rightarrow RDF$, $XML \rightarrow RDF$, or even $RDF \rightarrow RDF$ mapping, is it decidable if its virtual output satisfies Δ_{RDF} ? Given the translations we gave earlier, in at least two cases ($RDB \rightarrow RDF$ and $RDF \rightarrow RDF$) this question comes down to testing if a relational dependency holds in a relational conjunctive (or union of conjunctive) view. In [19] this was shown decidable for full dependencies (see [1]). Our dependencies in Δ_{RDF} are a little more general but we were able to show that the result extends and we believe that we can extend it also for $XML \rightarrow RDF$ views given suitable XPath restrictions.

7 Conclusions and Future Work

In this paper we presented the principles underlying the design of SWIM (Semantic Web Integration Middleware) and described the components that achieve semantic integration by mapping XML and relational data to RDF. The unifying framework proposed relies on the use of Datalog-like rules for expressing the mappings and reformulating RQL queries. Furthermore, this framework permits the optimization of RQL queries as well as their composition with the specified mappings in order to produce XML or relational database queries. Last, but not least, we showed how these ideas carry over to querying across mediated or personalized RDF schemas by expressing a class of RVL view definitions into SWIM's internal model.

Several issues require further investigation. Specifically, we have dealt so far with the case of conjunctive RQL queries and conjunctive RVL view definitions. In both these cases we obtain a translation into non-recursive Datalog programs to which we can apply well-known optimization techniques and for which the problem of determining the consistency of the mappings is decidable. We intend

to study the conditions under which similar results can be obtained for a broader class of RQL queries and RVL view definitions. Another issue is the exploitation of knowledge about the source schemas and data in order to perform further optimizations during the reformulation process. SWIM's internal model can also accommodate constraints such as the ones expressible in OWL [10]. It will be interesting to study the optimization potential that stems from the use of such constraints (e.g., uniqueness or disjointness constraints) in query reformulation / minimization.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-Based Integration of XML Web Resources. In *Proc. of the International Semantic Web Conference (ISWC)*, Sardinia, Italy, June 2002.
3. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Querying XML Sources Using an Ontology-Based Mediator. In *Proceedings of International Conf. on Cooperative Information Systems (CoopIS)*, pages 429–448, Irvine, California, USA, November 2002.
4. K. Benkerimi and J. Lloyd. A Partial Evaluation Procedure for Logic Programs. In *Proceedings of the North American Conference on Logic Programs*, Austin, TX, USA, 1990. MIT Press.
5. D. Brickley and R.V. Guha. Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation, March 27, 2000.
6. A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of the 20th Int. Conf. on Conceptual Modeling (ER 2001)*, pages 270–284, Yokohama, Japan, November 2001.
7. D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. XQuery: An XML Query Language. W3C Working Draft, May 2003. See <http://www.w3.org/TR/xquery/>.
8. S. Cluet, P. Veltri, and D. Vodislav. Views in a Large Scale XML Repository. In *Proc. of International Conf. on Very Large Databases (VLDB)*, pages 271–280, Roma, Italy, September 2001.
9. Sophie Cluet, Claude Delobel, Jérôme Siméon, and Katarzyna Smaga. Your Mediators Need Data Conversion! In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 177–188, Seattle, WA, USA, June 1998.
10. M. Dean, D. Connolly, F. Van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L.A. Stein. OWL Web Ontology Language Reference Version 1.0, W3C Working Draft. Technical report, W3C, December 12, 2002.
11. A. Deutsch, L. Popa, and V. Tannen. Physical Data Independence, Constraints, and Optimization with Universal Plans. In *Proc. of International Conf. on Very Large Databases (VLDB)*, pages 459–470, Edinburgh, Scotland, UK, September 1999.
12. A. Deutsch and V. Tannen. Optimization Properties for Classes of Conjunctive Regular Path Queries. In *Proc. of International Workshop on Database Programming Languages*, pages 21–39, Frascati, Italy, 2001.

13. A. Deutsch and V. Tannen. MARS: A System for Publishing XML from Mixed and Redundant Storage. In *Proc. of International Conf. on Very Large Databases (VLDB)*, Berlin, Germany, September 2003. To appear.
14. A. Deutsch and V. Tannen. Reformulation of XML Queries and Constraints. In *Proc. of International Conf. on Database Theory (ICDT)*, pages 255–241, Siena, Italy, January 2003.
15. A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
16. A.Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: data management infrastructure for semantic web applications. In *Proc. of International World Wide Web Conf.*, pages 556–567, Budapest, Hungary, 2003.
17. ISO/IEC 9075: Information technology – Database Languages – SQL, 1999.
18. G. Karvounarakis, A. Magkanaraki, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, and K. Tolle. Querying the Semantic Web with RQL. *Computer Networks*, 42(5):617–640, August 2003.
19. A.C. Klug and R. Price. Determining View Dependencies Using Tableaux. *ACM Transactions on Database Systems*, 7(3):361–380, 1982.
20. L.V.S. Lakshmanan and F. Sadri. XML Interoperability. In *Proc. of the International Workshop on the Web and Databases (WebDB)*, San Diego, California, USA, June 2003.
21. O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, February 1999. Available at <http://www.w3.org/TR/REC-rdf-syntax>.
22. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proc. of International Conf. on Very Large Databases (VLDB)*, pages 251–262, Bombay, India, September 1996.
23. B. Ludäscher, A. Gupta, and M. Martone. Model-Based Mediation with Domain Maps. In *Proc. of IEEE International Conf. on Data Engineering (ICDE)*, Heidelberg, Germany, April 2001.
24. A. Magkanaraki, S. Alexaki, V. Christophides, and D. Plexousakis. Benchmarking RDF Schemas for the Semantic Web. In *Proc. of the International Semantic Web Conference (ISWC)*, Sardinia, Italy, June 2002.
25. A. Magkanaraki, V. Tannen, V. Christophides, and D. Plexousakis. Viewing the Semantic Web Through RVL Lenses. In *Proc. of the International Semantic Web Conference (ISWC)*, 2003. To appear.
26. L. Mignet, D. Barbosa, and P. Veltri. The XML web: a first study. In *Proc. of International World Wide Web Conf.*, pages 500–510, Budapest, Hungary, May 2003.
27. EDUTELLA: A P2P Networking Infrastructure Based on RDF. W. Nejdl and B. Wolf and C. Qu and S. Decker and M. Sintek and A. Naeve and M. Nilsson and M. Palmér and T. Risch. In *Proc. of International World Wide Web Conf.*, Honolulu, Hawaii, USA, May 2002.
28. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. In *Proc. of IEEE International Conf. on Data Engineering (ICDE)*, pages 251–260, Taipei, Taiwan, March 1995.
29. F. van Harmelen, P. Patel-Schneider, and I. Horrocks. Reference description of the DAML+OIL ontology markup language. <http://www.daml.org/2001/03/-reference.html>, March 2001.
30. G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

Semantic Representation of Contract Knowledge using Multi Tier Ontology

Vandana Kabilan

Paul Johannesson

Department of Computer and System Sciences,
Stockholm University and Royal Institute of Technology
FORUM 100 , SE 164 40,Kista, Sweden
{Vandana, Pajo}@dsv.su.se

Business contract knowledge exists dispersed in different domains. For successful business process functioning, a precise, clear understanding and interpretation of contractual terms and conditions is required. A semantic interpretation of *contract obligations* and their required *performances to fulfill* the obligations, is aimed to bridge the existing gap between business process management and contract management. The increasing impact of e-commerce also necessitates the requirement for centralized, reusable knowledge bases. This paper presents conceptual models and an ontological representation methodology for capturing semantic interpretations of business contracts in a *Multi Tier Contract Ontology*.

1.Introduction

Humanity started trading using simple barter systems, goods in exchange for goods. Business trade relationships are now complex processes of building trust, understanding and mutual agreement. At the center of these processes are the business legal contracts. It is essential that all parties concerned have a clear understanding of the contents and implications of the agreed contractual terms and conditions. With the adoption of legal regulations facilitating e-commerce, electronic contracting and e-commerce should set new trends in the near future. Notable are e-commerce standardization efforts like that of ebXML[1], which enables partners across the globe to participate in electronic trade relationships using the available Internet technology. This means that business organizations can enter into contractual relationships with partners, hereto unknown and unseen. The need for human understanding of the established contract is thus obvious. Added to this, electronic contracting agents for drafting, negotiation and enforcement are the focus of several contemporary research efforts. Considering the current global perspective of contracting and business, the need for a meaningful semantic web for modeling, representing and exchanging knowledge is well established.

The semantic web[2] has a visionary goal and objective of making the World Wide Web into one gigantic knowledge resource. The semantic web is visualized as a grad-

ual stepwise tower of semantic languages as put forward by Tim Berners-Lee[3] (See figure 1). Our contribution to the semantic web movement is currently in the knowledge resource development in the form of ontology. Ontology vocabularies are a viable candidate for such global knowledge pools. The importance of ontology engineering for the success of the net commerce has been discussed by Howard Smith [4].

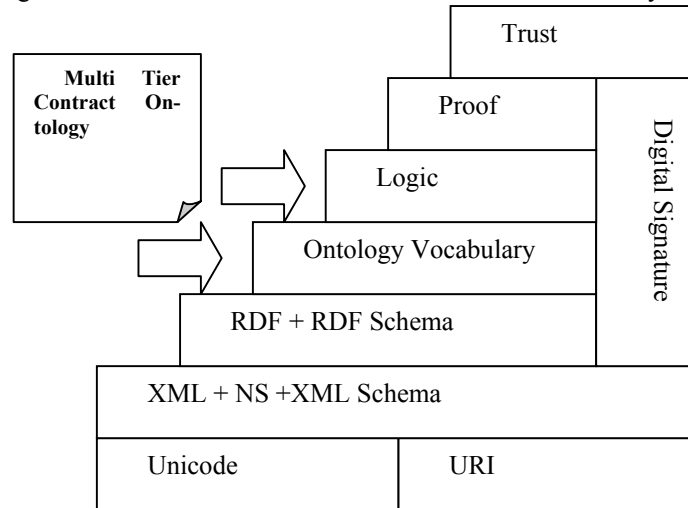


Figure 1: Semantic Web Architecture

Contract Management has existed for some time as well as Business Process Management. However, most available solutions like ERP, CRM or other database applications for enterprise management or contract management, have not managed to integrate the two disciplines seamlessly. A business contract is like a master plan for expected business behavior of the parties involved. Generally, it covers most contingencies and probable scenarios for planned execution of the commitments the parties make to each other. Thus, non-compliance to the contract terms could lead to legal, economic and business repercussions. A business contract should govern and establish the actual business process workflow of the parties. Efforts have been made to build discretionary enforcement agents using subjective logic [5], or deontic logic [6]. Others have treated the contract as documents or processes [7,8]. Our focus has been on the knowledge base representation and modeling methodology for capturing the semantics of a contract. This paper adopts the framework as proposed by the same authors in [9].

Business contracts are one specific application domain in the realm of enterprise application management. This paper presents a layered ontology structure for representing contractual domain perspectives. The conceptual meanings and interpretations of the contractual obligations inherent in a business contract are analyzed and represented in the multi tier contract ontology. The choice of knowledge representation methodology depends to a large extent on the purpose as well as the intended audience for the knowledge base. As mentioned earlier, business contracts are testaments to the commitments made by business entities to each other in the context of a busi-

ness trade relationships. Business contract management depends on several factors that cannot all be automated. Human intervention cannot be sidelined. Thus, the first targets for knowledge transfer are humans and thereafter, machines and software agents. Hence, we present conceptual models using UML [10] as the first step. Next, we propose a transformation of the same to machine understandable format using RDFS [11] /DAML [12].

The main contributions of this paper are the conceptual models of contract knowledge using UML in a multi tier ontology framework. Thereafter, the paper presents validations of the proposed methodology in the form of proof of concept implementations of the conceptual models in RDFS and DAML ontology representation languages.

The rest of the paper is structured as follows. In Section 2 we present a short summary of related research work in the domain of contracting and ontology. In section 3, we discuss our choice of UML as an ontology-modeling tool, followed by conceptual models for our proposed Multi Tier Contract Ontology (section 4). The paper outlines the overall structure of the Multi tier contract ontology and thereafter focuses on the detailed analysis of a specific contract type. We present a proof of concept transformations to DAML and RDFS using Protégé 2000[13] as a tool in section 5. The proof of concept illustrates that the transformation from conceptual model to machine understandable format is possible. In this process, we came across some practical hitches while transforming from UML to DAML or UML to RDFS. We present some of our observations in section 5.3. Thereafter we propose some applications for the Multi Tier Contract Ontology in section 6, followed by concluding remarks in Section 7.

2.Related Research

Contracting, especially electronic contracting has been the topic of interest for several groups of researchers. Though most have the same ultimate objective, each has adopted a different methodology to achieve the same. We see that a contract has been viewed in different perspectives in general:

- Document Centric – a contract is considered as a physical document and its contents are analyzed and modeled as entities. This has little or no semantics involved with it [14]
- Process Centric – a contract is viewed as a statement of business processes or workflows. In this aspect, though semantic interpretation has been tried, most efforts tend to interpret the contract conditions as rules, policies requiring stringent enforcements.
- Legal Centric – a contract is a legal instrument. Efforts are on to establish legal dictionaries [15].

Electronic contracting was pioneered by the efforts of Ronald Lee [16,17] who has amongst other things proposed the use of Petri Nets to model contract procedures like the Documentary Credits. Grosf in [18] has proposed Courteous Logic Programs as a declarative approach to model the business rules and policies as expressed in contracts. Grosf has further presented a XML based rule representation language RuleML and has also used it with ontologies to produce SweetDeal [19], an approach to aid automated creation, evaluation, negotiation and execution of contracts. He has

viewed contracts as specification for processes. There exist possibilities of integrating other contract ontologies like our proposed multi tier contract ontology to the system as proposed by Grosz.

Kimbrough, Moore [20] and others have worked on a Formal Language for Business Communication (FLBC), used to model and structure the communication for negotiation of agreements. Daskalopulu [21,22] has used subjective logic to monitor electronic contract performance.

Heuvel and Weigand [23] have presented integrated enterprise architecture to integrate contracts with business workflow and business objects. They have visualized contracts as the binding glue to cross-organizational business workflows. Contracts are scenarios denoting sequences of transactions.

Yao-Hua Tan has used deontic logic to model directed obligations and permissions in [24]. He has also used event semantics as proposed in FLBC to model the semantics of a contract. He then uses Prolog to implement the modeled contract conditions. His work gives this paper its foundation for the classification of obligation states.

Levine and Pomeroy [25] have proposed a methodology called ABC (Approach Based on Contract) to construct business models using contracts as a starting point.

Goodchild [26] has analyzed the fundamental concepts for a business contract and has modeled the contract using UML and represented them in XML. However, he has viewed the contract as a document and has placed emphasis on the physical characterization of a contract contents.

From the above discussion we see that though semantic interpretation and automated contracting is not novel but little has been done to model the semantics of a contract in the form of a knowledge base. A semantic knowledge pool would enhance and complement the various methodologies proposed for automated contracting. At the same time, contracts depend on human interaction. Thus human-to-human communication is the first line of approach for our proposed methodology. The Multi Tier Contract Ontology is the representation of contract knowledge for such a purpose.

We believe that one of the fundamental keys to the success of the semantic web is the reuse and integration of other related approaches and methodologies. We have been guided by the works of Noy and McGuinness [27] and Gruber [28] for design methodologies for the proposed Multi Tier Contract Ontology. McGuinness [29] has supported the role of ontology engineering in the domain of business process engineering. Howard Smith [4] advocates the importance of ontology for agents to rely on and to communicate with other agents.

3.UML as Ontology Modeling Language

Contract knowledge existing in different domains has to be modeled and represented using standard, comprehensible notations. Knowledge Base resources form an essential component of any information system, be it artificial intelligence agents, software tools or enterprise application software. Such a knowledge base should use a knowledge representation language that is independent of application domain. It should be clear, easy to understand and portable. As stated earlier, the first objective in this research is to facilitate human-to-human knowledge transfer. Later, we propose to progress towards deductive logic and automated inference systems for contract

term interpretation and decision support. We chose the Object Management Group's Unified Modeling Language (UML) [10] as our conceptual model representation language.

Advantages of UML as an ontology modeling language has been proposed by Cranefield [30], Hart, Baklawski et al in [31] as:

- It has a growing user audience in the software domain for object modeling languages and other information system design. In our case, those attempting to integrate business contracts with existing business management applications, are more likely to be familiar with UML than other knowledge representation languages like KIF.
- The graphical notation for UML is easy to comprehend and use and is suitable for human-to-human knowledge transfer.
- UML can be extended to suit the need of ontology definitions.
- Object Constraint Language allows expression of rules and constraints.

Moreover, UML conceptual models can be translated into other ontology languages like RDFS or DAML or even in to object oriented database systems. Cranefield in [32] has proposed mappings to transform UML ontology models in to RDF and to generate Java classes from UML using XSLT.

Ongoing research and open source development in the field of semantic web and ontologies have contributed to a rapidly increasing pool of reusable knowledge resources, tools and guidelines. We have used Protégé 2000[13] as our ontology editor tool. Open source plugins are available for automated generation of RDFS ontology from UML conceptual models, DAML storage etc. Others like DUET (DAML UML Enhanced Tool)[33] of the CODIP (Components for Ontology Driven Information Push)[34] project provide DAML support to UML tools like Rational, Argo UML. This paper adopts and uses such available technology and research methodologies in the aim of contributing productively to the vision of semantic web.

Baklawski [31] has presented some mappings for translating in between DAML and UML concepts and from UML to DAML, as illustrated in the figure (2) below, which have been adopted in this paper.

DAML Concept	Similar UML Concepts
Ontology	Package
Class	Class
As Sets (disjoint, union)	Difficult to represent
Hierarchy	Class Generalization Relations
Property	Aspects of Attributes, Associations and Classes
Hierarchy	None for Attributes, limited Generalization for Associations, Class Generalization Relations
Restriction	Constrain Association ends , including multiplicity and roles. Implicitly as class containing the attribute
Data Types	Data Types
Instances and Values	Object Instances and Attribute Values

Figure 2. High-Level Mapping of UML and DAML Concepts

4. Multi Tier Contract Ontology

4.1. Background

A business contract goes through different phases in its life cycle from the pre – conception, drafting phase, through negotiation and signing till the execution. Angelov has identified various phases and sub phases of the contracting process in [35] and as depicted in figure (3) below.



