Using Semantics to Enhance Query Reformulation in Dynamic Distributed Environments

by

Damires Yluska Souza Fernandes

PhD. Thesis

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, APRIL / 2009

Universidade Federal de Pernambuco

Centro de Informática

Damires Yluska Souza Fernandes

Using Semantics to Enhance Query Reformulation in Dynamic Distributed Environments

Thesis presented to the Graduate Program in Computer Science of the Federal University of Pernambuco as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.) in Computer Science.

Advisor: Ana Carolina Salgado

Co-Advisor: Patricia Azevedo Tedesco

RECIFE, APRIL / 2009

Fernandes, Damires Yluska de Souza
Using semantics to enhance query reformulation in dynamic distributed environments / Damires Yluska de Souza Fernandes - Recife : O autor, 2009.
xiv, 172 folhas : il., fig., tab. graf.

Tese (doutorado) - Universidade Federal de Pernambuco. CIN - Ciência da Computação, 2009.

Inclui bibliografia e apêndice.

1. Banco de Dados. 2. Reformulação de consulta. 3. Semântica. 4. Ambientes distribuídos e dinâmicos. I. Título.

005.74 CDD (22.ed.) MEI 2009-075

Tese de Doutorado apresentada por **Damires Yluska de Souza Fernandes** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Using Semantics to Enhance Query Reformulation in Dynamic Distributed Environments"**, orientada pela **Profa. Ana Carolina Brandão Salgado** e aprovada pela Banca Examinadora formada pelos professores:

Fernando da Fonseca de Souza

Prof. Fernando da Fonseca de Souza
Centro de Informática / UFPE

[illegible]

Prof. Frederico Luiz Gonçalves de Freitas
Centro de Informática / UFPE

[illegible]

Prof. Carlos Alberto Heuser
Departamento de Informática Aplicada / UFRGS

Ana Maria de C. Moura

Profa. Ana Maria de Carvalho Moura
Instituto Militar de Engenharia - RJ

Bernadette Farias Lóscio

Profa. Bernadette Farias Lóscio
Departamento de Computação / UFC

Visto e permitida a impressão.
Recife, 28 de abril de 2009.

[illegible]

Prof. FRANCISCO DE ASSIS TENÓRIO DE CARVALHO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

To my parents, ***Edmilson*** and ***Lourdinha***,

To my husband, ***Fabio***,

To my children, ***Victor***, ***Amanda*** and ***Renan***.

Without your unconditional love,

support and sacrifice,

this thesis would not have been possible.

Deep Inside

By Taylor Duncan

Standing on the beach, sand between my toes
What lays in my future, who will come and go
The sun beams down upon me, as I raise my head and look
At the vast ocean before me, its size which I mistook
I feel so insignificant compared to this great expanse
What difference can I make, will I even be given a chance
I realize then while standing there, that all I have to do
Is listen to my heart and it will pull me through
For strength and inspiration are not material things
They come from deep inside of you they give your soul its wings
So whenever you're in doubt and you begin to stray
Take a look down deep inside and the answer will come your way
If you believe in yourself you can make your dreams come true
For no one else can do it, the power must come from you

ACKNOWLEDGEMENTS

This thesis would not have become a reality if I have not got the contribution of a lot of people. First of all, I thank my scientific advisors for their constant support, patience, suggestions and criticisms during my PhD years. In particular, I would like to thank Prof. Ana Carolina Salgado for receiving me as her student, and believe that I could accomplish such hard task. I have learned a lot with you, *Carol*, not only by the academic lessons, but also by your way of being a great and special woman. You are an example for me! In the same way, I have been increasingly grateful with the opportunity to be advised by Prof. Patricia Tedesco. In fact, I consider that she has been my co-advisor since the first day I came into her classes, and we started to discuss about agents and context. *Paxi*, I am very grateful for all your help, patience, and hearing me not only in good, but in bad times. You have an incredible computational view of everything!

I would also like to thank Professor Mokrane Bouzeghboud who received me in Versailles with special attention and hospitality. Thank you, Professor, our meetings in France were actually essential for my research. A deep and sincere special thanks to my third "informal" advisor, Prof. Zoubida Kedad, who influenced my work by providing valuable ideas in many interesting discussions. *Zoubida*, you are an example of researcher, and I feel very lucky for the opportunity to work with you!

I am very grateful to my thesis committee members, Prof. Carlos Heuser, Prof. Ana Maria Moura, Prof. Bernadette Lóscio, Prof. Fred Freitas and Prof. Fernando Fonseca, for the time they have spent in reviewing my thesis. I would like to say that Bernadette is a very special person for me. *Berna*, I enjoyed a lot being with you in France. You were more than a friend in good times and in bad times.

I thank all CIn professors who taught me, helped me or with whom I had the opportunity to learn and discuss computational issues: Prof. Flavia, Prof. Silvio Meira, Prof. Jones, Prof. Fernando, Prof. Nelson, Prof. Valeria, Prof. Fred and Prof. Liliane. Thanks for the support of Lilia and Neide. I also thank my friends from CEFET/PB who gave me support and incentive to my studies.

Friends born here have been a big part of my life and I am grateful for all of them: Vaninha, Rosalie, Marcia, Leonardo, Silvia, Ana Paula. Thank you for your friendship and support. I learned a lot with each one of you! I also thank my friends of SPEED group, namely Thiago Pacheco, Thiago Arruda, Rocir, Rodrigo, Victor and Carlos Eduardo. Special thanks to Pacheco and Arruda for their substantial contributions to my work and for the implementation of the tools. Also, special thanks to Carlos Eduardo with whom I have shared most of my PhD studies, doubts and discussions. *Carlos*, you are a great guy!

The greatest acknowledgement I reserve for my family, for their support, patience and love. My parents, *Edmilson* and *Lourdinha*, thank you for giving me the best start in life I could ever have. You were, are and will always be references in my life!!! I love you! My brothers and sisters, *Andre* and *Sandra*, *Ed* and *Erica*, *Adriano* and *Andrea*, you are all very special for me. Your encouragement made me stronger and persistent! Special thanks to *Andre* and *Sandra* for receiving me in their home, supporting my stay in Recife with so tender hospitality! I will always be grateful to you!! Also, I would like to thank my parents-in-law, *Herculano* and *Lucia*, for their support and incentive.