Figure 3: Contract Cycle

Business management, process, requirements and strategic knowledge contribute to the pre conception contract phase. Legal counsel, recommended practices, contract model law play important roles in the contract-drafting phase. Contracts are then proposed to suitable partners by a *proposer*, counter offers and acceptances are then offered. This process of understanding and coming to a mutually satisfying agreement is the contract negotiation phase, followed by the actual signing and validating of the contract document. Finally, the contract is to be carried out and fulfilled. In the contract execution phase, the agreed conditions and promises are acted upon. Contract execution depends on the actual business process workflow. It needs to be monitored and commitments fulfilled within the contract execution phase. The contract execution is terminated once the contract period is over or it leads to a renewed or fresh contract being negotiated. The proposed Multi Tier Contract Ontology would be a central knowledge base for all the above-mentioned phases. But current work is focused on the contract execution phase, especially on deducing the business workflow from contract terms as well as monitoring and fulfillment of commitments.

Contract knowledge has been modeled based on domain input from the legal framework. Business process knowledge has been based on other ontologies and standards like REA ontology and ebXML. Finally, business workflow patterns as proposed by Van der Aalst [36], Sivaraman [37] and others, have been adapted to model contract workflow patterns.

4.2. Multi Tier Contract Ontology framework

Within the realm of business contracts alone, there exist many different types of contracts [38] having different scopes and applicability. It is impractical to represent all the different types by single contract ontology. Following the ontology design principle as proposed by Guarino [39], a structured and layered framework for contract ontology was envisioned. A layered structure provides the scope for defining an individual ontology for specific types yet coherently integrating under one unified framework. The multi tier contract ontology is envisioned to consist of the following layers:

- Upper Level Contract Ontology
- Specific Domain Level Contract Ontology
- Template Level Contract Ontology

The Upper Level Core contract conceptual model defines all the required and necessary components of a business contract in order to be legally valid. For electronic contracts, we would have additional concepts like digital signatures, public key encryption, security and archiving issues etc.

The second Specific Domain level contract ontology relates to specific contract types. As an illustration, we propose a specific contract type ontology specification for Buy-Sell of commercial goods. In this respect, we draw conclusions and guidance from various internationally adopted legal directives like UNCSIG [40], UNIDROIT [41] principles for commercial transactions, UNCITRAL model contract law [42] etc. The research has been focused specially on the *obligations* and the expected *fulfillment* through the execution of *performance events*. This has been done in order to facilitate easy integration and understanding of the required business process workflow to comply with the contract terms.

The third, template level ontology is visualized as a group of pre defined contractual obligation and their fulfillment patterns. These incorporate standard recommended contract forms like that of ICC's [43] contract model form for International Sale of Perishable Commercial Goods [44], or standard forms for sale of used vehicles etc. Each pertains to the same contract type but yet differ in specific information details contained within them.

This framework allows the Contract Ontology to be flexible, extensible and coherent. It can be easily extended horizontally and further layers are also possible. Moreover, users of the ontology can extract and use parts of the ontology as required for their domain of applicability. Multi Tier Contract Ontology is a hierarchy of ontologies moving from the general to the specific and then down to precise Meta data definitions.

In this paper, we present detailed analysis of a specific contract type, the sale of goods contract type. However, we present a brief overview of the basic concepts, which comprise the Upper Level Core Ontology model.

4.3. Overview Of Upper Level Core Contract Ontology

Any legal contract between two business organizations must have information pertaining to the parties concerned, that is the principal *actors*. Each *actor* has a certain part to play in the whole process of contracting, followed by its business execution. In the contract execution phase the actors may take on the roles of a *seller* or a *buyer*. A contract agreement is drawn up to affect the transfer or performance of certain deeds in exchange for some other deeds or money. This is known as *consideration* in legal terms. *Goods* are a common example for *consideration* in case of business contracts. *Services* or non-disclosure promises are could be other examples of considerations. The actors involved in the contract make certain promises or commitments to each other. These are known as *obligations*, which need to be honored or *fulfilled*. These testify to the intention of the two parties to *perform* to satisfy the conditions agreed for the same *obligation*.

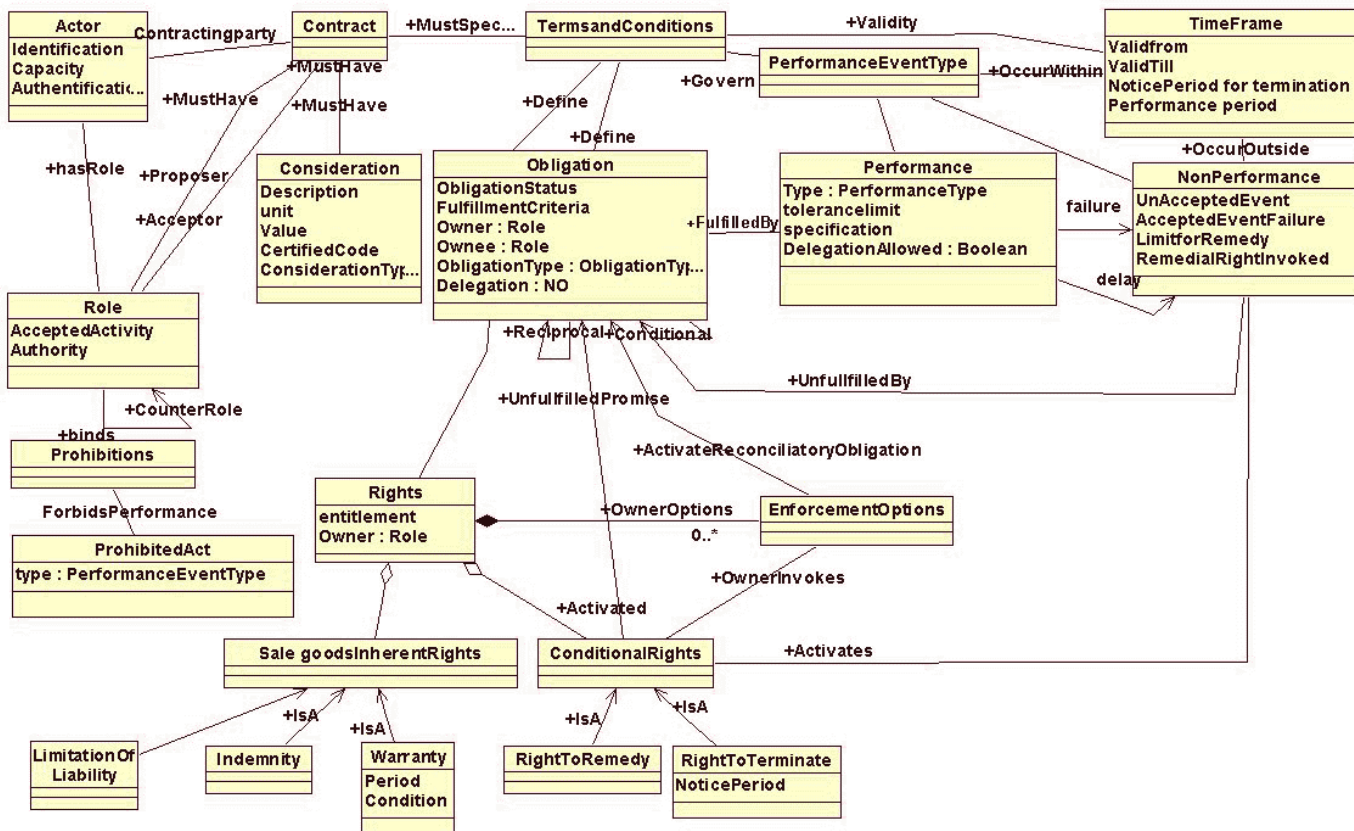


Figure 4: Basic Concepts defined in Upper Level Core Contract Ontology

The legal terms and conditions define all the expected behavior and conditions for satisfactory acceptance of the business behaviors. Like, if a party promises to deliver a pizza made to order within half an hour from the time it is ordered, then the satisfactory condition would be the actual delivery of the pizza that should be conformant to the type of pizza ordered and should be delivered within the time promised. A legal obligation is backed up by the possible consequences in case of failure or *non-performance*. In case the delivered pizza did not match with the type of pizza ordered or it was delivered later than promised, then the customer could reject the pizza or not pay for it or may be demand a replacement of the pizza with another etc. Again these remedial options are also agreed upon and specified in a business contract, to cover all possible eventualities. Thus, along with the definition of the principal actors, their undertaken roles, the object of consideration, the promised obligations, the expected performance events, the fulfillment conditions and terms, the business contract would also have certain *rights, remedies, and prohibitions* too. It is also customary to include terms to limit or protect the liability of the parties involved. Thus the *risks* involved are also defined and appropriately divided and transferred with respect to the execution of business activities.

We explain the above concepts in detail with help of a sale of goods contract type model as discussed in the following section.

4.4 Sale of Goods Business Contract Model

The Upper Core Level ontology defines all the necessary and relevant concepts for any legal business contract. As mentioned earlier, business contracts range over a wide area of application and scopes. Each business contract type has their own specific peculiarities as well as commonly used terms and conditions. However, each of them is a specialization of the same fundamental concepts as defined in the upper level core ontology. Thus the approach methodology adopted for each of the business contract type analyzed is that the upper level core ontology is taken as the point of reference and all specializations and extensions to the basic concepts are modeled based on the identified generic concepts. In other words, each of the shared specific domain level contract ontology inherits from the global upper level core ontology and extends the concepts according to its specific modalities.

For example, in a typical sale and purchase of goods scenario, the principal *actors* are known as *buyer* and *seller*. The *consideration* for business trade transactions are usually exchange of objects in return for other objects or more commonly money. More commonly the *consideration* are referred to as *goods*.

Goods are legally defined as commodities or items of all types, excepting services, which are involved in trade or commerce. *Goods* are characterized by their description, technical specification, type of packaging required, type of cargo etc. We find different international standard vocabularies existing for product categorization like that of the UNSPSC [45], or the CPV [46], which can be readily re, used and adopted within this ontology model. Similarly, UN Recommendation no 21[47] can also be modeled as an integrated or a separate ontology describing codes for types of cargo, packages and packaging materials.

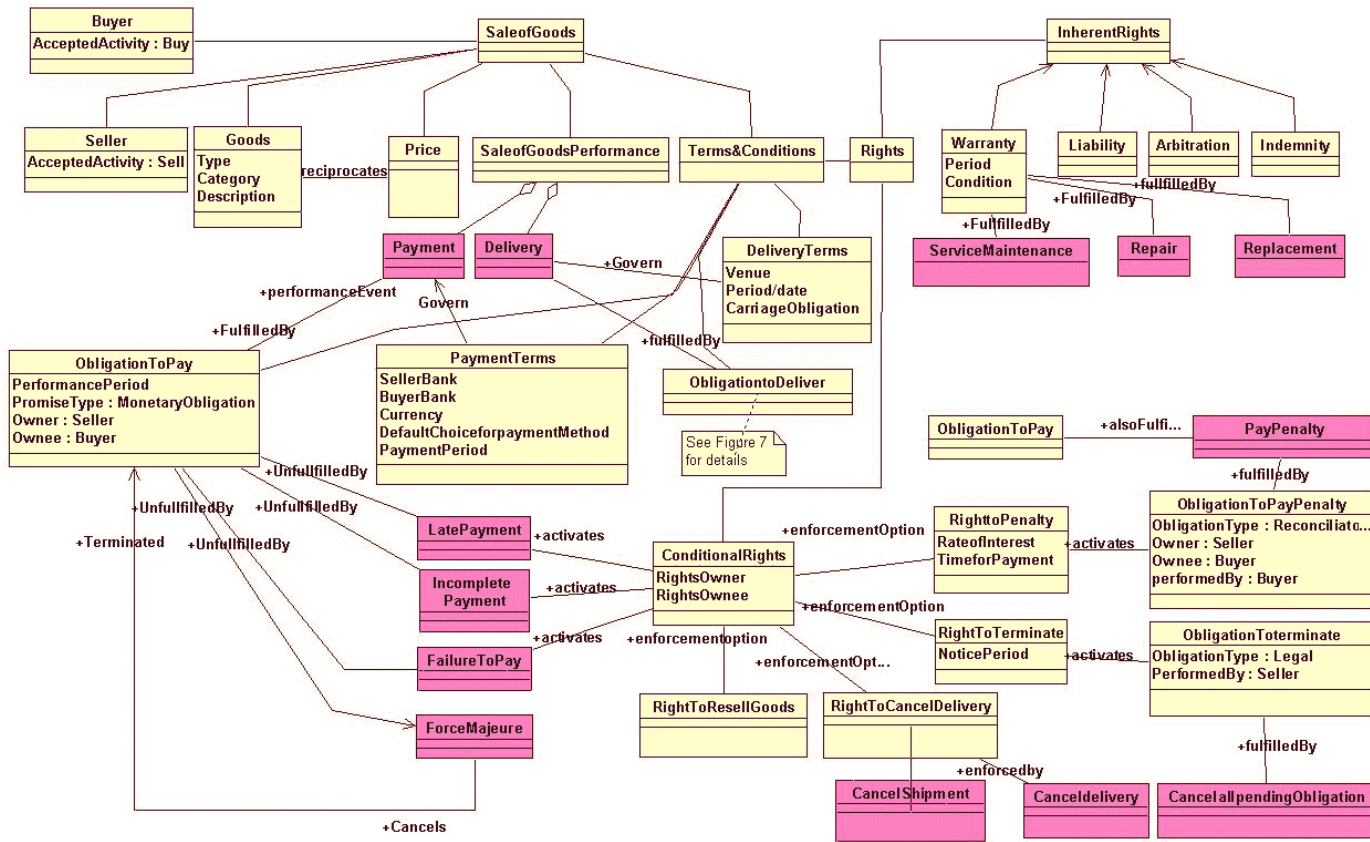


Figure 5: Extract from Sale of Goods Contract Model

Figure 5 shown above is an extract from the conceptual models for a typical sale of goods business contract type.

The *buyer* agrees to pay a certain *price* for the *goods* received. *Price* is usually monetary payment and currency, mode of payment is recommended issues to be discussed and settled between the contract parties. The sale of goods contract should include *payment terms*, which have the details of the agreed payment method and conditions.

Like if the *buyer* is to pay part of the payment amount at the time of ordering and the rest on delivery or if he pays only after delivery. Also payment mode like bank transfer or documentary credits is the preferred method is indicated. Under this concept, we can merge a vocabulary for the Unified Customs and Practice for Documentary Credits [48], which has been discussed and modeled by Lee [17].

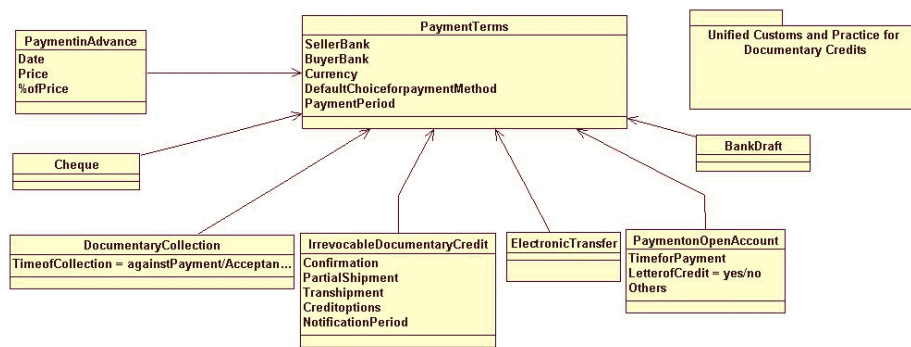


Figure 6: Illustration for common Payment Terms and Methods

Similarly, *Delivery Terms* are also negotiated and expressed in a contract. The *delivery terms* include details regarding the time, venue and choice of place of delivery. Standard delivery terms like International Chamber of Commerce’s INCOTERMS [49], can be used to describe the delivery terms. We see that such terms and conditions, either explicitly or implicitly defines some legal or business *obligations* on the part of the parties concerned. These commitments that bind a *role* player, like the *buyer* or the *seller*, to perform certain acts are called as *obligations*. The primary obligations of a buyer are *obligation to pay* and an *obligation to accept goods* as inferred from INCOTERMS. On the other hand, a *seller* is bound by the *obligation to deliver* s his primary obligation. Obligations need to be *fulfilled*By the execution of expected *performance*. Say for example, a seller’s obligation to deliver could be accepted as fulfilled, if and only if, he carries out the business activities that can be termed as a *delivery*. (See Figure 7 for details)

The actual performance of *delivery* would probably be comprised of several other business activities, which have been shown in figure. Such information, presented in the conceptual models of the sale of goods specific domain ontology, form a useful contribution towards generating the contract workflow models for the business entities. It also helps in business process integration by identifying and exposing shared business activities as possible points of business interoperability and interfaces.

In the extract shown, we see that the seller's obligation to deliver is fulfilled by delivery. In reality, it is quite possible that execution of a promised event is not always successful. A contract generally provides alternatives for handling such exceptions and unacceptable *non-performances*. For example, the *obligation to deliver* may be *UnfulfilledBy* if the delivery is late or delivery is not affected or the delivery is made, but the goods do not conform to the specification as described by the *goods* specification. In such cases, the buyer gets the right to seek redress for the failed performance. The buyer may be presented with one or more *enforcement options*, whereby he could make a choice from the available options, like choosing to have the order cancelled or simply imposing a penalty or opting for a re-delivery of the goods or even having the contract itself terminated. The buyer's choice then binds the defaulter, the seller to a *reconciliatory* obligation to fulfill the chosen form of remedy. The seller has a *secondary obligation* to package the goods he delivers.