Most of all, I thank my children and my husband. Thank you very much for being part of my life! Thank you for your love, confidence and patience during all the times, including when I was absent. My dear *Victor*, *Amanda* and *Renan*, you three are the meaning of my life! I love you so much! My dear *Fabio*, thank you for being this special husband, thank you for your love, help, support and for not letting me down. I love you!

At last, but not least, thanks *God* for giving me health and emotional conditions to accomplish the hard task of conducting my PhD Studies.

ABSTRACT

Query answering has been addressed as a key issue in dynamic distributed environments. An important step in this process is reformulating a query posed at a peer into a new query expressed in terms of a target peer, considering existing correspondences between them. Traditional approaches usually aim at reformulating queries by means of equivalence correspondences. However, concepts from a source peer do not always have exact corresponding concepts in a target one, what may result in an empty reformulation and, possibly, no answer to users. In this case, if users define that it is relevant for them to receive semantically related answers, it may be better to produce an enriched query reformulation and, consequently, close answers than no answer at all.

In this work, we propose a semantic-based approach, named *SemRef*, which brings together both query enrichment and query reformulation techniques in order to provide users with a set of expanded answers. Exact and enriched query reformulations are produced as a means to obtain this set of answers. To this end, we make use of semantics which is mainly acquired from a set of semantic correspondences that extend the ones commonly found. Examples of such unusual correspondences are closeness and disjointness. Furthermore, we take into account the context of the user, of the query and of the environment as a way to enhance the overall process and to deal with information that can only be acquired on the fly.

We formalize our definitions using $\mathcal{ALC}$ Description Logics and present the algorithm underlying our approach with properties that guarantee its soundness and completeness. We implement the *SemRef* algorithm within a query submission and execution module for a Peer Data Management System (PDMS). We provide examples illustrating its usage and advantages. Finally, we present the experimentation we have done with *SemRef* and the obtained results.

Keywords: Query Reformulation, Semantics, Context, Dynamic Distributed Environments

RESUMO

O processamento de consultas tem sido abordado como um problema central em ambientes dinâmicos e distribuídos. O ponto crítico do processamento, no entanto, é a reformulação da consulta submetida em um ponto origem em termos de um ponto destino, considerando as correspondências existentes entre eles. Abordagens tradicionais, em geral, realizam a reformulação utilizando correspondências de equivalência. Entretanto, nem sempre conceitos de um ponto origem têm correspondentes equivalentes no ponto destino, o que pode gerar uma reformulação vazia e, possivelmente, nenhuma resposta para o usuário. Neste caso, se o usuário considera interessante receber respostas relacionadas, mesmo que não precisas, é melhor gerar uma reformulação adaptada ou enriquecida e, por consequência, respostas aproximadas, do que nenhuma.

Dentro deste escopo, o presente trabalho propõe um enfoque baseado em semântica, denominado *SemRef*, que visa integrar técnicas de enriquecimento e reformulação de consultas de forma a prover usuários com um conjunto de respostas expandidas. Reformulações exatas e enriquecidas são produzidas para permitir alcançar esse conjunto. Para tal, usamos semântica obtida principalmente de um conjunto de correspondências semânticas que estendem as normalmente encontradas na literatura. Exemplos de correspondências não usuais são *closeness* e *disjointness*. Além disso, usamos o contexto do usuário, da consulta e do ambiente como meio de favorecer o processo de reformulação e lidar com informações que somente são obtidas dinamicamente.

Formalizamos as definições propostas através da Lógica Descritiva $\mathcal{ALC}$ e apresentamos o algoritmo que compõe o enfoque proposto, garantindo, através de propriedades aferidas, sua corretude e completude. Desenvolvemos o algoritmo *SemRef* através de um módulo de submissão e execução de consultas em um Sistema de gerenciamento de dados em ambiente P2P (PDMS). Mostramos exemplos que illustram o funcionamento e as vantagens do trabalho desenvolvido. Por fim, apresentamos a experimentação realizada com os resultados que foram obtidos.

Palavras-chave: Reformulação de Consultas, Semântica, Contexto, Ambiente Dinâmico e Distribuído.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

'How do you know I'm mad?' said Alice.
'You must be,' said the Cat,
'or you wouldn't have come here.'
from "Alice's Adventures in Wonderland", Lewis Carroll

Introduction

The increasing use of computers and the development of communication infrastructures have led to a wide range of data sources being available through networks. As a consequence, there has been a demand for high-level integration of such autonomous and heterogeneous data sources through the development of diverse distributed environments, including Data Integration Systems [Halevy et al. 2006; Lóscio 2003], Peer Data Management Systems (PDMS) [Arenas et al. 2003; Herschel and Heese 2005; Sung et al. 2005] and DataSpaces [Franklin et al. 2005]. These dynamic distributed environments are characterized by an architecture constituted by various autonomous data sources (e.g., sites, files, databases), here referred to as *peers,* which hold information, and which are linked to other ones by means of mappings (i.e. associations between schema elements), called hereafter as *correspondences*. One special problem concerning these architectures is how to exploit the correspondences between schema elements in order to answer queries posed to one peer [Calvanese et al. 2004; Stuckenschmidt et al. 2005] in terms of a target peer and provide users with results in conformance with their preferences.

The existence of multiple different schemas describing related data is a common phenomenon in those distributed settings. Examples of scenarios which may benefit from such settings are scientific research ones such as biology, geography, health, education. In these scenarios, people have overlapping data, and they want to access other sources' additional information. In general, query answering in such environments is usually accomplished by the following steps: (i) query submission; (ii) query analysis; (iii) relevant data sources' identification; (iii) query reformulation; (iv) query execution; (v) answers integration; and (vi) query result presentation. Among such steps, query reformulation has been considered one of the most important, since it is concerned with the ability of translating the queries according to a set of inter-schema correspondences. Thus, when a user, at a given peer P, formulates a query posed over its schema, answers are computed at P, and the query is reformulated and forwarded to other peers through correspondence paths in the network. Since peers usually do not contain complete information to answer a given query, any relevant peer may add new and/or complementary answers. Furthermore, different paths of correspondences to the same peer may yield different answers.

As an illustration, consider three peers which belong to the "Education" knowledge domain, as depicted in Figure 1.1. In this scenario, peers have data about academic people and their works (e.g., cooperative research projects) from different institutions (A, B and C). In this light, it is possible that each peer only stores part of the information about cooperative research projects, and, even though they may have overlapping data for the same project, the content might still be different. It is very likely that a query posed in one of the given peers may obtain a more complete result considering such diverse and complementary data sources. Suppose now that a user at peer UnivA poses query Q_A over UnivA's schema. Answers are computed at UnivA, and the query is reformulated and forwarded to the other peers through the available correspondence paths. In this example, UnivA has a correspondence C_{A_B} to UnivB. Using C_{A_B}, a query Q_A will be reformulated to Q_B over UnivB's schema. Q_B will be executed at peer UnivB. In the same way, UnivB will reformulate Q_B to its neighbor peer UnivC (Q_C). At each reformulation process, the query is adapted to the current peer schema, according to its own constraints. Query results will be sent back to the peer UnivA after the query local executions. Peer UnivA will integrate all the results and present the complete set of answers to the user.

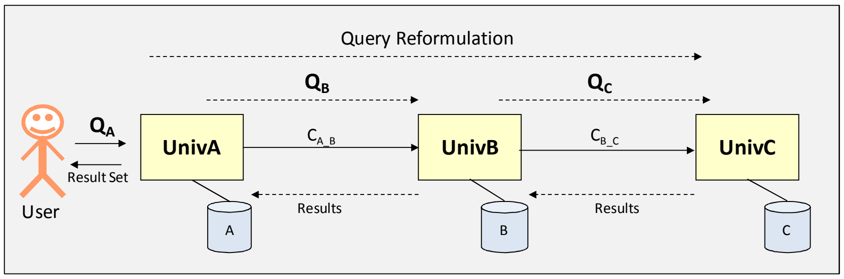


Figure 1.1 Query Answering in a Distributed Environment

In this sense, the crucial point we want to address is how to reformulate queries among the peers in such a way that the resulting set of answers expresses, as close as possible, what the users intended to obtain at query submission time, taking into account what kind of data the sources may contribute with and the dynamicity of the system.

Two aspects should be considered when dealing with query reformulation. First, querying distributed data sources should be useful for users, i.e., resulting query answers should be in conformance with users' preferences. On the other hand, it is not useful for users when they do not receive any answer at all. A second aspect is that concepts from a source peer do not always have exact corresponding concepts in a target one, which may result in an empty reformulation and, possibly, no answer to the user. Regarding the former aspect, we argue that user preferences and the current status of the environment should be taken into account at query reformulation time; regarding the latter, the original query should be adapted to bridge the gap between the two sets of concepts, using not only equivalence correspondences but also other ones that can approximate and/or enrich the queries.

In this perspective, we present a query reformulation approach, named *SemRef*, which uses semantics as a way to better deal with these mentioned aspects. In order to capture user preferences, query semantics and environmental parameters, we use contextual information [Dey 2001]. We accomplish query reformulation and adaptation by means of query enrichment. To this end, besides equivalence, we use other correspondences which go beyond the ones commonly found, namely: specialization, generalization, aggregation, disjointness and closeness. Through this set of semantic correspondences, we produce two different kinds of query reformulations:

i. an *exact* one, considering only equivalence correspondences; and
ii. an *enriched* one, resulting from the set of other correspondences.

The priority is producing the best query reformulation through equivalence correspondence, but if that is not possible, or if users define that it is relevant for them to receive semantically related answers, an enriched reformulation is also generated. As a result, users are provided with both exact and/or close answers, i.e., with a set of expanded answers according to their preferences.

The central questions we want to answer in this thesis are the following:

- What is the difference in producing query reformulations considering semantics and not considering semantics?
- To what extent does the use of semantics change the resulting set of query reformulations?
- In which situations could the use of semantics help to avoid an empty set of query reformulations?
- Is it possible to produce correct query reformulations, either exact or enriched, with the aid of semantics?

In order to answer these questions, we address the query reformulation problem in a setting containing just two peers, although our approach can also be used in an extended scenario composed by a set of diverse peers. In our work, we are not concerned with view-based query rewriting as works which deal with GAV/LAV strategies in order to reformulate queries posed through a global schema [Lóscio 2003]. Instead, we focus on reformulating a query posed at a source peer in terms of a target peer. We use ontologies as conceptual representations of peer schemas, and correspondences between these ontologies are identified to provide an understanding of their data sources. Figure 1.2 illustrates the central idea of our *SemRef* approach.

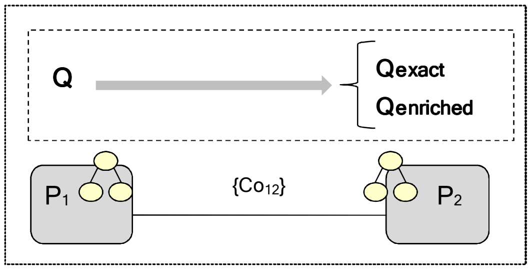


Figure 1.2 Query Reformulation Setting

Regarding this simplified setting, we define our problem as follows: given an ontology O_1 (at peer P_1), a user query Q expressed in terms of the concepts of O_1, a target ontology O_2, our goal is to find reformulated queries of Q expressed in terms of the concepts of O_2 in such a way that these reformulated queries not only include the best possible one (considering equivalence correspondences) but also the ones provided by other semantic correspondences between the ontologies. The reasons underlying that are twofold: (i) we aim to show that answers which are not an exact match, but which are a close match to the requirements specified in the query, can still serve the purpose of users, if they are in conformance with users' preferences; and (ii) we want to provide users with a set of expanded answers, in the light of the differences between the existing sets of concepts in the peers, and taking into account the context surrounding the query. This set of expanded answers will be obtained by executing exact and enriched reformulations.

In our work, we have conducted a set of experiments using as two main evaluation criteria the *degree of soundness* and the *degree of completeness* adapted from measurements commonly used in information retrieval systems (i.e., precision and recall) [Baeza-Yates and Ribeiro-Neto 1999]. Experimental results show that the use of semantics really improves both criteria.

Expected contributions of this research include:

i. The specification and implementation of an approach to identify the set of semantic correspondences between peer ontologies;
ii. The specification and implementation of a *context ontology* as a means to represent and store contextual information;
iii. The specification and implementation of the *SemRef* approach within a dynamic distributed environment;

The organization of this thesis is illustrated in Figure 1.3 and is described as follows.

Chapter 1 introduces and motivates the main ideas underlying this thesis. Furthermore, it outlines how it has been organized.