Similar detailed analysis has been done for each kind of obligation, rights, or prohibitions that can be included in a typical sale of goods contract. Thus a wide range of possible scenarios involved in a commercial business transaction is covered. This forms an essential knowledge base for the business decision, and strategic planning also. Awareness of possible consequences of non-performance or non-compliance to a contract terms could influence the business process management to a great extent. Contract compliance and performance monitoring have been a crucial concern for most business managements. Multi Tier Contract Ontology is visualized to contribute towards business knowledge management, improving efficiency and performance. On a wider horizon, the proposed ontology framework is visualized as a global network of integrated knowledge resources, exploiting the vast potential of the semantic web to its utmost.

In the following section, the paper presents illustrative proof of concepts for implementing the conceptual models in to machine understandable and searchable formats using semantic web ontology languages like RDFS and DAML.

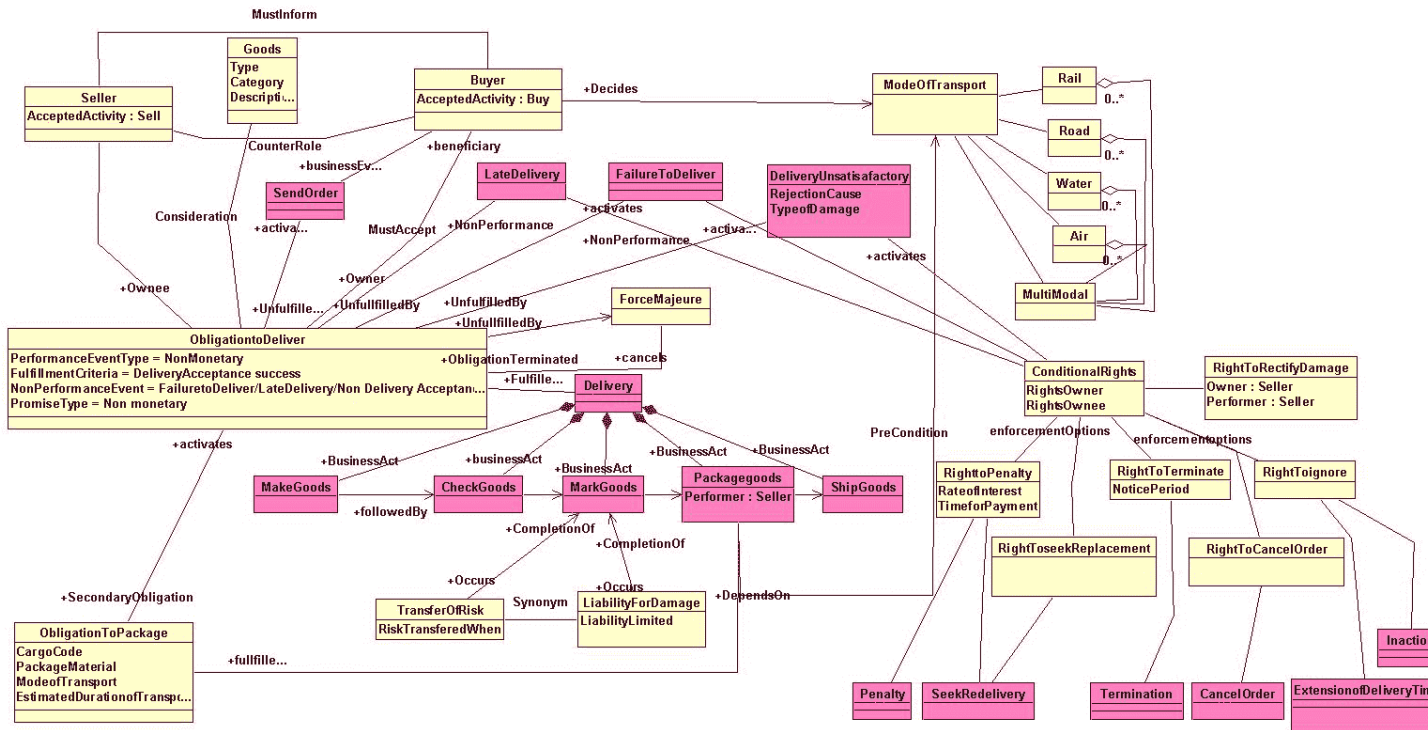


Figure 7: Seller's Obligation to Deliver (expanded view)

5. Proof of Concept for Multi Tier Contract Ontology

5.1. UML to RDFS Transformations

We present a simplified version of our conceptual model for the Upper Core Contract ontology layer, which has most of the main concepts illustrated in figure (8) below. As mentioned in section 2, we chose to represent our conceptual models in UML for the reasons stated therein. We model the concepts as UML classes which could be modeled as Resources in the Resource Description Framework [50]. UML class associations are characteristics or Properties linking the resources to their values in the RDF graphs. The UML association ends are used to depict the property roles or relation to the other resources or classes in this case.

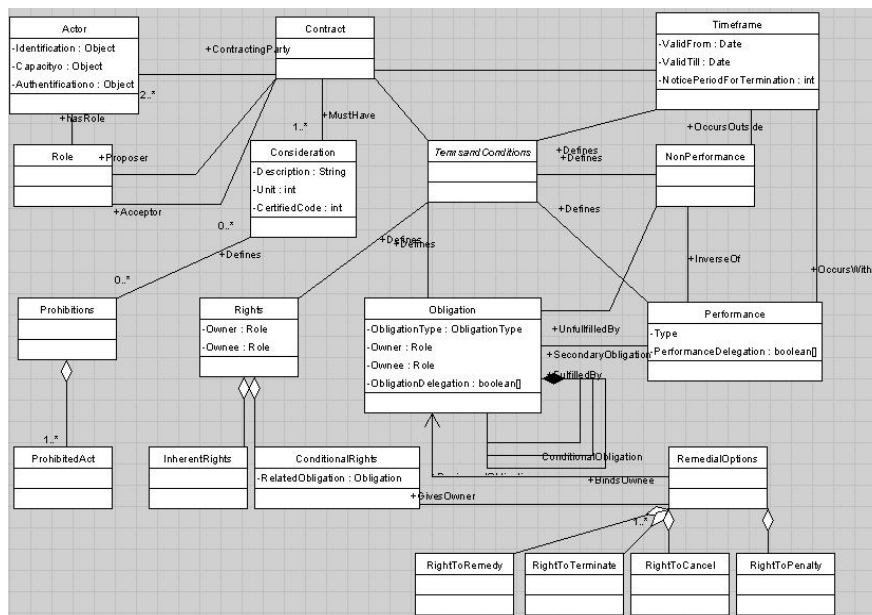


Figure 8: Sample Upper Level Core conceptual model

For example, the concept of an *actor* playing a *role* in context of the contract has been modeled as an association *hasRole* in the above figure.

Using RDF Schema specification, it can be represented as the following extract from our proof of concept RDFS implementation for the Upper Level Core Contract Ontology (Figure 9 below)

```

<rdf:Class
  rdf:about="&contractontology;Actor"
    rdfs:label="Actor">
  <rdf:subClassOf
  rdf:resource="&rdf;Resource"/>
</rdf:Class>
<rdf:Property
  rdf:about="&contractontology;hasRole"
  a:maxCardinality="1" a:minCardinality="1"
  rdfs:label="hasRole">
  <rdf:range
  rdf:resource="&contractontology;Actor"/>
  <rdf:domain
  rdf:resource="&contractontology;Role"/>
</rdf:Property>

```

Figure 9: Extract from RDFS proof of concept implementation

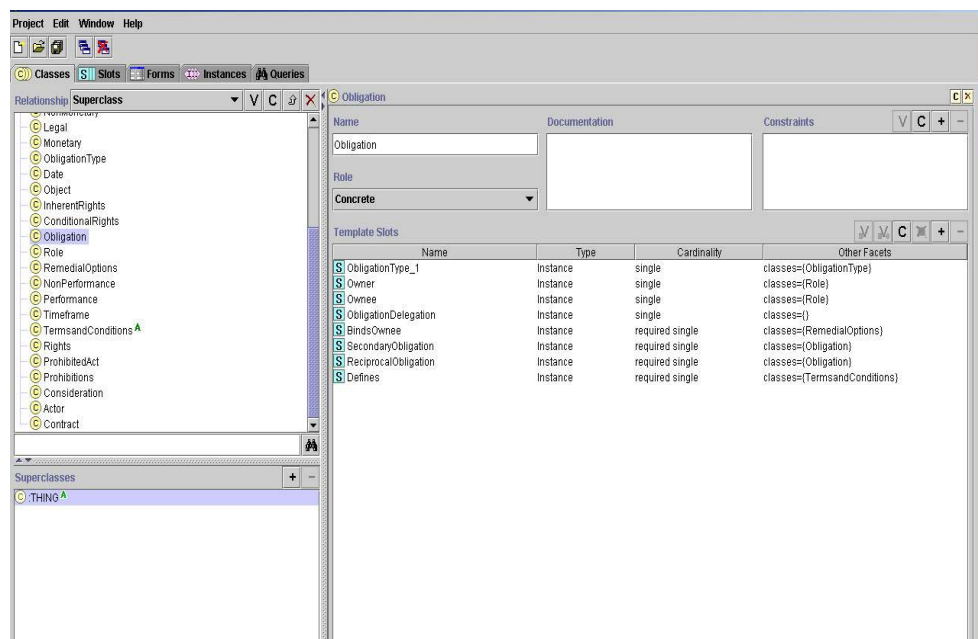


Figure 10: Screenshot from protégé 2000: Using Plugin to transform to RDFS

Current developments in the field of ontology have provided us with several ontology editor tools, including Protégé 2000. Protégé 2000 is a graphical knowledge base editor, which has an increasing user community. The UML plug in [51] supports the import of XMI files and generates the corresponding RDFS and can store in RDFS too. This utility has been used for transforming our UML conceptual models in to RDFS automatically (Figure 10).

5.2. UML to DAML Transformations

The DAML-UML Enhanced Tool (DUET) of the CODIP project provides a UML based environment for the development and manipulation of DAML ontologies. It supports UML to DAML generation for tools like Rational Rose, ArgoUML. Another DAML+OIL plugin is also available from the Protégé community for design and storage of ontology in DAML+OIL. We have used DAML+OIL plugin from SRI[52] for the sample illustrated below. As mentioned in section (3), we have adopted the UML to DAML mapping guidelines as supported by DUET [33,31].

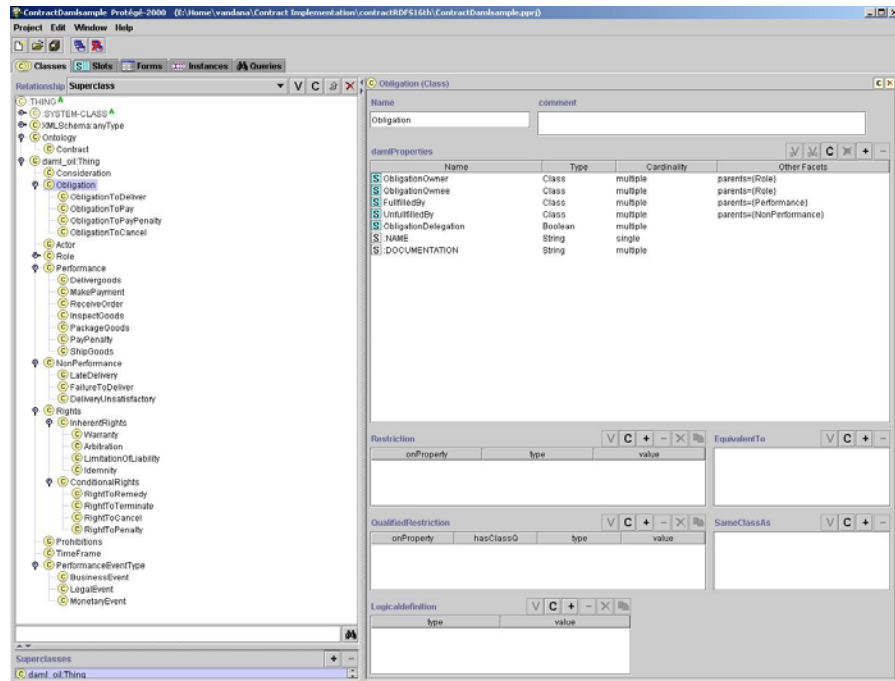


Figure 11: Screenshot of Protégé 2000, using DAML+OIL plugin

The same conceptual model as shown in figure (8), is translated in to DAML+OIL. For example, the concept of the role in the upper layer is reified to that of a buyer in the context of a sale of goods contract model as shown below:

```

<daml_oil:Class rdf:ID="Buyer">
  <rdfs:subClassOf>
    <daml_oil:Restriction>
      <daml_oil:toClass rdf:resource="#Seller"/>
      <daml_oil:onProperty rdf:resource="#hasCounterRole"/>
    </daml_oil:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Role"/>
</daml_oil:Class>

```

Figure 12: Extract from DAML proof of concept implementation

5.3. Observations

From the above discussion, we see that UML has wide usage as an ontology modeling language. The conceptual models are graphical and easy to understand by human users. UML can be translated to other machine understandable forms like RDFS, DAML or databases as mentioned earlier. In case of databases, the concepts would be translated into objects, associations into, and object properties into data properties. It seems a logical choice to represent knowledge in the simplest form and translate it into the required complex language based on the requirement of the application. Our contribution of conceptual models can be reused and extended by other users of the community tailored to their needs and objectives.

DAML+OIL is built on RDFS and has more constructs for expressing more details. It is also closer to natural language and is easy to follow for users not familiar with language constructs like those of RDFS. We found that DAML gave us a greater flexibility in our multi tier contract ontology since we could apply *restriction* to concepts from the upper level easily. DAML also allows us to differentiate between *object type properties* and *data type properties*. Concepts can be expressed in detail including *inverseOf* relations and *equivalentTo* relations.

As the semantic web standardization efforts progress, more such libraries of ready to use language constructs should be available. Structured and predefined common *object type properties* and *data type properties* could be assembled in reusable vocabularies or libraries, for rapid design and development of ontologies in general.

Mapping rules from UML to RDFS and UML to DAML would also need to be standardized, so that everyone uses the same rules and notations. Also, methodologies for transformation from one storage model to another are sorely needed in the semantic web approach. For example RDFS to DAML interchangeability issues need to be addressed. Such methodologies would help the migration from traditional database storage approach towards the semantic web.

6. Applications Of Multi Tier Contract Ontology

Based on the proposed multi tier contract ontology, we are working on a methodology to deduce contract workflow models, which will aid the business entities to organize, restructure or design their business process workflow.

Another direct application of our Multi Tier Contract Ontology is a methodology to monitor and track obligation fulfillment based on the obligation categorization and their states. Detailed specification of obligation states is an ongoing research work.

Other possible application would be to use the proposed knowledge base for automated or semi automated wizard like tools to help monitor contracts or to interpret the required actions for fulfilling obligations etc.

7. Conclusion

The semantic web is meant to establish a network of machine understandable data for software agents and search engines. But, the semantic web can have a far wider

impact and use as a universal medium for commerce. It would be an advantage if all pertinent business information, which so far is stored in traditional databases or other knowledge bases, were also made accessible, machine understandable, and available as a part of the semantic web.

In this paper, we have presented conceptual models for representing contract knowledge in the form of multi tier contract ontology. We have identified the existing gap between business processes and their governing contractual terms and conditions. The proposed ontology is the central knowledge base for deducing the business process workflow, to affect business process interoperability by identifying the shared processes, to improve business performance. One future focus is on mapping to other related ontologies and contract related vocabularies like the UNSPSC, CPV etc.

We have also validated the use of UML as an ontology-modeling tool through our conceptual models and the subsequent proof of concept illustrations using RDFS and DAML. We have presented some observations based on our practical experiences in the implementation process in section 5.3. But we believe that these are minor issues that would be resolved as the semantic web efforts evolve.

Contracting is a large area, and currently we have focused mainly on the contract execution phase. In the approach adopted, there is no distinction between a traditional paper contract and an electronic contract. The ongoing research work is focused on gradual extensions to cover all the phases of the contract life cycle. The Multi Tier Contract Ontology is visualized as a central role player in all the aspects of contracting. In our approach methodology, we have aimed to reuse other related work and methodologies, especially the design guidelines and principles of ontology design.