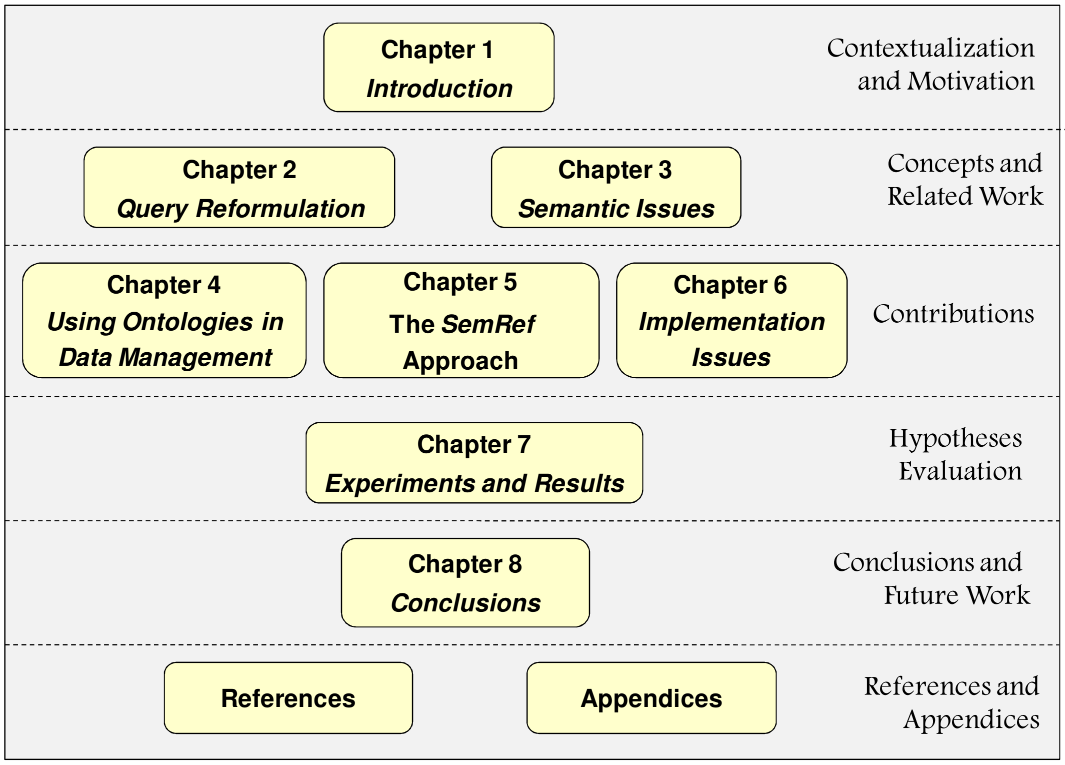


Figure 1.3 Thesis Organization

Chapter 2 reviews the theoretical foundations of query reformulation in distributed environments and presents how this problem has been considered in existing related approaches.

Chapter 3 explores some semantic issues, particularly, *ontologies*, *context* and the *Description Logics* formalism which have been increasingly used as a means for enhancing query answering in distributed environments.

Chapter 4 presents the way we use ontologies in our approach: (i) as background knowledge in order to identify semantic correspondences between matching ontologies; (ii) as a mechanism to represent and store contextual information and (iii) as a means for defining Ontology-based PDMS.

Chapter 5 presents the proposed *SemRef* approach by means of important definitions regarding the use of semantics and the algorithms underlying it.

Chapter 6 introduces the PDMS where our approach has been instantiated, provides details of the *SemRef*'s implementation and discusses the solutions we gave for bridging the gap between $\mathcal{ALC}$/DL and SPARQL semantics, thus providing users with queries in both languages.

Chapter 7 provides experiments of the proposed *SemRef* approach and the results that have been obtained.

Chapter 8 summarizes the proposed work by discussing the achieved contributions and indicating some directions in which the presented research could be extended.

Finally, used references are pointed out, and appendices are provided as follows: Appendix A shows *SemRef* complementary functions; Appendix B presents the ontologies concerning *Education* Knowledge domain we have used; Appendix C depicts additional screenshots of the query interface module; Appendix D shows ontologies regarding *Tourism* knowledge domain we have also used; and Appendix E presents the experiments we have performed.

CHAPTER 2

"If we knew what it was we were doing,
it would not be called research, would it?"
Albert Einstein

Query Reformulation

This chapter reviews the theoretical foundations of query reformulation in distributed environments, mainly focusing on techniques and approaches related to Databases and Data Integration environments. Our objective is to discuss the terminologies and techniques used in the subsequent chapters and to illustrate some of the techniques that are most relevant to our work. Furthermore, this chapter presents how query reformulation has been considered in existing related approaches and presents a comparison among them.

This chapter is organized as follows: Section 2.1 introduces query reformulation in Data Integration settings; Section 2.2 and 2.3 discusses query reformulation by expansion and personalization, respectively. Existing query reformulation approaches are described and compared in Section 2.4. Finally, Section 2.5 concludes the chapter with some considerations.

2.1. Query Reformulation in Data Integration

Data integration has been a research area in Computer Science for several years under diverse approaches: multi-database systems [Litwin et al. 1990], federated database systems [Sheth and Larson 1990], mediator-based systems [Wiederhold 1992], data warehouses [Chaudhuri and Dayal 1997], and, more recently, peer database management systems (PDMS) [Sung et al. 2005], dataspaces [Franklin et al. 2005] and pay as you go systems [Salles et al. 2007]. While these types of data integration systems differ with respect to their level of coupling or materialization, all of them have in common the need of dealing with heterogeneity, mappings and query answering. In this section, we focus on query reformulation in both mediator-based systems and PDMS. The reasons underlying that are twofold: (i) significant research effort has been already done towards mediator-based systems that query data sources through a mediated schema (a single central schema) [Lenzerini 2002, Lóscio 2003, Bilke 2007], and (ii) PDMS have received considerable attention because their underlying infrastructure (with no single central point) is appropriate for scalable and flexible distributed applications over the Web [Adjiman et al. 2007; Tatarinov and Halevy 2004].

2.1.1. Query Reformulation in Mediator-based Systems

Mediator-based systems attempt to provide users with a uniform interface to access and retrieve information from distributed data sources. The most important advantage of these systems is that they enable users to specify what they want without thinking about how to obtain the answers [Levy 1999].

A mediator-based system is responsible for reformulating, at runtime, a user query on a single mediated schema into a composition of sub-queries over the local source schemas [Lenzerini 2002]. To achieve this, mappings that capture the relationship between the local source descriptions and the mediator schema are required. Specifying these mappings is a fundamental step, since it influences both how difficult query reformulation will be and how easily new sources are added to or removed from the system. Mappings are usually described as declarative specifications of the data transformation between a source and the mediator.