8. REFERENCES

1. UNCEFACT and OASIS, ebXML, e commerce business standard, <http://www.ebxml.org/>
2. World Wide Web Consortium, Semantic Web, <http://www.w3.org/2001/sw/>
3. Berners-Lee, T., Hendler, J., and Lassila, O., "The Semantic Web," *Scientific American*, May, 2001
4. Howard Smith, The role of ontological engineering in B2B Net Markets, published online at www.ontology.org
5. Zoran Milosevic, Audun Jøsang ,Mary Anne Patton, Theo dimitrakos ,Discretionary enforcement of Electronic Contracts, EDOC 2002
6. Yao-Hua Tan, Walter Thoen. Using Event Semantics for Modeling Contracts. Proceedings of 35th Hawaii International Conference on System Sciences –2002
7. Griffel, M. Boger, H. Weinreich, W. Lamersdorf, M. Merz. Electronic Contracting with COSMOS - How to Establish, Negotiate and Execute Electronic Contracts on the Internet. EDOC '98, 1998
8. Kamalakara Karlapalem, Ajay R Dani and PP. Radha Krishna; A frame Work for Modeling Electronic Contracts; ER 2001, LNCS 2224 pp 193 – 207
9. V Kabilan, P Johannesson, D Rugaimukammu, An ontological approach to Unified Contract Management, to be published in the proceedings of 13th European Japanese Conference on Information Modeling and Knowledge Bases, held on June 6-7th 2003, Kitakyushu, Japan
10. Unified Modeling Language, <http://www.uml.org/>, accessed on 5th June 2003
11. Resource Description Framework Schema, W3C candidate recommendation 27 march 2000, <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>
12. J. Hendler and D. L. McGuinness, The DARPA Agent Markup Language, *IEEE Intelligent Systems journal*, November 2000
13. N Noy, M Sintek, R Ferguson et al , Creating Semantic Web Contents with Protege 2000, *IEEE Intelligent Systems* , 2001 .
14. Metalex , XML standard for mark up of legal resources , <http://www.metalex.nl/> ,last accessed on 18th June 2003
15. Legal RDF Dictionary, European Legal RDF Dictionary initiated by John Mc Clure, <http://www.lexml.de/rdf.htm>

16. Ronald M Lee, A logic Model for Electronic Contracting, 1988
17. Ronald M Lee, facilitating International Contracting: AI Extensions to EDI, published in International Information Systems, January 1992
18. B N Groszof, Yannis Labrou, Hoi Y Chan. A Declarative Approach to Business Rules in Contracts: Couteous Logic Programs in XML, Proceedings of 1st ACM Conference on Electronic Commerce (EC99).
19. B N Groszof, T Poon , SweetDeal :Representing Agent Contracts with Exceptions using XML Rules, Ontologies and Process Descriptions , Proc. Intl. Conf. on the World Wide Web 2003.
20. Steven O Kimbrough , Scott A Moore, On Automated Message Processing in E Commerce and Work Support Systems: Speech Act Theory and Expressive Felicity, Transactions on Information Systems , October 1997
21. A Daskalopulu, Evidence Based Electronic Contract Performance Monitoring. *The INFORMS Journal of Group Decision and Negotiation*. Special Issue on Formal Modeling in E-Commerce, 2002
22. A Daskalopulu & T S E Maibaum Towards Electronic Contract Performance. *Legal Information Systems Applications*, 12th International Conference and Workshop on Database and Expert Systems Applications, 2001 IEEE C. S. Press, pp. 771
23. W van den Heuvel, H Weigand , Cross Organizational Workflow Integration using Contracts, *Business Object Component workshop* ,OOPSLA 2000.
24. Yao-Hua Tan, Modeling Directed Obligations and permission in Trade Contracts.31st Annual Hawaii International Conference on System Sciences, vol 5, 1998.
25. P Levine , J Pomerol,From Business Modeling Based on the Semantics of Contracts to Knowledge Modeling and Management,34th Annual Hawaii International Conference on System Sciences,2001
26. A Goodchild, Charles Herring, Z Milosevic. Business Contracts for B2B. Proceedings of the CAISE00 Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing, 2000
27. N Noy , D McGuinness ,Ontology Development 101: A Guide to Creating Your First Ontology
28. T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Presented at the Padua workshop on Formal Ontology, March 1993.
29. D L McGuinness, Conceptual Modeling for Distributed Ontology Environments, , published online at www.ontology.org also, in the proceedings of the Eight International Conference on Conceptual Structures Logical, Linguistic, and computational issues (ICCS 2000)
30. Cranefield, S., and Purvis, M. "UML as an Ontology Modeling Language," *Proc. of the Workshop on Intelligent Information Integration, 16th Int. Joint Conference on AI (IJCAI-99)*, Stockholm, 1999
31. Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Holmes, W., Letkowski, J., and Aronson M., "Extending UML to Support Ontology Engineering for the Semantic Web." *Proc. of the Fourth International Conference on UML (UML2001)*, Toronto, October 2001
32. Cranefield, S. "UML and the Semantic Web," *Proc. of the International Semantic Web Working Symposium*, Palo Alto, 2001
33. DAML UML Enhanced Tool, available <http://grcnet.grci.com/maria/www/CodipSite/Tools/Tools.html> last accessed on June 24th 2003
34. Components for Ontology Driven Integration Push Project, <http://codip.grci.com/>
35. S Angelov, P Grefen , B2B eContract Handling – a survey of projects, papers and standards CTIT Technical Report,University of Twente , 2001
36. W.M.P. van der Aalst. The application of PetriNets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21--66, 1998
37. E Sivaraman, K Kamath, On the use of Petri nets for business process modeling', *Proceeding of the 11th Annual Industrial Engineering Research Conference*, Orlando, FL., May 2002
38. <http://www.lectlaw.com/def/g012.htm>, Duhaima, Lloyd, Duhaime's law dictionary, www.duhaime.org
39. Guarino, N. 1992. Concepts, Attributes and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases. *Data & KnowledgeEngineering*, 8: 249-261.
40. United Nations Convention on Contracts for International Sale of Goods , 1980, <http://www.cisg-online.ch/cisg/conv/convuk.htm>, last accessed on June 24th 2003
41. International Conventions on Unification of Private Law,Uniform law on international sale of goods , 1964. <http://www.unidroit.org/english/conventions/c-ulis.htm>
42. UNCITRAL : United Nations Commission on International Trade And Law. <http://www.uncitral.org/english/texts/>
43. International Chamber of Commerce, <http://www.iccwbo.org>
44. ICC International contract for sale of goods, published by ICC books, 2002
45. United Nations Standard Products and Service Codes (www.unspsc.org)
46. Common Procurement Vocabulary (CPV) <http://simap.eu.int/EN/pub/src/main5.htm>

47. UN Recommendation no 21 : Types of cargo, packages and packaging materials.
<http://www.unece.org/cefact/rec/rec21e4a.htm>
48. Unified Customs and Practice for Documentary Credits UCP 500, ICC publication,
<http://www.iccwbo.org/home/banking/778rev9.asp>
49. Jan Ramberg; ICC Guide to Incoterms 2000. Understanding and Practical Use; International Chamber of Commerce 2000
50. Resource Description Framework, www.w3.org
51. UML storage backend , Holger Knublauch, Stanford University, available online for download at
<http://protege.stanford.edu/plugins.html>
52. DAML +OIL backend , developed by SRI , Grit Denker, John Pacheco, Ouissem Ghorbel available online at <http://protege.stanford.edu/plugins.html> , last accessed on June 24th 2003.

The Visual Semantic Web: Unifying Human and Machine Semantic Web Representations with Object-Process Methodology

Dov Dori

Technion, Israel Institute of Technology, Haifa 32000, Israel

dori@ie.technion.ac.il, and

Massachusetts Institute of Technology, Cambridge, MA 02139, USA

dori@mit.edu

Abstract

The Visual Semantic Web (ViSWeb) paradigm enhances human accessibility to the current Semantic Web technology by enabling the visualization of knowledge. Arguing against the claim that humans and machines need to look at different knowledge representation formats, Object-Process Methodology (OPM) is shown to enable modeling of systems in a single graphic and textual model. ViSWeb provides for representation of knowledge over the Web in a unified way that caters to humans as well as machines. ViSWeb is developed as an OPM-based layer on top of XML/RDF/OWL to express knowledge visually and in natural language. Both the graphic and the textual representations are strictly equivalent. Being intuitive yet formal, they are not only understandable to humans, but are also amenable to computer processing. The advantages of the ViSWeb approach include equivalent graphic-text knowledge representation, visual navigability, semantic sentence interpretation, specification of system dynamics, and complexity management. The ability to use such bimodal knowledge representation that is both human understandable and machine processable is a major step forward in the evolution of the Semantic Web.

1. The Human-Machine Language Orientation Dilemma

The development of the Semantic Web is driven by the assumption that humans and machines must each use a different format of knowledge representation. For example, the RDF [7] introduction reads: "The World Wide Web was originally built for human consumption, and although everything on it is *machine-readable*, this data is not *machine-understandable*" (emphasis in source). Berners-Lee, Hendler and Lassila [5] have noted that "*the Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*" Constructing a comprehensive Web-based knowledge management system must reconcile this human-machine language orientation dilemma. The bulk of knowledge that continues being gathered on the Web in an ever accelerating rate is expressed in free natural language, which is currently indigestible to machines (e.g. [11]). Still, current technologies for Web-based knowledge management are developed based on the premise that while humans prefer natural language, machines must use XML-like scripts, which humans have to invest great efforts to decipher: "... *instead of asking machines to understand people's language, the new technology, like the old, involves asking people to make some extra effort, in repayment for which they will get substantial new functionality.*" [4]. The ViSWeb approach provides this functionality without requiring that effort. This is true not only for RDF [21, 38], but also for OWL, the Web Ontology Language [32], which reads: "*In order to map this (the*

*World Wide Web] terrain more precisely, computational agents require machine-readable descriptions of the content and capabilities of web accessible resources. These descriptions **must be in addition** to the human-readable versions of that information."*

In contrast, the Visual Semantic Web (ViSWeb) approach is founded on the premise that human and machine Web-based knowledge need not necessarily be represented by two distinct formats. ViSWeb is based on Object-Process Methodology (OPM) [14]. Using a bimodal representation of graphics and text, OPM models knowledge about systems of various types and different complexity levels in a single model, which integrates structure and behavior. OPM, described in more detail below, combines a subset of natural language, called Object-Process Language (OPL), with a formal, yet intuitive graphic model, a set of one or more Object-Process Diagrams (OPDs) of exactly the same knowledge expressed in OPL. This dual graphic-textual representation constitutes a solid foundation for generic knowledge representation over the Web.

2. Combining Graphic and Textual Knowledge Representations

A powerful knowledge modeling and communication modality, which is complementary to language, is graphics. Diagrams are often invaluable for describing models of abstract things, especially complex systems. The fact that people from the early caveman days to date have been using some kind of sketching or diagramming technique to express their knowledge or ideas is a testimony to the viability of the graphic representation. However, such representation of our knowledge is valuable only if it is backed by a comprehensive and consistent modeling methodology. Such methodology is essential if we want to represent knowledge, understand complex systems in any domain, and communicate our understanding to others. An accepted diagramming method has the potential of becoming a powerful modeling tool if it constitutes an unambiguous language. In such visual formalism, each symbol must bear defined semantics and the links among the symbols must unambiguously convey some meaningful information that is clearly understood by the diagram readers.

Knowledge Representation Approaches

A number of knowledge representation approaches have been designed with the goal of graphically and/or textually representing knowledge aimed at facilitating human understanding and communication of knowledge. These approaches include concept maps [2, 18, 19, 26, 27], semantic networks [22, 6, 16, 25], XML Topic Map (XTM) [31] conceptual graphs (CGs) [33, 34, 9, 13, 23], Knowledge Interchange Format (KIF) [20], Cyc [11], the Common Logic (CL) standard initiative [35], Unified Modeling Language (UML) [21, 16, 30], and Object-Process Methodology (OPM), which is the basis for the Visual Semantic Web presented in the paper.

Object-Process Methodology

Most interesting and challenging systems are those in which structure and behavior are highly intertwined and hard to separate. Motivated by this observation, Object-Process Methodology (OPM) [14] is a holistic approach to the study and development of systems, which integrates the object-oriented and process-oriented paradigms into a single frame of reference. Structure and behavior, the two major aspects that each system exhibits, co-exist in the same OPM model without highlighting one at the

expense of suppressing the other. Due to its structure-behavior integration, OPM provides a solid basis for modeling complex systems in general and those documented through the Semantic Web in particular. The elements of the OPM ontology are entities and links. Entities, the basic building blocks of any system modeled in OPM, are of three types: *objects* with *states*, and *processes*. *Objects* are (physical or informatical) things that exist, while *processes* are things that transform objects. *Links* can be structural or procedural. *Structural links* express static, time-independent relations between pairs of entities. The four fundamental structural relations are Aggregation-participation, generalization-specialization, exhibition-characterization, and classification-instantiation. *Procedural links* connect entities (objects, processes, and states) to describe the behavior of a system.

Behavior is manifested in three major ways: (1) processes can *transform* (generate, consume, or change the state of) objects; (2) objects can *enable* processes without being transformed by them; and (3) objects can *trigger* events that invoke processes if some conditions are met. Accordingly, a procedural link can be a transformation link, an enabling link, or an event link. A *transformation link* expresses object transformation, i.e., object consumption, generation, or state change. An *enabling link* expresses the need for a (possibly state-specified) object to be present, in order for the enabled process to occur. The enabled process does not transform the enabling object. An *event link* connects a triggering entity (object, process, or state) with a process that it invokes. The event types that OPM supports include state entrance, state change, state timeout, process termination, process timeout, reaction timeout, and external events. External events include clock events and triggering by environmental entities such as a user or an external device.

Two semantically equivalent modalities, one graphic and the other textual, jointly express the same OPM model. A set of inter-related Object-Process Diagrams (OPDs), showing portions of the system at various levels of detail, constitute the graphical, visual OPM formalism. Each OPM element is denoted in an OPD by a symbol, and the OPD syntax specifies correct and consistent ways by which entities can be connected via structural and procedural links, each having its specific, unambiguous semantics. OPM assigns special graphical symbols for a selected set of relations (similar to UML class diagrams, only for a larger set of relations). OPCAT (Object-Process CASE Tool) [5] is a Java-based software environment that supports OPM system modeling and evolution. The Object-Process Language (OPL), defined by a context-free grammar, is the textual counterpart modality of the graphical OPD set. OPL is a dual-purpose language, oriented towards humans as well as machines. Catering to human needs, OPL is designed as a constrained subset of English, which serves domain experts and system architects, jointly engaged in analyzing and designing a system, such as an electronic commerce system or a Web-based enterprise resource planning system. Every OPD construct is expressed by a semantically equivalent OPL sentence or phrase. This dual representation of OPM increases the processing capability of humans according to the cognitive theory of multimodal learning proposed by Mayer [8, 1]. The knowledge that OPM can represent is not restricted to just structural, as in CGs and most other knowledge representation formats. It can also be procedural, showing temporal order and enabling cause and effect analysis. Designed also for machine interpretation through a well-defined set of production rules, OPL provides a solid basis for automating the generation of the designed application. Indeed, OPCAT currently generates complete Java code from OPL script and enables the generation of any other formal language.

Unlike the graphic representation in the Semantic Web, which is an auxiliary means to illustrate the machine-oriented, XML-based content, OPDs constitute a complete and consistent visual formalism that goes hand in hand with the OPL. A basic OPM principle is the text-graphic equivalence principle: *Anything that is expressed graphically by an OPD is also expressed textually in the corresponding OPL paragraph, and vice versa.* Following this principle, our goal in developing ViSWeb, the Visual Semantic Web, as in OPM in general, is to specify a system by a set of inter-related Object-Process Diagrams and their completely equivalent corresponding OPL paragraphs. This equivalence implies that both modalities, the graphic and the textual, contain exactly the same information, albeit in two different ways of expression. Due to this complete equivalence, each can be reconstructed from the other. As noted, in spite of the apparent graphics-text redundancy, from a human factors engineering viewpoint, these two modalities activate different cognitive processes and therefore reinforce the understanding of each other and of the system as a whole.

A major problem with most graphic modeling approaches is their scalability: As the system complexity increases, the graphic model becomes loaded with shapes and cluttered with links that cross each other in all directions. The limited channel capacity [24] is addressed by OPM and implemented in OPCAT with three abstraction/refinement mechanisms. These enable complexity management by providing for the creation of interrelated OPDs (along with their corresponding OPL paragraphs) that are limited in size, thereby avoiding information overload and enabling comfortable human processing.

3. Concept Graphs vs. Object-Process Diagrams

The dual graphic and equivalent natural language representation of the single OPM model is both human understandable and machine processable. A modeling system that comes closest to OPM in its dual graphic-textual representation is Conceptual Graphs (CGs), based on [29, 33]. Table 1 compares both the graphic and the textual CG and OPM model representations of the system represented by the natural language sentence "John is going to Boston by bus." Graphically, the CG model has a compact set of symbols: boxes for concepts, ovals for relations, and arrows for the directed links between concepts and relations. OPM has a richer set of symbols, allowing it to be more expressive. While OPDs use boxes and ovals, like CGs, their semantics is different, denoting respectively objects and processes rather than CG concepts and relations. OPM relations are expressed via the various link types. For example, the link from **John** to **Going**, which ends with the black circle, is the agent link, denoting that the **Person** called **John** is the agent of (the human who executes) the process **Going**. Similarly, the link from **Bus** to **Going**, which ends with the white circle, is the instrument link, denoting that the **Bus** is the instrument of the process **Going**. As noted earlier, these special symbols save the need to annotate these links textually. Agent and instrument links are procedural links: they connect an object and a process.

Other OPD procedural links are the result, consumption, and effect links. In addition to procedural links, OPDs feature a family of structural links, each of which connects an object with an object. An example of a structural link in the OPD in Table 1 is the exhibition-characterization relation, denoted by the black-in-white triangle along the line connecting the **Person John** to the **City**. The exhibition-characterization symbol from **Person** to **Location** states that **Location** is an attribute of **Person**,

The CG makes no underlying semantic difference between "John", "Boston" and "Bus" on one hand and "Go" on the other hand: all are concepts. In OPM there is a principal difference between these two entity types: The OPM ontology stipulates that objects are things that exist, while processes are things that transform objects by changing their state or by generating/consuming them. The inability of CGs to distinguish between objects and processes is a major hindrance to enhanced expressive power with respect to system dynamics: CGs may be fine for declarative assertions, i.e., statements about what exists in the world and how what exists related to other things that exist. However, when it comes to describing the dynamics of the system, namely its time-dependent behavior, CGs lacks the basic concept of process and makes no distinction between objects and processes, relating to all as concepts. A somewhat similar difference exists between OPM and the OO paradigm, in which Object is the only top-level concept, and processes can only be expressed as operations that objects own.

Table 1. Comparison between the CG (left) and OPM (right) models of "John is going to Boston by bus."

<pre>[Go] - (Agnt) -> [Person: John] (Dest) -> [City: Boston] (Inst) -> [Bus] .</pre>	<p>The Person John exhibits the Location City. City is Boston. John handles Going. Going requires Bus. Going changes City to Boston.</p>

Comparing the two textual representations in Table 1 reveals that while CGs may bear direct mapping to language, they do not translate to any subset of natural language, but rather to a symbolic representation. The OPL sentences, on the other hand, are understandable and the OPL paragraph above can indeed be summarized by the original sentence. Note that the OPL script unfolds a small five-sentence "story." In order to set up the framework for this story, **John** was assigned the attribute **City**, which is an instance of the class **Location** and is assigned the value **Boston**. OPL sentences are written in plain English that is easily readable and understandable to humans with no prior training whatsoever. The same cannot be said about the LF script of CGs, as demonstrated in the bottom left of Table 1. Another quantum leap is still required to convert this script to a natural language sentence like "John is going to Boston by bus."

Summarizing the main differences between CGs and OPM we note that:

- (1) The symbol set of CGs is more compact than that of OPDs but expressing the same complex semantics in CGs requires to use at least twice the number symbols required in OPD, yet the semantics is more explicit in OPDs.
- (2) The CG formalism is probably better than OPM with respect to support for logic. While OPM does allow AND, OR, and XOR relations, as well as

Boolean objects, it currently does not have the notion of quantifiers and cannot deduce new knowledge from existing knowledge. This is an issue that is being considered for inclusion in the next OPM versions.