Formally, a data integration system I is a triple <G, S, M> where [Lenzerini 2002] G is the global schema (structure and constraints), S is the source schema (structures and constraints), and M is the mapping between G and S, constituted by a set of assertions of the form $\{q_S, q_G\}$, in which q_S is a conjunctive query over the source schema, while q_G is a conjunctive query over the global schema.

To provide query reformulation, mappings are directional. This feature determines the query reformulation approach [Ullman 1997]. *Global-as-view* (GAV) systems describe mediator entities as views over the source schemas. To translate the user query, which is formulated in terms of the mediator schema, into one or several source queries, view expansion (or query unfolding) is used [Bilke 2007]. In this case, the mediator entities in the user query are replaced by their definitions in the mappings, resulting in a query containing only source relations. In *Local-as-View* (LAV) systems, the sources are described as views over the mediator [Halevy 2001]. There are also approaches which aim to combine GAV and LAV: GLAV and BAV. The former, named *Global-Local-As-View (GLAV)*, is a combination of answering queries using views followed by a query unfolding step [Madhavan and Halevy 2003]. A GLAV mapping is specified by a containment or an equivalence relationship between a conjunctive query over a source schema and a conjunctive query over a target schema. In the latter, named *Both-as-View (BAV)*, schemas are mapped to each other using a sequence of bidirectional schema transformations which are called transformation pathways [Mc. Brien and Poulovassilis 2006]. From these pathways it is possible to extract a definition of the global schema as a view over the local schemas (i.e., GAV), and it is also possible to extract definitions of the local schemas as views over the global schema (i.e., LAV). Next, we provide more details concerning GAV and LAV strategies, since they are the basis for all the approaches.

Query Reformulation in GAV Approach

When using GAV, the mapping M associates to each element g in G a query q_s (view) over S. The mapping M provides the way the system will be able to retrieve data related to each element from the mediated schema. This approach facilitates the query reformulation strategy, although it is considered effective only when the set of data sources are stable [Souza 2007; Bilke 2007].

Considering a data integration system I composed by relational data sources and a mediated schema which is a set of relations, we illustrate GAV query reformulation using the example depicted in Figure 2.1. This figure shows a single mediator relation R and two source relations S_1 and S_2. Arrows depict correspondences between attributes. To simplify the description, we use conjunctive queries to describe a mapping.

In this example, the GAV mapping would be:

$R(Name, Surname, Age, Salary)$:- $S_1(N, SN, A)$, $S_2(N, SN, S, D)$

which represents a join of S_1 and S_2 on the first and second attributes (N and SN), respectively. The schema of the mediator relation does not contain an attribute for the department D. User queries are formulated in terms of such mediator relation R.

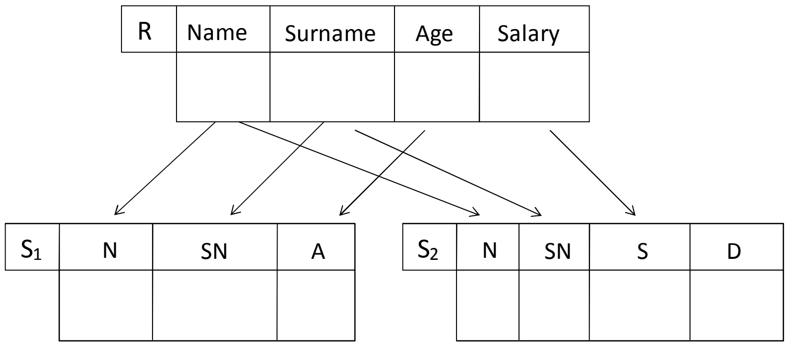


Figure 2.1 GAV Matching between sources and the mediator [adapted from Bilke 2007]

Thus, if the user asks for the salary of forty-year old people, the query can be formulated as follows:

$Q(S)$:- $R(Name, Surname, Age, Salary)$, $A = 40$.

The mediator will expand the view R with the mapping definition, resulting in the following query:

Q(S):- S_1(N, SN, A), S_2(N, SN, S, D), A = 40.

Based on that expanded query, the mediator produces a query plan, which describes how and in what order data sources are accessed.

Query Reformulation in LAV Approach

When considering the LAV approach, the mapping M associates to each element s of the source schema S a query q_G over G. In this case, each source s is characterized in terms of a view q_G over the mediated schema. This means that adding a new source implies only in adding a new assertion in the mapping. This makes the system's maintainability and extensibility easier [Souza 2007; Bilke 2007].

As an illustration, consider the same example described for GAV approach, now concerning LAV correspondences (Figure 2.2). For every data source relation S_1 and S_2, we write a mapping to the mediated schema relations R_1 and R_2, such as:

S_1(N, SN, A) :- R_1(Name, Surname, Age)

S_2(N, SN, S, D) :- R_1(Name, Surname, Salary), R_2(Name, Dept)

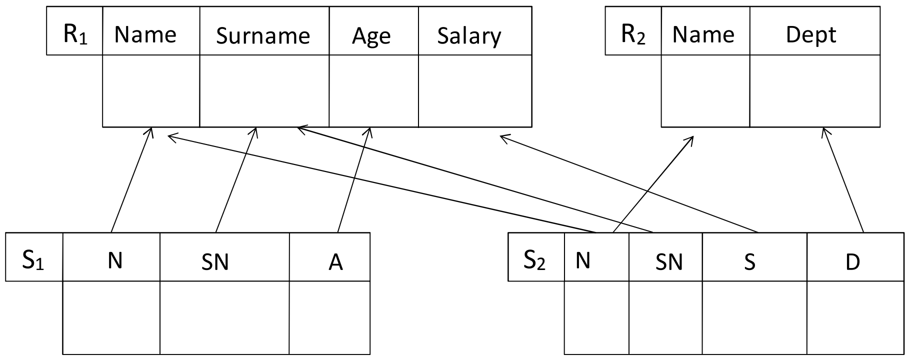


Figure 2.2 LAV Matching between sources and the mediator

Query reformulation in LAV is not as direct as in GAV. Because of the form of the LAV mapping descriptions, each of the sources can be viewed as containing an answer to a query over the mediated schema. A user query is also posed over the mediated schema. The problem is to find a way of answering the user query using only the answers to the queries describing the sources.

For instance, suppose that the user asks for people whose age is below fifty-years and which belong to dept = "Education", the query would be:

Q(Name, Surname, Age, Dept):- R_1(Name, Surname, Age, Salary), R_2(Name, Dept), A < 50, D = "Education".