- (3) The text generated by OPM, the OPL paragraph, is a subset of English, enabling any English speaker to readily understand it, while the LF, the textual form of CGs is still in symbolic form that is not legible to untrained humans.
- (4) CGs are purely declarative and have no notion of system dynamics, which is a major feature of OPM.
- (5) CGs are not scalable, while OPM has this capability via its scaling mechanisms.

In view of the fundamental differences, the choice of OPM as a basis for the Visual Semantic Web is quite obvious. We continue with a brief survey of RDF and the use of graphics in the Semantic Web.

4. The Semantic Web and the RDF Syntax

RDF, the Resource Description Framework [21, 8] aims at making the knowledge resources that are available on the Web amenable to machine interpretation, compilation, or other types of processing, by imposing some structure on the pieces of knowledge. RDF provides a basis for a number of emerging initiatives, such as the Dublin Core Metadata Initiative [17], an open forum engaged in the development of interoperable online metadata standards. The RDF Syntax document [21] introduces a model for representing RDF metadata as well as a syntax for encoding and transporting this metadata for interoperability of independently developed Web servers and clients. The syntax it presents uses the eXtensible Markup Language (XML), because one of the goals of RDF is to enable specifying semantics for data based on XML in a standardized manner. RDF and XML are complementary in that RDF is a model of metadata and only addresses encoding issues by reference. Such issues include internationalization and character sets, required by transportation and file storage. More importantly, the XML syntax of RDF is only one of the possibilities for encoding RDF and, as noted in [21], alternate ways to represent the same RDF data model may emerge. Indeed, this paper proposes an OPM-based alternative on top of the XML syntax that is human- and machine-oriented at the same time. This syntax enables bimodal, dual graphic-textual representation of the system model for human consumption, while possessing a level of formality that makes it amenable to machine processing.

The Semantic Web makes only limited use of graphical models. In RDF these are directed graphs, where subjects and objects are nodes, and predicates are labels along the edges, directed from a subject to an object. However, since the Semantic Web in its current form and philosophy is geared primarily to cater to needs of machines, and since machines do not need to read diagrams, the visual aspect of the information and knowledge modeling is not well developed. Semantic Web documents show few graphs early on but then abandon them and focus on the XML-based syntactical aspects of the machine-oriented language. RDF is a model for representing named properties and property values [21]. RDF properties may be thought of as attributes of resources and, in this sense, correspond to traditional attribute-value pairs. RDF properties represent relationships between resources, and RDF Schemas, which are instances of RDF data models, are Entity-Relationship (ER) diagrams. The basic RDF data model consists of

the following three object types: **Resource** – Anything described by an RDF expression; **Property** – A specific aspect, characteristic, attribute, or relation used to describe a resource; and **Statement** – A specific resource – the *subject*, together with a named property – the *predicate*, plus the value of that property for that resource – the *object*. Following [8], consider first the sentence "Ora Lassila is the creator of the resource <http://www.w3.org/Home/Lassila>." Translated to RDF format, this sentence can be interpreted as having the subject (resource) <http://www.w3.org/Home/Lassila>, the predicate (property) creator, and the object (literal) "Ora Lassila". RDF uses directed graphs to specify these notions graphically, where subjects and objects are nodes, and predicates are labels along the edges, which are always directed from a subject to an object, as in Figure 1. A resource node in the graph is drawn as an oval (ellipse), while a literal node is drawn as a rectangle.

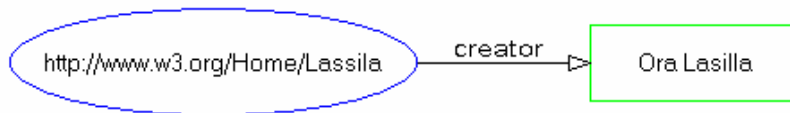


Figure 1. A simple RDF graph example from [21]

the graph in Figure 1 is to be interpreted as "<http://www.w3.org/Home/Lassila> has creator Ora Lassila", and in general "*<subject> HAS <predicate> <object>*". Applying the RDF/XML Validation Service [38] using the RDF/XML script in Table 2 yields the graph in Figure 2.

Table 2. The RDF/XML script that generates the graph in Figure 2.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```



Figure 2. The RDF graph of the data model listed in Table 2 generated automatically by the RDF/XML Validation Service [37]

The Semantic Web is based on a principle similar to that of OPM, where relations (called properties in the SW nomenclature) are edges of a graph rather than nodes, as in CGs. The Visual Semantic Web [16] (ViSWeb) alternative to the RDF/XML knowledge representation [13] takes advantage of the integrated graphic-text formal yet intuitive infrastructure that OPM provides. Figure 3 is a ViSWeb spec (Visual Semantic Web specification), which expresses the example in Figure 1 in a bimodal fashion, both as an Object-Process Diagram (OPD) and an Object-Process Language (OPL) text. The OPD contains two object instances: **Ora Lassila** and [WWW.w3.org/Home/Lassila](http://www.w3.org/Home/Lassila). To conform to

OMG UML 1.4 [10, 15], object names (i.e., instances of object classes) in OPDs are underlined, as in UML object diagrams.

A tagged structural link, depicted as an open arrow, such as the one pointing from **Person** to **URI** in Figure 4, expresses the nature of the relation between these two objects. The tag is the text recorded along the structural link. The value of this tag is '**is the creator of**'. The value is a phrase, such that when the name of the source object, **Ora Lasilla** (an instance of the class **Person**) is concatenated with the tag value (i.e., the phrase) '**is the creator of**' followed by the name (value) of the **URL**, one automatically gets the following OPL sentence, which is also generated automatically by OPCAT and recorded at the bottom of the OPD in Figure 3.

Ora Lasilla is the creator of WWW.w3.org/Home/Lassila.

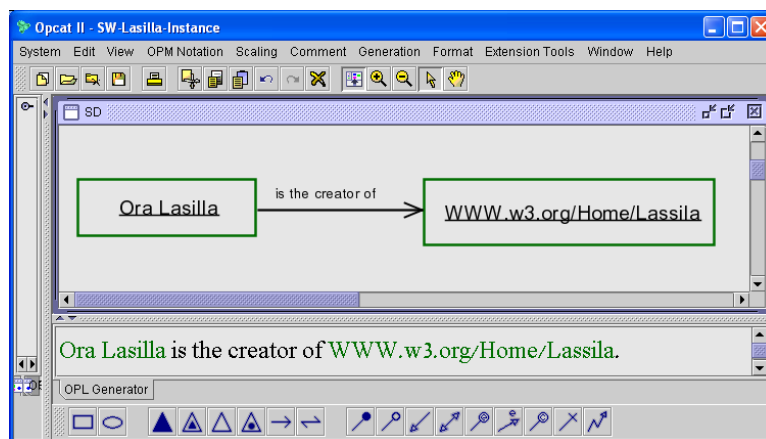

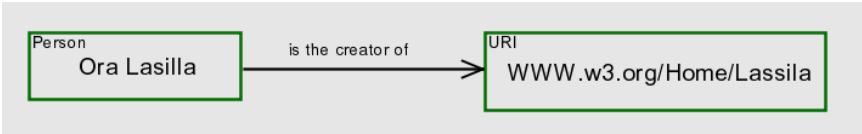


Figure 3. The example in Figure 1 expressed as a ViSWeb spec (Visual Semantic Web specification), consisting of an Object-Process Diagram (OPD) at the top window and its corresponding, automatically-generated Object-Process Language (OPL) one-sentence paragraph at the bottom window.

The automatic generation of the OPL sentence in this simple case was done by concatenating the name of the object at the source of the tagged structural link, **Ora Lasilla**, with the text string of the structural link's tag, **is the creator of**, with the name of the destination object, WWW.w3.org/Home/Lassila.

In RDF terminology, this OPL sentence is a *statement*, in which a specific resource – the *subject*, **Ora Lasilla** in our case, together with a named property – the *predicate*, '**is the creator of**' in our case, plus the value of that property for that resource – the *object*, WWW.w3.org/Home/Lassila in our case. Each word in an object (and process) name is capitalized, while in link names (tags) they are not. As Figure 5 shows, names of objects and link names (tags) appear in different colors (which can be set by the user). Even though there are spaces between the words, using the capitalization rule above it is possible to mechanically parse the sentence even without the human-oriented color cues. Table 3 compares RDF and OPM with respect to this example. For each method, the three elements and the respective parts of the example are written first, and below them are the graphical and textual representations of the RDF graph in Figure 1 and the OPD in Figure 5.

Table 3. Comparison between RDF and OPM applied to the example in Figure 1 and Figure 5

RDF/XML	<i>Object (domain):</i> Http://www.w3.org/Home/Lassila	<i>Predicate (property):</i> Creator	<i>Subject (range):</i> Ora Lassila
	<i>Graphics:</i> 		
<i>Text:</i> <pre><?xml version="1.0"?> <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:s="http://description.org/schema/"> <rdf:Description about="http://www.w3.org/Home/Lassila"> <s:Creator>Ora Lassila</s:Creator> </rdf:Description> </rdf:RDF></pre>			
OPM/VisWeb	<i>Source object:</i> Ora Lasilla	<i>Structural link tag:</i> is the creator of	<i>Destination object:</i> WWW.w3.org/Home/Lassila
	<i>Graphics:</i> 		
<i>Text:</i> Ora Lasilla is the creator of WWW.w3.org/Home/Lassila.			

While the OPM model still does not account for namespaces, which are treated in the sequel, comparing this OPL/VisWeb sentence to the RDF/XML script in Table 3, it is not difficult to see the benefit of using a more human-readable version, which, while still machine-readable, does not require the human reader to act like a mechanical XML parser.

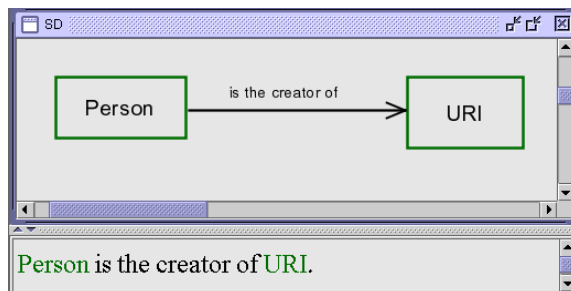


Figure 4. A VisWeb schema showing the class OPD along with its corresponding OPL sentence, to which the VisWeb spec in Figure 3 conforms

5. The ViSWeb Schema: A Template for a ViSWeb Spec

The lines under the two objects in Figure 3 denote the fact that these are object instances, not object classes. The class information is still missing in this OPD. Figure 4 shows a *ViSWeb schema*, an OPD-OPL template that contains class information. Note that the ViSWeb schema follows the OPM text-graphic equivalence principle: the OPD and the OPL paragraph are completely reconstructible from each other. Each ViSWeb spec conforms to a ViSWeb schema. Thus, the ViSWeb schema in Figure 3 conforms to the ViSWeb spec in Figure 4. This ViSWeb schema can be thought of as a template that expresses a rule. In our example, the rule stipulates that the source (which in RDF schema terminology is termed the *domain*) of the relation (the RDF *predicate*) '**is the creator of**' is an object that belongs to the class **Person**, and that the destination (*range*) of that relation is an object that belongs to the class **URI**.

Having established the **Person-URI** ViSWeb schema, we can now use it to add the object instance for each of the two classes. This is done in the instantiated schema shown in Figure 5, where the ViSWeb schema of Figure 4 and the ViSWeb spec of Figure 3 are combined. The combination uses the OPM classification-instantiation relation, which is denoted as a bulleted triangle whose tip is linked to the class and whose base is linked to the instance. Note that the instances **Ora Lasilla** and WWW.w3.org/Home/Lasilla need not be underlined here to denote that they are instances. The underlining of the instance names is only mandatory if the class information is not present in the OPD, but here this is indicated by the classification-instantiation links from the classes to the respective instances.

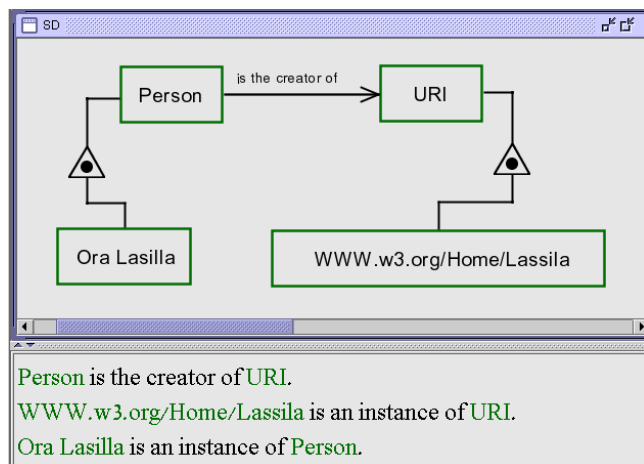


Figure 5. The instantiated ViSWeb schema generated by adding the instance specification of Figure 3 to the class information in the ViSWeb schema in Figure 4.

The OPL paragraph of the OPD in Figure 5 is shown in Figure 5 as well:

Person is the creator of URI.
WWW.w3.org/Home/Lassila is an instance of **URI**.
Ora Lasilla is an instance of **Person**.

Note that predicates (such as **is the creator of**) do not have explicit instance names that are distinct from their class names. Thus, for example, we use the same predicate in the tagged structural relation '**is the creator of**' from the class **Person** to the class **URI** is

inherited to their respective instances, so there is an implicit tagged structural relation with the same tag, 'is the creator of', from **Ora Lasilla**, an instance of the class **Person**, to WWW.w3.org/Home/Lassila, an instance of the class **URI**. Applying template information and using chaining rules one can establish that **Ora Lasilla is the creator of WWW.w3.org/Home/Lassila** although this is not explicit in the OPM model in Figure 5. However, the instantiated ViSWeb schema in Figure 5 is space-consuming and it requires the reader to realize the existence of the implicit tagged structural relation. These two problems are solved in the compact version of the instantiated ViSWeb schema of Figure 5, shown in Figure 6.

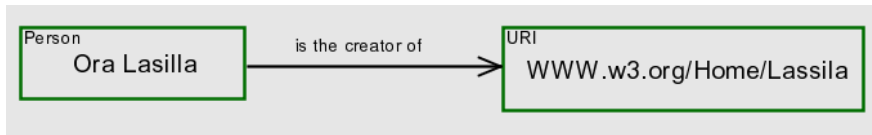


Figure 6. A compact version of the instantiated ViSWeb schema in Figure 5

The OPL paragraph that corresponds to the OPD in Figure 6 is also more compact than the three-sentence OPL paragraph of Figure 5, as it consists of just one sentence:

The **Person Ora Lasilla is the creator of the URI WWW.w3.org/Home/Lassila.**

This sentence combines the OPL schema sentence from Figure 3, which is "**Person is the creator of URI.**" with the OPL instance sentence Figure 4, which is "**Ora Lasilla is the creator of WWW.w3.org/Home/Lassila.**" In the new OPL sentence, which reflects both the classes and the instances, we added the class information of both **Ora Lasilla**, which is **Person**, and of WWW.w3.org/Home/Lassila, which is **URI**. **Ora Lasilla** is classified in the OPL sentence as belonging to the class **Person** by preceding the name of the instance by the reserved word "The" followed by the class name **Person**. Likewise, the string WWW.w3.org/Home/Lassila was classified as belonging to the class **URI** by preceding the value of the string by the reserved word the followed by the class name **URI**. The corresponding quoted sentence is The 'Person' 'Ora Lasilla' 'is the creator of' the 'URI' 'WWW.w3.org/Home/Lassila'.

6. OPM Namespace Specification

Namespaces [7] are definitions of terms and relations of some domain ontology. The OPL sentence "The **Person Ora Lasilla is the creator of the URI WWW.w3.org/Home/Lassila.**" does not specify the namespaces which contain the definitions of **Person**, **URI**, and the structural link tag (predicate) 'is the creator of'. In contrast, the XML script in Table 3 does mention two namespaces, *rdf* and *docs*. The namespaces, which are part of the XML tags, enable us to know that we are looking at a *Description* in the sense defined in the *rdf* namespace definition, that the value of its *about* attribute is "<http://www.w3.org/Home/Lassila>", and that the value of the *Creator* entity, as defined in the *docs* namespace, is *Ora Lassila*. This information is clearly richer than what can be extracted from the RDF graph in Figure 1, since that graph does not specify any namespace information.

The RDF graph is only an auxiliary means to make it easier for humans to "get the picture." It is not required to contain all the information expressed by the corresponding XML script and therefore cannot replace it (although the RDF Validator [37] does so,

albeit in a manner that is not very user friendly). The OPM text-graphics equivalence principle mandates that any piece of information contained in the OPL paragraph that corresponds to an OPD be represented in the OPD, and vice versa, making the OPD and its OPL paragraph fully equivalent in terms of information content. To keep up with the text-graphics equivalence principle, we introduce the concept of namespace to both the OPD and the OPL.

Let us assume that the subject **Person** and the object **URI** are both defined in the namespace whose name is **Semantic Web** and whose URI is WWW.SemanticWeb.org/definitions. We further assume that the predicate 'is the creator of' is defined in the namespace whose name is **Documents** and whose URI is WWW.Documents.org/definitions. The OPD in Figure 7 elaborates on that of Figure 6, as it provides the complete namespace information. The ViSWeb convention is to stack all the namespaces used in the OPD at its top left corner. Each namespace is recorded in an object box, with the string "Namespace: <blank> <namespace_name>" appearing at the top left corner of the box and the corresponding URI recorded at the bottom of the box. Here, <namespace_name> is the name of the namespace. Two namespaces names appear in Figure 7: **Semantic Web** and **Documents**. Using these two namespace specifications, instances in an OPD can be annotated not just with the class specification, as in Figure 6, but also with the namespace within which the class is specified, preceding the class name, as in Figure 7. Thus, **Semantic Web: Person** is the complete namespace and class specification of the **Person** instance **Ora Lasilla**, and **Semantic Web: URI** is the complete namespace and class specification of the **URI** instance WWW.w3.org/Home/Lassila. Finally, **Documents: is the creator of** is the complete namespace specification of the predicate (tagged structural link in OPM terminology) 'is the creator of'.

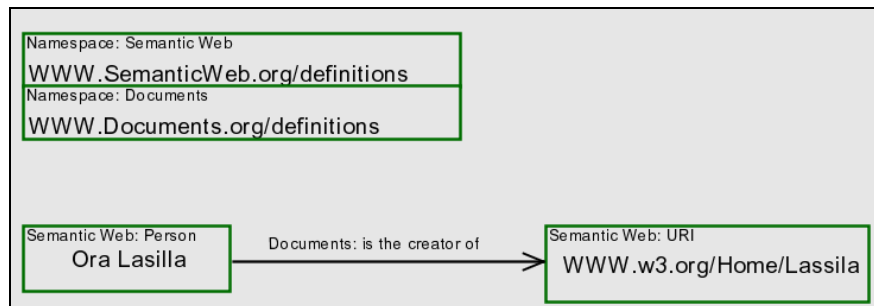


Figure 7. The OPD of Figure 5 with the namespace object boxes at the top left corner of the OPD.

The following two OPL namespace declaration sentences are the textual equivalents of the two namespace boxes stacked at the top left of Figure 7.

The namespace **Semantic Web** is at URL WWW.SemanticWeb.org/definitions.
 The namespace **Documents** is at URL WWW.Documents.org/definitions.

Just as namespace graphical specifications in an OPD are part of the graphical syntax of ViSWeb, OPL namespace declaration sentences are part of the textual syntax of ViSWeb. Based on the above two namespace declarations, the following class and relation definition sentences are:

The namespace **Semantic Web** defines the class **Person**.

The namespace **Semantic Web** defines the class **URL**.
The namespace **Documents** defines the relation '**is the creator of**'.

The Default Namespace Convention

Usually, most if not all the names in a single OPD are defined in the same namespace. Thus, it is redundant and cumbersome to specify separately for each name that it is defined within that namespace. A simplifying OPM *default namespace convention* is that the namespace at the top of the namespace stack in the OPD is the default namespace, so any class in the OPD which is defined within this default namespace does not require that the namespace name precedes it. In our example, the default namespace declaration sentence is:

The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.

Applying this default namespace convention, the OPL paragraph (collection of OPL sentences) that corresponds to the OPD in Figure 7 is:

The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.
The namespace **Documents** is at WWW.Documents.org/definitions.
The namespace **Documents** defines the relation '**is the creator of**'.
The **Person Ora Lassila is the creator of** the URI WWW.w3.org/Home/Lassila.

The XML script that specifies the analogous semantics, where the **Semantic Web** namespace is replaced by `rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"` and the **Documents** namespace is replaced by `s="http://description.org/schema/"` is the one listed in Table 2.

Wrapping the ViSWeb specification with XML and the DRF Equivalent

Having finalized the ViSWeb specification, we now describe how it is all wrapped with XML for being amenable to Web transfers and manipulations. We call this form XML/ViSWeb. We also describe how this XML/ViSWeb is translated into the XML/RDF standard. The following namespace declaration sentence, which is a constant part of any XML/ViSWeb text, specifies the OPM namespace:

The namespace **OPM** is at WWW.ObjectProcess.org/definitions.

The entire ViSWeb OPL specification is incorporated into the XML syntax by simply enclosing it within the `<OPM:OPL>` and `</OPM:OPL>` tags, as shown in Table 4. The OPD is likewise enclosed within the `<OPM:OPD>` and `</OPM:OPD>` tags. For human consumption, the actual graphic display of the OPD is presented in the XML/ViSWeb specification, as shown in Table 4. For machines, the actual OPD is replaced in the corresponding XML/RDF script by its XMI [11] representation. Comparing the human-oriented XML/ViSWeb specification to its corresponding machine-oriented XML/RDF translation in Table 4, the advantages for humans of the former over the latter are evident:

- Graphically, the machine-oriented XMI [28] specification of the ViSWeb OPD is rendered and displayed for humans as an intelligible diagram that contains the same information as the corresponding ViSWeb OPL script below it.
- Textually, the ViSWeb OPL script contains only sentences in a subset of natural English, which humans can read and understand with significantly less effort than required for performing "mental compilation." Such mental compilation is what humans are effectively required to execute when they encounter any XML/RDF script that they wish to interpret.

Table 4. Comparison between the complete human-oriented XML/ViSWeb specification of our example and its corresponding machine-oriented XML/RDF translation

Human-oriented XML/ViSWeb	<pre><?xml version="1.0"?> <OPM:OPD> Nmaespace: rdf WWW.w3.org/1999/02/22-rdf-syntax-ns# Namespace: Documents WWW.Documents.org/definitions Person Ora Lasilla Documents: is the creator of URI WWW.w3.org/Home/Lassila </OPM:OPD> <OPM:OPL> The namespace OPM is at WWW.ObjectProcess.org/definitions. The default namespace rdf is at WWW.w3.org/1999/02/22-rdf-syntax-ns#. The namespace Documents is at WWW.Documents.org/definitions. The namespace Documents defines the relation 'is the creator of'. The Person Ora Lasilla is the creator of the URI WWW.w3.org/Home/Lassila. </OPM:OPL></pre>
Machine-Oriented XML/RDF	<pre><?xml version="1.0"?> <OPM:OPD> -- Here comes the XMI [28] specification of the OPD, which enables its rendering shown above for human consumption. -- </OPM:OPD> <OPM:OPL> xmlns:OPM="http://www.ObjectProcess.org/Definitions" <rdf:RDF> xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# xmlns:Documents="http://documents.org/defintions"> <rdf:Description about="http://www.w3.org/Home/Lassila"> <Documents:'is the creator of'> Ora Lassila </Documents:'is the creator of'> </rdf:Description> </rdf:RDF> </OPM:OPL></pre>

The idea, then, is to show humans human-oriented XML/ViSWeb specification, exemplified by the top part of **Table 4**, while the machine will still be able to process its "pure" XML/RDF translation, shown at the bottom of **Table 4**. In order to do this, we must have a utility for bi-directional translation between ViSWeb and RDF.

7. Adding Attributes

Continuing with the example from [21], for specifications that are more complex, a compound resource can be created, as the following sentence and the corresponding graph in Figure 8 demonstrate:

"The individual referred to by employee id 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource <http://www.w3.org/Home/Lassila> was created by this individual."

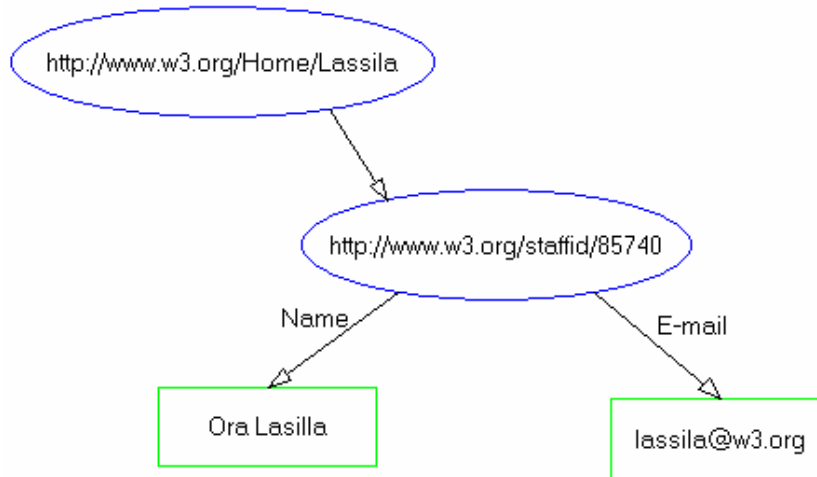


Figure 8. An identified property with structured value [21]

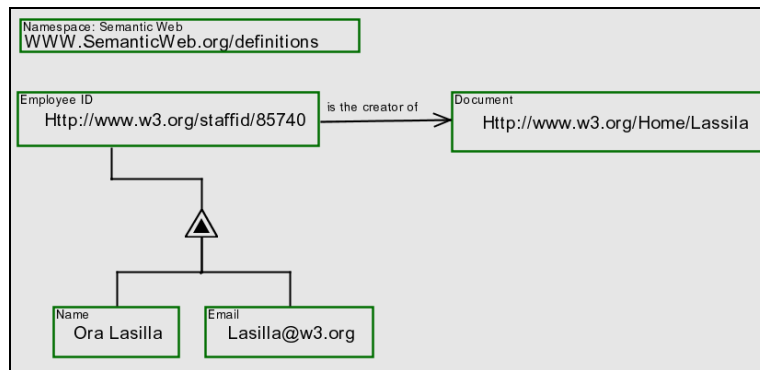


Figure 9. The OPD that corresponds to the graph in Figure 8

The OPL paragraph that corresponds to the OPD in Figure 9 is:

The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.
 The **Employee ID** WWW.w3.org/staffid/85740 **is the creator of** the **Document** WWW.w3.org/Home/Lassila.
 The **Employee ID** WWW.w3.org/staffid/85740 exhibits the **Name** **Ora Lasilla** and the **Email** Lasilla@w3.org.

The OPL reserved word exhibits expresses the exhibition-characterization relation (the relation between a class and its attributes, symbolized by a black-in-white triangle) from

The **Employee ID** <http://www.w3.org/staffid/85740> to the **Name Ora Lasilla** and to the **Email** Lasilla@w3.org.

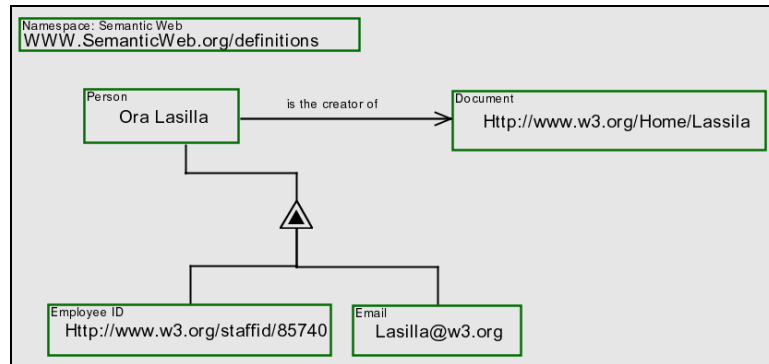


Figure 10. A better representation of the information presented in the OPD in Figure 9

A better representation of the information presented in the OPD in Figure 9 is shown in the OPD of Figure 10. The **Employee ID** is now an attribute of the **Person** rather than the other way around. That this is a better way of modeling is clearly seen when we compare the OPL paragraph below, which corresponds to the OPD in Figure 10, to the previous OPL paragraph, which corresponds to the OPD in Figure 9.

The default namespace **Semantic Web** is at WWW.SemanticWeb.org/definitions.
The **Person Ora Lasilla** is the creator of the **Document** WWW.w3.org/Home/Lassila.
The **Person Ora Lasilla** exhibits the **Employee ID** WWW.w3.org/staffid/85740 and the **Email** Lasilla@w3.org.

8. Advantages of the Visual Semantic Web Paradigm

The ViSWeb paradigm has a number of important advantages over present OWL/RDF/XML approaches, which are summarized in this section.

1. **Graphic-text knowledge representation:** The powerful graphic-text bimodal representation of OPM is extended to the Visual Semantic Web paradigm. Rather than having to mentally parse cryptic XML scripts, knowledge is presented to the user in a subset of natural language as well as diagrammatically. The two modalities complement each other, so if something is unclear in one representation, the other can be consulted for clarification. Using ViSWeb, one can ask for a translation from XML/RDF to XML/ViSWeb in order to get both visualization and a human-readable version of the XML syntax. The graphic representation can then be manipulated, changed, or augmented. Any such change would be reflected in the ViSWeb OPL script, and through it transparently back to the XML/RDF machine-oriented syntax. This way, working in a round-trip engineering mode, the human gets to think and develop ideas in a user-friendly environment without compromising the technical soundness of her/his work.

2. **Visual navigability:** In addition to the advantages of putting to work the "two sides of the human brain," the visual and the lingual, there are benefits that are unique to the Semantic web. The formal robust, yet intuitive, diagrammatic display enables users to surf and navigate the Web in a visual way in search for knowledge.

3. Semantic sentence interpretation: In spite of the aspiration of the Semantic Web, the basis of the RDF framework is syntactic rather than semantic: it draws on the concepts of *subject*, *predicate* and *object*, which are parts of speech used to analyze natural language sentences from a syntactic viewpoint. The same semantics can be expressed by inverse syntactic expressions. For example, without changing the semantics, we could easily switch the roles of subject and object in the example of Figure 1 by writing the sentence as "*The resource <http://www.w3.org/Home/Lassila> was created by Ora Lassila.*" Now the (syntactic) subject is <http://www.w3.org/Home/Lassila>, the object is *Ora Lassila*, and the predicate is "*was created by*". While the parts of speech are turned upside down and the predicate was changed from active to passive, the meaning of the sentence is still the same. The proposed OPM-based ViSWeb paradigm is based on a sound ontology of *objects* with *states* and *processes*: Objects are things that (at least potentially, and possibly at some state) exist, while processes are things that happen to objects and transform them (i.e., create or destroy them, or change their state). Based on this ontology, sentences can be interpreted semantically rather than syntactically. In OPM, each structural relation pair has a forward direction and a backward direction [14], so for example the forward relation "*is the creator of*" is paired with "*was created by*." This helps overcome syntactic differences and establish semantic equivalence.

4. Specification of system dynamics: Current work on the Semantic Web places emphasis on declaratively specifying structural knowledge, which relates to the static aspect of systems. Structural knowledge pertains to relations among objects that are not related to the objective of the system or the way it operates. According to Berners-Lee [3], "*the RDF model is basically an opening of the ER model to work on the Web.*" A typical ER model involves entity types, each with its set of relationships. "*The RDF model is the same, except that relationships are first class objects: they are identified by a URI, and so anyone can make one.*" This is a purely static world view, where everything can be expressed in terms of structural, time-independent relations. However, a major part of the knowledge about a system is functional (what is its purpose) and dynamic (how it operates). The current SW offers very little in this regard. Since OPM combines function, structure, and behavior in the same bimodal model, it provides a sound infrastructure for representing system dynamics and function in the ViSWeb model. While the details are beyond the scope of this work, suffice it to mention that knowledge about reactive and real-time systems requires treatment of events, conditions, actions, state transitions, and time exceptions, to name but a few major issues. All those and more can be modeled in OPM.

5. Complexity management: A major problem in real-life systems is their complexity due to the sheer amount of knowledge details. In addition to the OWL [32] set operators that can be translated into OPM as demonstrated above, OPM has built in abstraction-refinement mechanisms, including in-zooming and out-zooming, unfolding and folding, and state expression and suppression. These provide for building hierarchies of knowledge representation in general and over the Web in particular, enabling navigation up and down abstraction-refinement hierarchies.

9. Summary and Future Work

The Visual Semantic Web (ViSWeb) paradigm proposes to unify human and machine representations of knowledge. The foundation for this unification is Object-Process

Methodology (OPM), which advocates the integration of a system's structure and behavior is a single, graphic and textual model. The paper has presented the principles and outlined an implementation for the ViSWeb. Like OPM, the ViSWeb model enables the representation of static and dynamic knowledge using a combination of Object-Process Language (OPL), a subset of English, and Object-Process Diagrams (OPDs), an equivalent visual formalism. The advantages of this approach include graphic-text knowledge representation, visual navigability, semantic sentence interpretation, specification of system dynamics, and complexity management. As noted in [21], "*It is also important to understand that this XML syntax is only one possible syntax for RDF and that alternate ways to represent the same RDF data model may emerge.*" Indeed, this work presents an OPM-based approach to representing the Semantic Web on top of the RDF data model, which is expressed graphically, using OPDs, and textually in OPL, a subset of natural English which is also "machine understandable," i.e., amenable to parsing and converting back to the XML-based RDF syntax.

Future work will proceed in both the theoretical and practical paths. The theory will focus on extending the idea behind the ViSWeb paradigm and its initial specification, presented in this work, to cover other important knowledge and system representation aspects. Based on OPM, ViSWeb will be able to handle not only the declarative static structural aspects of knowledge, which is the focus of the current Semantic Web initiative, but also procedural, dynamic behavioral aspects, as well as functional ones. The practical work will augment the current capabilities of OPCAT so it will be suitable for modeling the various ViSWeb requirements presented here, and provide the services of bi-directional RDF-ViSWeb compilation. An even more ambitious goal is to design and build a Web crawler which will automatically generate ViSWeb representations of knowledge stored in Web pages. Accomplishing even some of these goals will greatly benefit the huge World Wide Web user community by providing them with a friendly semantic surfing tool and relieving them from the need to mentally compile XML scripts.