The reformulated query on the sources would be:

$Q'(N, SN, A) :- S_1(N, SN, A), S_2(N, SN, S, D)$

In general, the complexity of answering queries using views is exponential, but there are some algorithms that work efficiently in practice, such as the method of *inverse rules* [Duschka and Genesereth 1997].

Some considerations

In a mediator-based data integration system, the need to establish the mediated schema (a central point) and mappings between the data sources and such mediator is a major bottleneck in integration efforts for real applications. In some cases, the data is so diverse that a mediated schema would be almost impossible to build or to agree upon, and very hard to maintain over time [Tatarinov and Halevy 2004]. PDMS have been considered as a natural extension of mediator-based integration systems [Herschel and Heese 2005], as some peers in such system may act as mediators to other peers. Next, we describe query reformulation in PDMS.

2.1.2. Query Reformulation in PDMS

Recently, Peer Data Management Systems (PDMS) came into the focus of research as a natural extension to distributed databases in the peer-to-peer (P2P) setting [Herschel and Heese 2005; Souza 2007]. While research on P2P file sharing considers very large networks (i.e., hundreds or thousands of peers), PDMS are usually conceivable on a smaller scale [Bilke 2007]. They are considered the result of blending the benefits of P2P networks, such as lack of a centralized authority, with the richer semantics of a database [Zhao 2006]. They can be extensively used for data exchanging, query answering and information sharing. For instance, in the areas of scientific research, the idea of setting up a PDMS for research in the related area to share data among peers has already been widely discussed [Ives et al. 2005; Zhao 2006; Ng et al. 2003].

A PDMS consists of a set of peers. Each peer has an associated schema that represents its domain of interest. However, PDMS do not consider a single global schema. Instead, each peer represents an autonomous data source and exports either its entire data schema or only a portion of it. Such schema, named *exported schema*, represents the data to be shared with the other peers of the system. Among those exported schemas, mappings, i.e., correspondences between schema elements are generated. In a PDMS, usually, schema matching techniques are used to establish such mappings which are themselves the basis for query reformulation. In general, queries submitted at a peer are answered with data residing at that peer and with data that is reached through mappings that are propagated over the network of peers. In this sense, we can define a PDMS as a set of peers $\{P_i\}_{i \in [1..N]}$. For each peer P_i, let S_i and M_i be respectively the exported schema of P_i and the set of correspondences stated at P_i between S_i elements and P_i's neighbors schemas elements.

Several network topologies have been proposed for P2P systems (as well as to PDMS), including: (i) pure unstructured [Lv et al. 2002], where peers establish connections to a fixed number of other peers, creating a random graph of P2P connections, and flooding is used to locate and retrieve shared content; (ii) pure structured topologies [Stoica et al. 2001] that use a Distributed Hash Table (DHT) to capture the relationship between content name and content location; (iii) super-peer topology [Yang and Garcia-Molina 2003], where special peers (super-peers) act as dedicated servers for other peers and can perform complex tasks such as query answering and data integration. Next, we provide an overview of query reformulation in two of these PDMS topologies: pure and super-peer.

Query Reformulation in Pure PDMS

In order to query a peer in a pure PDMS, its own schema is used for query formulation and mappings are used to reformulate the query over its immediate neighbors, then over their immediate neighbors, and so on. In other words, there is a flooding of reformulated queries over the network.

Figure 2.3 depicts a representation of a pure PDMS network in the light of *research centers*. In order to facilitate information exchange and query answering between different centers, four (illustrative) peers (with their respective data sources) are connected. The arrows indicate schema correspondences, which are used to reformulate the query over the peers' immediate neighbors, which themselves reformulate the query to their own neighbors, and so on. In this illustration, consider a user in Brazil that poses query Q_B based on his/her local schema. Q_B will first be reformulated to peer Portugal, according to the set of correspondences Co_{B-P}. Then such query will be reformulated to peers France and Germany, considering the correspondences between the source and target peers. After executing the queries, the answers are returned to the submission peer (Brazil) which is responsible for integrating all the answers. At end, the submission peer presents the final result to the user. This result will contain Brazil's own data in addition with data from the other three locations.

Query Reformulation in Super-Peer PDMS

In a super-peer PDMS [Yang and Garcia-Molina 2003], the system addresses query reformulation through its super-peers that have indexed information about their set of peers. Thus, query reformulation is broken into clusters of super-peers and peers that are able to answer the query.

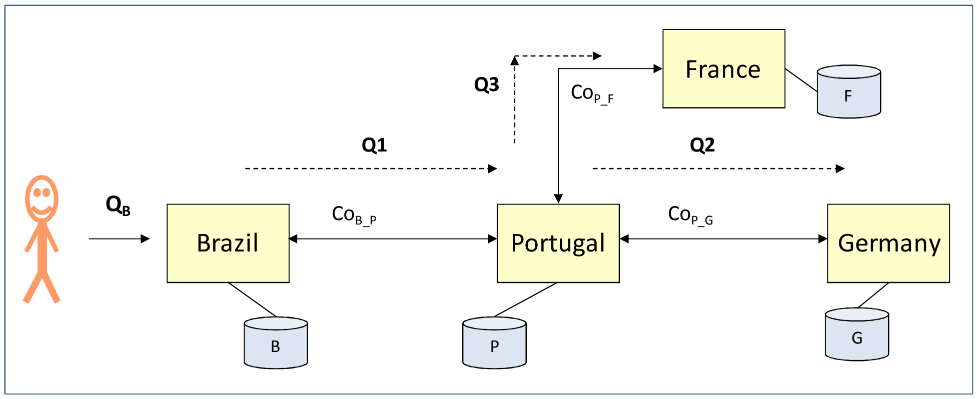


Figure 2.3 Query Reformulation in a Pure PDMS

For example (see Figure 2.4), assume that a user poses a query Q on Brazil's peer that is forwarded to its corresponding super-peer (SP_1). The super peer identifies peers in the cluster that are able to answer the query. Then, the super-peer reformulates the original query (Q) into a set of queries (Q_1 and Q_2) which will be sent to the peers (Portugal and Germany). It also reformulates Q into another query Q_3 which will be forwarded to another super-peer (SP_2). This super-peer reformulates such query into queries Q_4 and Q_5, sends them to their respective peers and carries out the integration of the returned answers. Finally, answers from every peer (neighbor super-peer and cluster peers) are returned to the original super-peer, where they are integrated to produce the final result. The final result is sent back to the Brazil's peer from which the query was submitted. The user at Brazil will get answers not only from its local data source but from other peers as well, similarly to pure PDMS.