References

- [1] Anderson, J. R., Bower, G. H. Human associative memory. Winston and Sons, Washington, D.C., 1973.
- [2] Arnheim, R. Visual Thinking. University of California Press, Berkeley, California, 1969.
- [3] Berners-Lee, T. What the Semantic Web can represent, 1998.
<http://www.w3.org/DesignIssues/RDFnot.html>
- [4] Berners-Lee, T. and Hendler, J. Scientific publishing on the semantic web. Nature. 2001.
<http://www.nature.com/nature/debates/e-access/Articles/bernerslee.htm>
- [5] Berners-Lee, T. Hendler J. and Lassila, O. The Semantic Web. Scientific American, May 2001.
- [6] Brachman, R. On the epistemological status of semantic networks. In Associative Networks: Representation and Use of Knowledge by Computer. N.V. Findlee, Ed. Academic Press. New York, NY, pp. 3-50, 1979.
- [7] Bray, T., Hollander, D. and Layman, A. Namespaces in XML, World Wide Web Consortium Recommendation, 14 January, 1999. <http://www.w3.org/TR/REC-xml-names>
- [8] Brickley, D. and Guha R.V. RDF Vocabulary Description Language 1.0: RDF Schema, W3C work in progress draft, 30 April, 2002. <http://www.w3.org/TR/rdf-schema>
- [9] Chein, M., M.L.Mugnier. Conceptual Graphs: Fundamental Notions. Revue d'Intelligence Artificielle, vol.6,n.4,p.365-406,1992.
- [10] Corby, O., Dieng, R., and Hebert C. [A Conceptual Graph Model for W3C RDF](#) Proceedings of the Int. Conf. on Conceptual Structures (ICCS), 2000.
<http://www.int.gu.edu.au/kvo/reading/oliviericcs2000.pdf>
- [11] Cyc. OpenCyc, org, 2002. <http://www.opencyc.org/>
- [12] Delteil, A. Faron, C. A Graph-Based Knowledge Representation Language. Proc. 15th

- European Conference on Artificial Intelligence (ECAI) 2002. <http://www-sop.inria.fr/acacia/personnel/Alexandre.Delteil/ecai.pdf>
- [13] Delugach, H. Conceptual Graphs Homepage, 2003. <http://www.cs.uah.edu/~delugach/CG/>
- [14] Dori, D. [Object-Process Methodology - A Holistic Systems Paradigm](#), Springer Verlag, Berlin, Heidelberg, New York, 2002. www.ObjectProcess.org
- [15] Dori, D. [Why Significant Change in UML is Unlikely](#). Communications of the ACM, pp. 82-85, Nov. 2002.
- [16] Dori, D. Reinhartz-Berger, I. and Sturm, A. OPCAT – A Bimodal CASE Tool for Object-Process Based System Development. Proc. IEEE/ACM 5th International Conference on Enterprise Information Systems ([ICEIS 2003](#)), Angers, France, pp. 286-291, 2003. www.ObjectProcess.org
- [17] Dublin Core Metadata Initiative. 2002. <http://www.dublincore.org/>
- [18] Gaines, B.R. and Shaw, M.L.G. Concept maps as hypermedia components. International Journal of Human Computer Studies, 43(3), pp. 323-361, 1995.
- [19] Genesereth, M.R. Knowledge Interchange Format. Draft proposal American National Standard (dpANS), NCITS.T2/98-004. 1998. <http://logic.stanford.edu/kif/dpans.html>
- [20] Lassila, O. and Swick, R. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, 22 February, 1999. <http://www.w3.org/TR/REC-rdf-syntax>
- [21] Lehman, F. (Ed.) Semantic Networks in Artificial Intelligence. Pergamon Press, Oxford, UK, 1992.
- [22] Martin, P. and Eklund, P. Embedding knowledge in web documents: CGs versus XML metadata languages. Proc. 7th Int. Conf. on Conceptual Graphs (ICCS'99), Springer-Verlag, 1999.
- [23] Mayer, R.E. Multimedia Learning. Cambridge University Press, New York, NY, 2001.
- [24] McTear, M.F. (Ed.) Understanding Cognitive Science. Ellis Horwood, Chichester, UK, 1988.
- [25] Novak, J.D. A Theory of Education. Cornell University Press, Ithaca, Illinois, 1977.
- [26] Novak, J.D. and Gowin, D.B. Learning How to Learn. Cambridge University Press, New York, NY, 1984.
- [27] OMG UML1.4. Object Management Group, Unified Modeling Language Version 1.4, September 2001. <http://www.omg.org/technology/documents/formal/uml.htm>
- [28] OMG XMI. Object Management Group, XML Metadata Interchange (XMI) Specification Version 1.2, January 2002. <http://cgi.omg.org/docs/formal/02-01-01.pdf>
- [29] Peirce, C.S. Collected Papers of Charles Sanders Peirce. In Hartshorne, C. and Weiss, P. (Eds.), Harvard University Press, Cambridge, MA, 1932.
- [30] Peleg M. and Dori, D. [The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods](#). IEEE Transaction on Software Engineering, 26, 8, pp. 742-759, 2000.
- [31] Pepper, S. and Moore G. XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification, 2001. <http://www.topicmaps.org/xtm/1.0/>
- [32] Smith, M. K., McGuinness, D. Volz, R., and Welty, C. Web Ontology Language (OWL) Guide Version 1.0. W3C Working Draft, 4 November, 2002. <http://www.w3.org/TR/2002/WD-owl-guide-20021104/>
- [33] Sowa, J.F. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Reading, MA, 1984.
- [34] Sowa, J.F. Conceptual Graph Standard, 2000 http://users.bestweb.net/~sowa/cg/cgstandw.htm#Header_44
- [35] Sowa, J.F. The Common Logic Standard initiative. 2002 <http://suo.ieee.org/email/msg08241.html>
- [36] Sowa, J.F. (1999) Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, CA.
- [37] W3C RDF Validation Service, 2003. <http://www.w3.org/RDF/Validator/>

Interaction and navigation for a document database : a concrete case study

Isabelle Berrien, François Laburthe and Jean-David Ruvini

e-lab BOUYGUES SA
1, avenue Eugène Freyssinet,
78061 Saint Quentin en Yvelines, FRANCE
Mail : iberrien@bouygues.com
Tél : +33 1 30 60 53 66 Fax : +33 1 30 60 22 15

Summary

In this article, we present the application Wishbone, an innovative search engine to explore complex document database. Dedicated to the intranet document site of our society, we present in this paper how we managed to improve the request process by avoiding the “all or nothing” syndrome for a too fuzzy or too demanding request respectively, offering also the maximum of smoothness in the way the user formulates the query. We particularly focused our attention on the interaction between a web database and its users, either during a consultation or an interrogation process. This web context can be found in multiple environments like an intranet, a FAQs site, self-assistance, sale catalogues or document data bases. First we analyse document consultation sites in regard to quality criteria. In a second step, we describe our approach to furnish a software response to the problematic of settling, management and interaction of an online document database. All this work was accomplished using as example an enterprise document database with various documents like contracts, business notes and bug reports.

Keywords : document database, web mining, research engine, knowledge management

I. Introduction

Surfing in a content site appears to be one of the major Internet's problematics encountered these days with search engines. Therefore it has become one of the striking research axes about the World Wide Web, which not only has to face increasing amounts of pages but also is supposed to render the information they contain inside easy to read and understand. It is a huge problem since the multiple factors which are linked to it seem quite complex: different information sources, multiple publishers, site architecture, request systems... We therefore asked to ourselves the following question:

How can we help an internaut these days while searching information or investigating a data base using the internet?

Part of our research topics is concerned by this problem and this paper presented here discloses our first results.

In order to both answer the question and propose a concrete solution, we chose to investigate our research program in an enterprise intranet context. The data sources we got in hand could be therefore better identified. We can as a consequence narrow the field of our work to "Intranet Mining" (vs. "Web Mining"), which presents the following advantages. :

- **Intranet** : the problematics encountered in an intranet are quite identical to those of the world wide web but in a restricted context.

- **Data are easy to access** and smaller. It is a context well appropriate to conceive, develop and validate components which should be *in fine* exploited in the World Wide Web.

This article is made of three main parts. First we describe the problematic of document site we got interested in. In the second part, we explain in detail the issue we bring to it by studying the difficulties faced by the **C2S** enterprise about its document database and the last part presents our future work.

II. Document database consultation problematics

In order to clearly understand the problems encountered with the management of a document database and orientate our investigation properly, we first identify the main difficulties the user has to face when she connects to such a site. We briefly point out the main criteria which make a document site appealing, then we focus our attention to the complexity of such sites.

2.1. Quality measurement of a site

Rating a site quality notion is quite fuzzy and quite complex (aesthetic aspect, ergonomics content quality).

Nevertheless, our bad experience of frustrated internaut when not finding what we want makes us feel that a web site reaches its goal¹ when a user gets easily the advice or the information she's searching for.

This could be measured according to the following criteria:

Objective :

- percentage of times a user gets an answer
- number of clicks required to access the answer

Subjective/qualitative :

- easiness for the user to understand the database logic (does she get lost pretty early in the process or not ?)
- answer pertinence according to the user's expectation

We have now in hands some quantifiers to estimate the quality of a documentary database access. This will be our evaluation protocol for every site concerned.

2.2 Complexity factors

An information site must implement several different components like multiple choice forms with depending on the context, preceding selections, surfing access to side. This implies a continuous user interface during the process taking place between the site and the user. This process successively switches from questions, choices, fields of values presentation on one side and on the other side answers, selection,

¹ <http://www.auditweb.net/>

criteria. This also requires having a good capability of managing the process logic while the interface keeps on changing its content (apparition and disappearance of criteria, values changes...).

Le Grand has identified in her PhD. thesis three major difficulties:

- **Organization level of the documents:** in the user mind, the desired information takes usually part of a structured frame. Invoking a rich document organization does not refer directly to the general descriptors of the documents, but rather to the way the documents are related to each other. We are talking about specific descriptors depending on the document type which allow the information to be classified harmoniously in a hierarchy. The purpose here is to offer an intuitive access to the information.
- **The volume of information feed back:** quantity does not guarantee quality. A system that has plenty of data is not necessarily a good service. On the contrary, it often procures the user an uncomfortable feeling because of incapacity for her to get an overall view of the environment.
- **Richness of the documents content:** this time, we're really talking about the document itself. The more the document is annotated, classified, the more the final interaction is pertinent and smooth. This allows, moreover it implies, that the research data (*e.g.* documents descriptors) should be rich, multidimensional, giving a user the possibility to navigate or express her request with maximum of smoothness and comfort. Richness of content concerns the data production phases.

From those three points, we deduce that the route between weakly structured, heterogeneous data sources and its consultation on line can be long and complex. We can qualify this as an "information cycle", which equilibrium is fragile. Next we detail our solutions.

III. WISHBONE : a tool for the publishing and the consultation of documents.

The solution we propose, the Wishbone application [Berrien and Laburthe, 2003] is cut in three parts :

- **1. A research and exploration service** capable of displaying overall views of solutions if there are too many, or re-direct the user in case of no response. In summary, we prevent the user to fall into the “all-or-nothing” scenario which often forces her to click back during the browsing process (*e.g.* each time the request does not imply a small amount of responses).
- **2. A research model detached from the real data storing.** To each real document is associated a descriptor object. These objects are organized following a rich model and detailed for numerous possible types of documents. It has many advantages like a higher flexibility (for it allows data to be stored in a business model different from the research model used for the query process), and also enriched investigation techniques. The model permits to define a service structured like a product configuration component, completely disjoined from the document nature (classes/types/options/containers/associated objects...).
- **3. A help to the data production.** First of all, we must have in hand rich and complete data, even after a lazy publication process. We also need to prevent any storage of absurd information. Thinking about perfect automatic data storage does more look like a utopia, but it still should be feasible to narrow the amount of errors generated during the process. This could be done with the help of assistance mechanisms for document publishing.

Wishbone has been settled in an intranet frame dedicated to the management of documents published by a computer branch society of BOUYGUES Company (C2S) which covers all problematics mentioned upside. The C2S company is a hundred people staff society², with a flux of thousands documents published per year. These documents concern a large part of the company patrimony (checking flow of contracts, product description documents, marketing papers, technical points, internal procedures...).

In the following section, we describe each of the 3 preceding points.

² <http://www.c2s.fr/>

3.1. Research and exploration service: using views or relaxations to browse

As it was upper mentioned, a important criterion for a successful document database site is to be able to generate interrogation and response screens well adapted to the user (external or professional). We must keep, whatever the degree of advancement of the research process is, a friendly and clear interface whereas it could easily turn into a deep complex one if the information volume gets too big. Our key rule was to accompany the internaut by giving her views of documents groups instead of a huge list.

3.1.1 Too many answers

When answers are too many to be decently displayed and easily understood by the user, the Wishbone platform developed in our laboratory presents the solution set grouped in several packages. As example, the picture below shows a typical interface of our application (**Figure 1**).

The request in the case below is the following :

“General documents, written by external authors, of text format and published after april 4, 2002”.

We distinguish two parts, the form (up) and the responses (below):

The screenshot shows a search interface with the following sections:

- RECHERCHER UN DOCUMENT**: tous les champs sont optionnels. Search scope: spécifique à un client, général, indifférent.
- je cherche un document**: voir..., auteur interne, auteur externe.
- tous les auteurs**: voir..., format texte, format image, autres formats.
- types de format**: voir..., format texte, format image, autres formats.
- themes generaux**: voir...
- date de publication**: entre le 04 avril 2002 et le jour mois annee.
- Buttons: OK, RECOMMENCER.

VOTRE LISTE DE DOCUMENT GENERALS: Il y a 144 réponses à votre recherche. Ces réponses étant très nombreuses, nous vous invitons à affiner votre recherche. Vous pouvez vous laisser guider pour affiner votre critère de **date de parution**:

1. avant le 11/09/02	71 document general(s):	Responses
	- auteur : temporary authors - format : .doc ou .xls	
	- thème : ou Commercial ou Comite d'entreprise/Les comptes-rendus ou ou Architectures ou Tierce Maintenance ou Vie pratique ou Direction/CR C. Management/Archives	
2. apres le 11/09/02	73 document general(s):	Responses
	- auteur : temporary authors - format : .doc ou .xls	
	- thème : Developpement/Projets specifiques/Suivi ou Architectures ou Qualite/Procedures/Specifique DOP ou Ressources humaines/homes_themes/Publications HTML ou Relations clients/ProDiAL/Presentation	

Pour affiner sur le critère: [format](#), [thème](#), [auteur](#)

Vous pouvez aussi consulter toutes les réponses

- Figure 1 -

The query solution set contains 144 answers. Instead of forcing the user to refine her request in order to reduce the number of solutions, or displaying all solutions, the application returns an image of all answers classified into groups depending a chosen criterion.

The engine grouped them in 2 packages chronologically classified (published before and after September 11, 2002). We notice that each cluster has been enriched with common values for fields like the author, the format and the theme). This is the result of a segmentation performed by the engine upon all solutions (according to a criterion, either chosen by the engine as the most performing one from a clusterisation point of view (default behaviour), or either by the user if she wants another view).

Moreover, in each group, if all documents show one or more common characteristics, (for instance here, all documents published before September the 11th are xls or doc files), these are disclosed (**Figure 1**). This takes part of a enhanced converging process: the user is informed of the most relevant characteristics of each cluster, a sort of “one step in advance”, thus allowing her to choose the most appropriate one in the less hazardous way.

If the current filter criterion doesn't satisfy the user, she can switch to another one as she wants (here below the solution set was switched to a *format* point of view) (**Figure 2**):

VOTRE LISTE DE DOCUMENT GENERALS : Il y a **144** réponses à votre recherche

Ces réponses étant très nombreuses, nous vous invitons à affiner votre recherche.

Vous pouvez vous laisser guider pour affiner votre critère de **format** :

1. format : .doc	137 document general(s) <ul style="list-style-type: none">- auteur : temporary authors- format : .doc- thème : tous les themes proposes
2. format : .xls	7 document general(s) <ul style="list-style-type: none">- auteur : mvacquie ou Bureautique ou L.LASKÉ ou A Darnedru ou Idurand- format : .xls- thème : Qualite/Processus/Indicateurs ou Comite d'entreprise/Les comptes-rendus/CE ou Architectures/Support technique/CUBBE et CUPS

Pour affiner sur le critère: [auteur](#)
[thème](#)
[date de parution](#)

Vous pouvez aussi [consulter toutes les réponses](#)

- Figure 2 -

By clearly grouping the answers and attributing each group pertinent characteristics, while allowing the user to see the answers from the desired point of view, we obviously improve the user productivity in accelerating the converging process. Even if use of different views for the same data is a well-known method in databases, allowing access to numerous data using a monodimensional clustering process has never been reported before. This therefore brings a solution to the second problem mentioned in the beginning, that is to say how to manage a large amount of information in the most ergonomic manner.

3.1.2. No answer

In case of there is no solution to the request, instead of falling into the usual scenario of forcing the user to re-formulate her request, the engine proposes alternatives by displaying the closest solution spaces corresponding to the initial request. For instance, the request in the following example is “**find documents published by an external author and with a image format**”. This present request has an empty solution set, but still the user gets the following response (**Figure 3**):

RECHERCHER UN DOCUMENT : tous les champs sont optionnels

je cherche un document spécifique à un client général indifférent

tous les auteurs voir... auteur interne auteur externe

types de format voir... format texte format image autres formats

themes generaux voir...

date de publication entre le 04 avril 2002 et le jour mois année

(les commentaires entre parentheses sont les caracteristiques communes a vos solutions)

Il n'y a pas de solution satisfaisant tous ces choix mais nous vous proposons..

✔ choix respecté 🟡 choix proposé ✖ choix non respecté

✔ auteur : auteur externe	137 document general(s)
🟡 format : .doc	
✔ format : format image	3013 document general(s)
🟡 auteur : auteur ou departement non precise	

- Figure 3 -

This example shows that the two criteria (e.g. **external author** and **image format**) could not be both kept in order to get a response. The engine smoothly relaxed each of them until it gets valid solution spaces [Laburthe, 2002]. The user is then proposed 2 alternatives: the first one retains the external author request and looks in the closest values for the format field (**Word file instead of image**) and therefore can choose the “137 solutions” set. The second proposition kept the format value request (image) but had to extend the author field initial value (initially “external author”). We herein find out that quite a large part of the image documents has its author field value not filled!

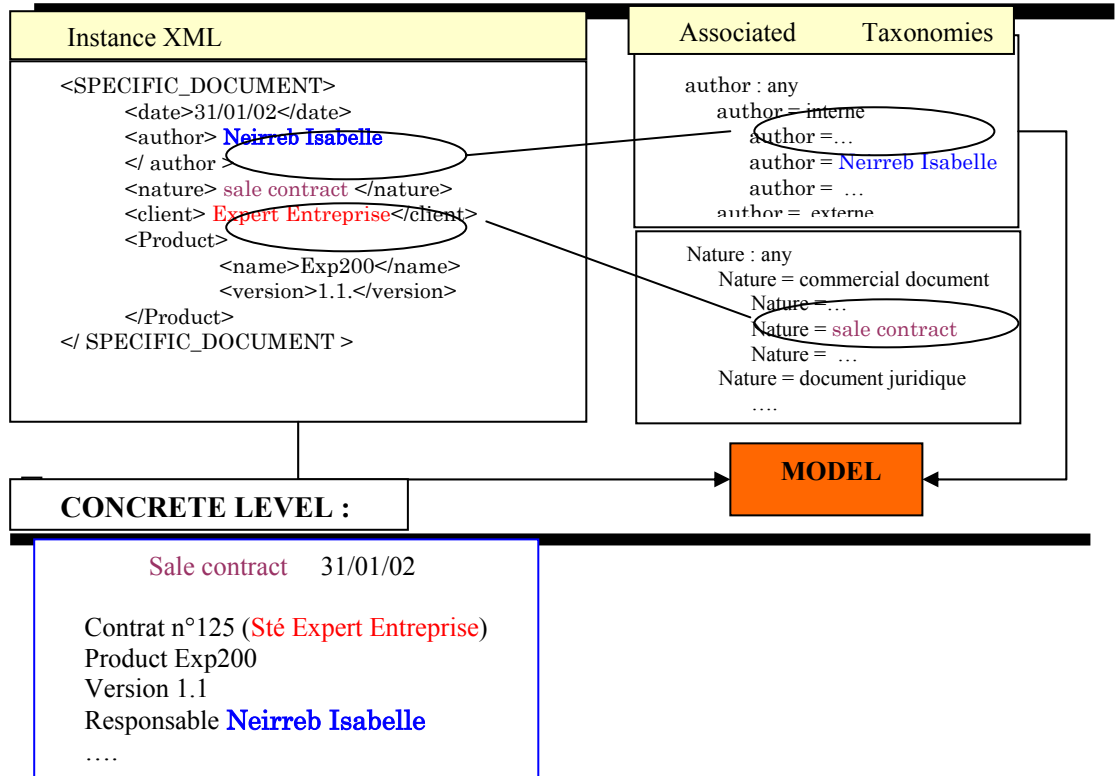
3.2 A research model

The application relies on a two level description (**Figure 4**).

The first level (the abstract one) concerns descriptive objects of documents. This is the level in which the research application acts and is responsible for the user dialog logic. We organize this level in concepts, classes and relations.

The second level (the concrete one) deals about the documents themselves, and is taken as reference for all the classes’ instances which constitute the whole object database.

ABSTRACT LEVEL :



- Figure 4 -

The **Figure 4** illustrates the organization abstract/concrete for a particular document (here a sale contract). This sale contract document is linked in the abstract level to an XML described instance (label on left part), of which certain elements take their values in an appropriate ontology [Gruver, 1993] according to the object model.

In this example, we have an instance of the SPECIFIC_DOCUMENT class. The SPECIFIC_DOCUMENT class in our model is a subclass of the DOCUMENT class and represents documents containing specific information like the reference to a client (a regular instance of the DOCUMENT does not). Here it refers to a client, “Expert Entreprise”, the author is Isabel Neirreb, and it’s got a reference to a product (PRODUCT class) which contains itself 2 slots: its name and its version.

At this point, this organization is similar to the one of Topic Maps³ [XTM Authoring Group, 2001]. In effect, in both cases we have a 2 layers separation, one

³ The Topic Maps concepts relies on:

being abstract and the other being concrete. The couple (“topic”, “association”) could be completely identified to our research model, and the “occurrences” to the document catalogues used by the application. Besides, as it is possible to type topics and associations (associations being also topics)⁴ within a Topic Maps model, this research model allows naming of the descriptive elements and link them to a type hierarchy.

As similarities are quite many, our model proposes nevertheless a deeper organization in the structure.

First of all, the model marks the objects, giving a quicker access to class instances. For example, if the request points to a commercial document which is considered in our model as a specific document, the application works directly on the instances of the `SPECIFIC_DOCUMENT` class. Therefore, it allows a straight access to sub-objects like the product. It also interacts with disjointed hierarchies of concepts: the model lets us define class slots pointing to graphs (associated taxonomies in Figure 1) which nodes are specific values of the given slot. In the upper example, we use different hierarchies of defined values (author, type document...). This gives the opportunity for each slot to be referenced in a specific domain of organized values. From a semantic point of view, this lets the application describe each class slot in the most real manner.

At last, in Topic Maps, whereas the links between occurrences require going through the abstract level in order to navigate, our research model allows a direct access within transversal pointers.

3.2.2. UML implementation

Even if our model has functionality similarities with Topic Maps, it mainly differs by our model which follows a UML implementation.

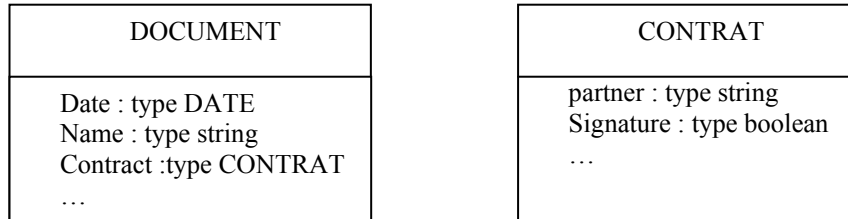
In the application we present here, data are encapsulated in a class hierarchy and are manipulated not as typed literals (cf. Topic Maps URI) but as objects. The user can therefore express his request upon more complex components, like for instance find all partners having dealt with a given contract.

-
- a *graph of topics* (computing representation of a subject) linked by
 - *associations* (relation between 2 or more concepts, where each concept is an actor of a predefined *role* defined by the association), each of this topic being instantiated by
 - *on or more occurrences* (a URI pointing to a resource (a Web page, a document part, a picture..)).

⁴ Marking an association in Topic Maps corresponds to naming fields in our model. Besides, occurrences in Topic Maps could be of different ranges. It could be pictures, numbers, text... The Topic Maps standard handles these distinctions through occurrence role concepts and types (being themselves topics). This is equivalent in our application to the range specification of a slot class .

This has been made possible with a document modelisation containing a DOCUMENT class which contains itself a complex type slot CONTRACT, which contains itself a slot pointing to the name of the partner (**Figure 6**).

Exemple :



- Figure 6 -

Multidimensionnal annotation

The object model we refer to is issued from a class hierarchy, each class of it having slots of different types :

Basic: *Boolean, string, num* and *date*

Extensible: *symbol*, from a set of predefined taxonomies, *object* (in a class hierarchy).

Our goal is to improve the precision and the fastness in the information retrieval by offering the user the opportunity to handle with coherence her own research process depending on different criteria. For instance, the user can choose to retrieve a document from a **publishing date** point of view, and, if she wants it, change filter and go further in the exploration by choosing the **author name** criteria.

Slots with values in a hierarchy of concepts

A large effort is made today to organize data. It appears that a classification of superior order should allow an easier retrieval of information. For instance, Yahoo⁵ performs classification by organizing links in sections. This engine, labelled as directory, shows undoubtedly a semantic dimension since it can classify responses in topics from the most general to the most specific ones. Another example is the search

⁵ <http://www.yahoo.com/>

engine Lycos⁶ which accomplishes research upon predefined categories like journeys or computers⁷.

Nevertheless, even if these approaches are close to our, they can't guarantee to return the most pertinent pages. Moreover, it is sometimes impossible to express a request giving a satisfying solution. For instance, taking Yahoo in order to find all information about Paris (as keyword input) in an educational context (for this, we click in the education area), we get answers like Paris airport, Paris-Match (a people magazine). The Sorbonne University, which is as matter of fact, an interesting result, only appears among a huge amount of inconsistent responses.

Semantic annotations

It is of course possible for a single object to be defined by numerous descriptive systems. It can fit in different classifications, a classification being considered in our case as a point of view referencing a domain of concepts. We propose here to define hierarchies of concepts and transform the document data into instances having slots with space values falling into these hierarchies.

In a related work, conceptual clustering approach has been studied [Chu *et al*, 1996], based on frequency and value distributions of data. But if this allows discovering high level concepts of numerical attribute values, it does not apply to semantic ones, and also is performed in a multidimensional approach.

The taxonomies files used by the application are references to hierarchies of symbols (concepts). They are trees in which each node is an identifier linked to the others by a specialisation relation. A taxonomy represents for a given slot the set of all values taken by the database objects for this slot, still with a deeper organization given by the graph. We distinguish several levels of details, which purpose is to allow a better instances organization and also to permit parent concepts inheritance. As matter of fact, performing a query about a construction site in Dunkerque (a French northern city) implies that it could be done just by refining documents issued from a request upon northern construction sites.

This hierarchical structure gives mainly during the request process an additional dimension to the one simply given by atomic values slots (see example §2.3.4).

Request example

Let's imagine a set of Topic Maps about distinct domains on which we would add a supplementary organization, letting us manipulate the values in a different manner than associations. A concrete request example is the retrieval of recent documents written by external partners of the **Expert Enterprise** society. We present below a

⁶ <http://www.lycos.com/>

⁷ search engines like « dogpile » (<http://www.dogpile.com/>) are beginning to appear (<http://mc42.free.fr/moteurs.htm>) which allow the user to express a query about specific document types (picture, MP3 files..)

part of the reference taxonomy file (*e.g.* the concepts hierarchy) for the slot “author” of the class DOCUMENT (**Figure 7**).

```
author = any
  author = external
    author = Society ZZ
      author = Dupont Albert
      author = Petit Roselyne
    author = Expert Enterprise
    author = UK Associates
  author = internal
    author = computer department
      author = Neirreb Isabelle
      author = Nent Arthur
      author = Uirvin JD
    ...
    author = jurist department
      author = Nerry Julie
      author = Oujat Franck
    ...
```

- Figure 7 -

All instances of the DOCUMENT class have an author slot with values pointing to this graph of concepts. Values like **Neirreb Isabel**, **Nent Arthur** or **Uirvin JD** could all be described by the concept **computer department**. This given hierarchy lets the user know that these authors are a part of the external authors, also that there are 3 different partners’ enterprises (Society ZZ, Expert Enterprise, UK Associates). This concepts organization let us also deduce that the topic **computer department** is closer to one of the three mentioned authors than **Roselyne Petit** for instance. We must assimilate the term “closer to” to “best approximation”.

All the solutions held inside the inheritance given by the tree, which are in our example the different departments in which the authors work.

Criteria combination

The search engine Excite⁸ for instance, like other numerous engines, order the solution sets depending on the combination of various criteria like the keywords frequency, links pointing to the pages,...These engines first of all sort the results according to a keywords filter (it is the only given field), the other criteria coming after. In order to get an adequate answer, we get two choices: either we remain patient and take a look of all the answers, or we give more than a single keyword in order the

⁸ <http://www.excite.com/>

precise the context. But this method drawback is that we can face a “no solution situation”⁹.

The model gives the possibility to handle rich data, and the opportunity to formulate request with many criteria is therefore accessible.

In response to the first difficulty invoked in the beginning of the article about the benefit of working with well described data, our model allows to handle a rich data organization. It lets the user express her request in a manner close from her wish and therefore get pertinent answers.

3.3. Production and consultation of data

From a new document publishing to the criteria required to capture it during the query process, we distinguish several steps:

3.3.1. Assisted publishing of documents (Saisame)

Wishbone has integrated dynamic forms for publishing or querying based on techniques analogous to those proposed by [Hermens et al, 1993].

The user, when publishing a new document, is proposed values for each field gradually while filling the form, that he can validate or not. These values are the result of prediction in extension of the most probable values for each slot. It is not only for facilitating the work but also to prevent from wrong publishing. For instance, if an author is in charge from its beginning of a contract coming made from the UK Associates Enterprise about the product **UKPro**, while filling the field corresponding to the name product, the service may propose the value “**UKPro**” (from statistically results issued from the database). Besides, the database content evolves in time and this assisted process can be view as a preventive procedure against absurdity [Sullivan, 2001]. For instance, we can mention the site Yahoo which hires at least 200 librarians only to ensure pertinence and quality of the exposed concepts [Russom, 2001].

3.3.2. Capture of request criteria

We describe below the help given to the user while searching for a document.

Model-assisted values capture

For fields pointing to a hierarchy of concepts, the application form only proposes values of the corresponding taxonomy. For instance, the taxonomy of authors only

⁹ Search engines like Google (<http://www.google.fr>) improve enhance the pertinence of hits by gathering the 2 approaches (directory + engine). For instance, Google shifts its request to Yahoo if it doesn't find a solution.

concerns the documents authors and the hierarchy of the document types only the field of document nature. For this, we settled a workshop for interface production. This workshop is used for generating periodically a synchronized interface in regard to the model. This allows integrity and also source data conformity of the form fields to be fully preserved. If a new document is published and its author (for instance **Mrs Roselyn Petit**) does not yet appear in the corresponding form field (her first published document then), an automatic refreshing process of the form taxonomies will induce the display of **Roselyn Petit** in the list of selectable authors.

Light capture

The form presents an interesting feature in that all fields are optional. In return, the query process seems smoother. The user can fill whatever she wants (from 0 to all fields) while knowing that whatever her request is, she will get a response allowing her to continue in the navigation process without having to go backwards.

Dynamic capture

Also, the form can display if wanted additional fields associated to the slots of a sub-object (of the query object). For instance, when searching for commercial documents (which instances are of class `SPECIFIC_DOCUMENT`), the form displays a part dedicated to the contract (which is a sub-object of the `SPECIFIC_DOCUMENT` class). This dynamic display renders the interface lighter (not on the screen if not needed), plus, it gives the user an opportunity to understand the base logic more quickly.

3.3.3. Integrity rules: model role

The model is very important in the sense that it fills several functions: the first and principal one is that it acts as a validation component for data. For instance, navigation using clusters is quite sensitive to isolated elements. If the database contains a document which was published in year 1299, the application will point out to this document by creating a set of clusters starting from the 13th century. Thus it is important to have a precise model of the admissible values for each slot. We have implemented a mechanism of constraint validation, similar to the XML Schemes [WWW Consortium, 2001] according to the following criteria:

- definition of a values space for each class slot
- implementation a default values
- required slots, types slots
- key-referenced objects (“idref” in XML descriptors like XLINK [W3C, 2001]).

The Wishbone engine loads the database and validates each object. The ones which don't verify constraints defined by the model are ignored and thrown. The use of constraints for allowed data values is a regular data modeling approach allowing complete data cleaning. Therefore, they are inaccessible by the research engine. This is called the **validation data phase**.

All these mechanisms (assistance, validation) have been implemented in order to make easier the maintenance and the publishing of coherent data. It gives a response to the document database management.

3.3.4. Scalability and performances

The Wishbone engine has been tested over a 50000 objects database within an average time response of less than a second per request.

Tests have been performed on a Pentium 4 Xeon (2.4 GHz), but charge could be distributed through a client-transparent in-between component enabling load balancing and queuing.

3. Conclusion and future work

Our approach of navigation into content improve the user experience not only from a qualitative point of view (quality tests in work), but also from a quantitative level (good appreciation from the persons in charge of the protocol in study). The generated interaction by the Wishbone application, being mostly inspired from human dialogs, settles an atmosphere of confidence with the user.

“You want all documents treated by UK Associates since 1998? No problem! We’ve got 7, the most recent ones, concerning the product UK200. Otherwise, we have 10 documents published between 1999 and 2001 and they’re all commercial contracts; and we’ve got 15 documents published before 1999 and all the authors are internal to the enterprise.”

The user disposes of a friendly interface (dynamic capture of criteria values, coherence field values, optional fields), which enhances the data accessibility through a high pertinence of the response display (views, relaxation).

This could be done with a light and safe maintenance circuit we call information cycle, with a major attention to apply an organized -intelligent like- frame to the data according to an appropriate business model.

Still, part of our future work will focus on

- The portability of the engine in regard of internet stored data. This means of course previous meta-data extraction but this complex process is not the purpose of the engine, rather takes place as second phase in the loop. Many related works like annotating data have already focused on that subject [Shet *et al*, 2002][Staab *et al*, 2001].
- Extension of clustering algorithms on semantic structures in order to improve the pertinence of clustering

References

[Le Grand, 2001] Le Grand, B. (2001). *Extraction et visualisation de systèmes complexes sémantiquement structurés*, Thèse de doctorat Université Paris 6.

[Sullivan, 2001] Sullivan, D (2001). *Five Principles of Intelligent Content Management* Intelligent Enterprise Magazine, August 31, 2001 – pp1-5

[Chu *et al*, 1996] Chu, W. W. ;Chiang, K. ; Hsu, C.-H. ; Yau, H. (1996). *An Error-based Conceptual Clustering Method for Providing Approximate Query Answers*, Communications of the ACM, 39,12es, pp 216-230.

[Staab *et al*, 2001] S. Staab, A. Maedche, and S. Handschuh (2001). *An annotation framework for the semantic web*, In Proceedings of the First Workshop on Multimedia Annotation, Tokyo, Japan, January 30-31, 2001,.

[Shet *et al*, 2002] Shet, A. *Relationships at the Heart of Semantic Web: Modeling, Discovering, Validating and Exploiting Complex Semantic*

Relationships, 29th Annual Conference on Current Trends in Theory and Practice of Informatics Nov. 24 -- Nov. 29, 2002

[Russom, 2001] Russom, P. (2002). *Managing Spaghetti Content* Intelligent Enterprise Magazine, May 28, 2002 – pp1-2

[Caseau and Laburthe, 2002] Caseau, Y. and Laburthe, F. Bouygues S.A. (2002) *Method for the data-driven adjustment of queries over a database and system for implementing the method*, European patent request 02290920.4

[XTM Authoring Group, 2001] TopicMaps.Org XTM Authoring Group (3 March 2001), *XTM: XTM Topic Maps (XTM) 1.0 : Topic Maps.Org Specification*.

[Gruber, 1993] Gruber, P.(1993) A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199-220, 1993

[Hermens et al, 1993] Hermens, L. A. and Schlimmer J. C. (1993). *A Machine-Learning Apprentice for the Completion of Repetitive Forms*, Proceedings of the 9th Conference on Artificial Intelligence for Applications ({CAIA'93}) pp164-170, IEEE Computer Society Press.

[Berrien and Laburthe, 2003] Berrien I. and Laburthe F. (2003). *Meilleures Interfaces entre services et utilisateurs*, JFT 2003 (accepted for publication).

[WWW Consortium, 2001] World Wide Web Consortium (2001), *XML Scheme Definition Language*, W3C Recommendation, 3 May 2001.

[W3C, 2001] W3C (2001). *XML Linking Language (XLink) Version 1.0* W3C Recommendation, 27 June 2001.