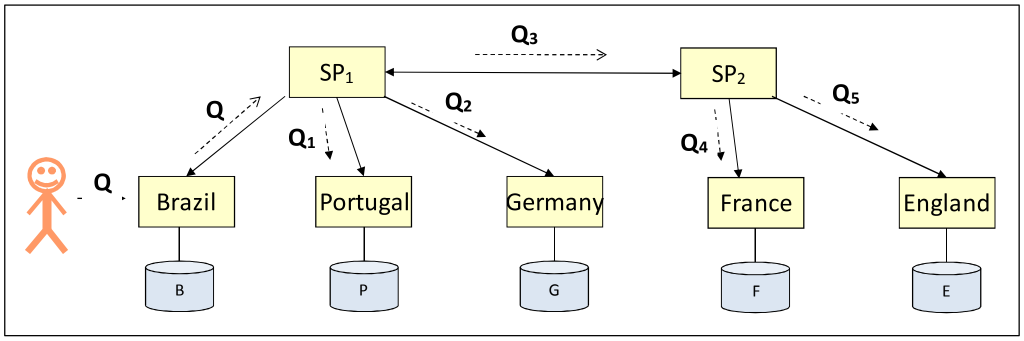


Figure 2.4 Query Reformulation in a Super Peer PDMS

Quality of Answers in PDMS

In distributed systems such as a PDMS, the quality of query answers depends not only on data quality of a particular local data source, but also on the quality of the correspondences [Yatskevich et al. 2006] among the peers. The semantics of a query can be distorted and/or data loss can happen if some correspondences are imperfect [Zaihrayeu 2006]. Thus, high quality level of query answering has been considered as the fact that data can flow among the databases preserving (at the best possible level of approximation) their soundness and completeness [Giunchiglia and Zaihrayeu 2002].

Particularly, there are (at least) three kinds of runtime factors, peculiar to PDMS, which influence the answer to a given user query, and, therefore, which also influence the quality of query answers [Giunchiglia and Zaihrayeu 2002; Zaihrayeu 2006]:

- Network (dependent) variance: peers may change the data of their sources, change their schemas, redefine correspondences, and new peers may join or leave the system at any time. Thus, the same query submitted to a given peer, but at *different times*, may yield different answers of different quality.
- Peer (dependent) variance: correspondences are established differently from one peer to any other peer. Therefore, the same query submitted at the same time but, by *different peers*, will result in different query propagation graphs. Consequently, the query results may be different and of different quality.
- Query (dependent) variance: *different queries* submitted to the same peer and at the same time may result in different query propagation graphs, and, thereby, may produce different results and of different quality.

Naumann [2001] shows that in large scale environments, users cannot expect correct and complete query answers, but they accept incomplete and partially incorrect answers. Indeed, a given user query may not need the best possible answer, but simply need some answer. Such kind of answer has been called *good-enough* [Zaihrayeu 2006]. The idea is that "an answer will be *good-enough* when it will serve its purposes given the amount of effort made in computing it" [Zaihrayeu 2006]. To this end, an answer does not absolutely need to satisfy all the constraints specified in the query. Currently, this notion is one of the main research lines of the EU FP6 project OpenKnowledge[1].

2.2. Query Reformulation by Expansion

In some query answering settings (e.g., Web search engines and Digital Libraries), queries, especially short queries (usually key-based ones), do not provide a complete specification of the information need. Many relevant terms can be absent from them and terms included may be ambiguous [Bai et al. 2007]. Also, they do not uniquely identify a single object in the data collection. Instead, several objects may match the query, perhaps with different degrees of relevancy [Baeza-Yates and Ribeiro-Neto 1999]. In these settings, query reformulation can be performed through the use of techniques like *query expansion*, which aim at increasing the likelihood of retrieved answers.

[1] http://www.openk.org/

Query expansion is a process which adds new terms or information to the query, implementing the query representation with information not directly explicit by the query [Grootjen and van der Weide 2006]. As an illustration, consider that a user searches for the term "Jaguar". In general, the system will not be able to disambiguate the term between "Jaguar, the car brand" from "Jaguar, the animal". Nevertheless, if the system has some additional knowledge classifying "Jaguar" as a kind of animal, the system will be able to apply query expansion and return answers with the correct disambiguation by adding this term. Thus the query could be "Jaguar animal" instead of only "Jaguar". In this technique, the idea is to treat the query as an initial attempt to retrieve information and use it to construct a new query [Grootjen and van der Weide 2006].

The purpose of query expansion is to reduce the gap between user intention when formulating a query and system interpretations of such query. It can be both interactive and automatic, and can be done taking into account knowledge stored in a *thesaurus* or in an *ontology*. For instance, considering an ontology where there are relationships between two single terms such as $t_1 \rightarrow t_2$, if a query contains term t_1, then t_2 is always considered as a candidate for query expansion.

The efficiency of a process as query expansion may be measured in terms of recall and precision. Recall is the ratio between the number of relevant documents retrieved to the total number of relevant documents, and precision is the ratio between the number of relevant documents retrieved to the total number of retrieved documents [Manning et al. 2008]. Expanding a query with synonym terms is known to improve the recall. To improve precision by reducing the ambiguity of ordinary terms, the use of taxonomies, classification schemas or ontologies is recommended [Styltsvig 2006].

2.3. Query Reformulation by Personalization

The goal of *query personalization* is to assist users when formulating queries in order to enable them to receive relevant information, according to their intentions. The relevance of the information is defined by a set of criteria and preferences specific to each user. These criteria may describe the users' domain of interest, the quality level of the data they are looking for or the modalities of the presentation of the data [Kostadinov 2007]. In this light, query personalization may be defined as the process of dynamically enhancing a query with related user preferences stored usually in a user profile with the aim of providing personalized answers [Koutrika and Ioannidis 2005].

The underlying idea of query personalization is that different users may obtain diverse answers which are considered as relevant due to what is identified from the user model. In some areas (e.g., Information Retrieval), query personalization is considered a machine learning process based on some kind of user feedback [Kostadinov 2007]. In other areas, such as Human-Computer Interaction, user profiles generally define user expertise with respect to the application domain to provide them with appropriate interfaces and dialogs [Eisenstein and Puerta 2000]. Thus, it is essential to take into account the *user model*, including the *user context*. The *user context* (e.g., location and preferences) can be exploited by the system either to answer queries or to provide recommendations, so users at different locations may expect different results, even from a same query. Among these elements, *user preferences*, whatever they concern system adaptation or content delivery, are the main knowledge which characterizes a personalization system whose target is to increase user satisfaction [Kostadinov 2007].

Considering the user model and a query composed by concepts, we can say that it is likely that a concept y is relevant for the query if we know that a concept x is in the user model (e.g., in the user profile), and a binary relationship r(x,y) holds. In addition, some kind of quantitative or qualitative attribution can be stated for these relationships. As an illustration, consider a database schema (in relational model) with information about restaurants depicted in Table 2.1. Users have preferences about types of restaurants that they express by providing a numerical score between 0 and 1 that quantifies their degree of interest. When formulating queries about restaurants, the system will take into account these preferences to provide more meaningful answers. Thereby, a query "Select Name From Restaurant Where Region like 'Tambau';" will take into account not only the required data expressed in the query, but also the preferences concerning the types of restaurants which exist in the asked region.

Table 2.1 Database for "Restaurants"

Rid	Name	Region	Cuisine Type
132	Bella Mamma	Manaíra	Italian
111	Lion	Tambaú	French
654	Boulange Bistro	Tambaú	French
077	Casa do Bacalhau	Manaíra	Portuguese
234	Indian Arr	Tambaú	Indian
123	La Espanhola	Tambaú	Spanish
537	Nostra Casa	Tambaú	Italian
098	Adega do Alfredo	Tambaú	Portuguese
055	Sapore d'Itália	Tambaú	Italian
564	La Isla	Cabo Branco	Spanish

Assuming a set of preferences (Table 2.2) stated by the user, and a threshold equal to 0.5 (this implies that only the preference scores which are equal or higher to 0.5 will be considered), the answer to the mentioned query is shown in Table 2.3. This set may be a subset of the answers to the original query, but it is supposed to contain interesting answers with respect to users' preferences.

Table 2.2 Preferences for Types of Cuisines

Cuisine Type	Interest Score
French	0.5
German	0.8
Greek	0.4
Indian	0.45
Italian	0.9
Mexican	0.6
Portuguese	0.55
Spanish	0.3

Table 2.3 Answers Set

Answers
Nostra Casa
Sapore D'Itália
Adega do Alfredo
Lion
Boulange Bistro

2.4. Existing Query Reformulation Approaches

Query reformulation techniques have been addressed in different computational environments. In the following, we review existing related approaches and provide an overall comparison among them.

2.4.1. Query Reformulation in a Single Database Using an Ontology

Necib and Freytag [Necib and Freytag 2004; Necib and Freytag 2005; Necib 2007] have presented an approach for query reformulation within single relational databases using ontology knowledge. They use ontologies to transform a user query into another query that may provide a more meaningful answer to the user. To the authors, "meaningful answer" is one that is more complete than the initial one w.r.t. (with respect to) user's intension. To this end, they define and specify different mappings that relate concepts of an ontology with those of an underlying database. In addition, they propose a set of semantic rules for transforming queries using terms derived from this ontology. The rewriting rules are classified according to the result of the query transformation, as follows:

- Extension rules: aim at extending the query answer with results that meet user's expectations. The rules are grouped into six categories: *Synonymy* rule, *Collection* rule, *Part-Whole* rules, *Support* rules, *Feature* rules, and *Consistency* rules.

- Reduction rules: aim at reducing the number of irrelevant tuples from the query answer. In this case, the *sensitivity* rule is intended to provide answers which contain as few as possible false positives.

As an illustration, we describe the *part-whole* rule. The basic idea of this rule is the use of *part-whole* properties to discover new database objects (parts or wholes of a concept), which are closely related to a given user query.

Assuming that we have an ontology O_1 and a relational database DB_1. O_1 describes product concepts (Figure 2.5). DB_1 contains information about technical items of a store and includes two relations called *Article* and *Component*, as follows:

Article: A-ID (PK), Name, Model, Price

Components: S-ID (PK and FK to Article), M-ID (PK and FK to Article)

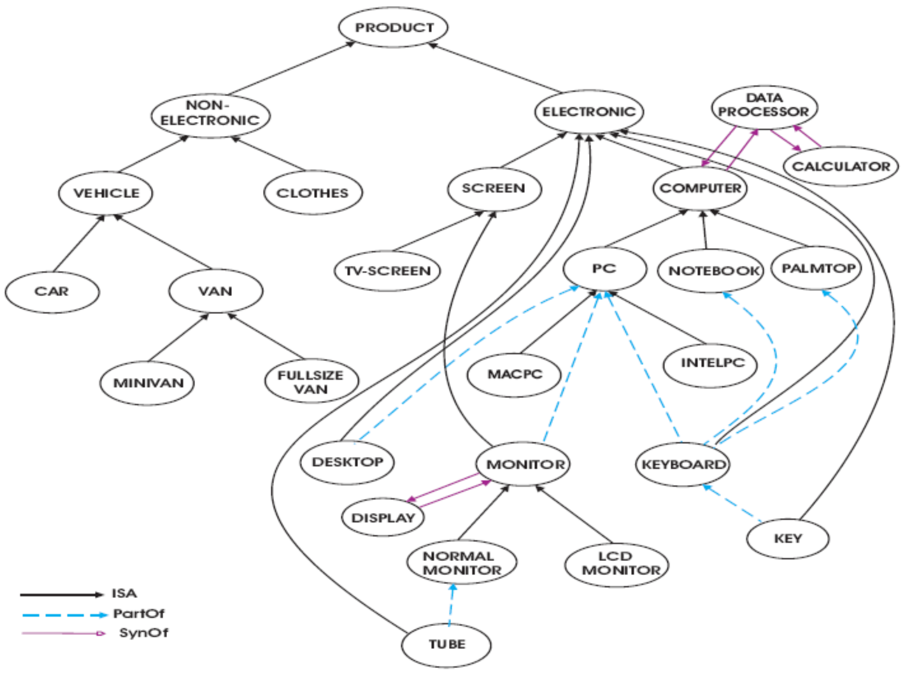


Figure 2.5 Product Ontology

Suppose that DB_1 contains the instances shown in Tables 2.4 and 2.5. To extract ontology semantics related to the content of the associated database, mappings between the ontology and the database are established. If a user wants to retrieve information about the article pc from the database DB_1, his submitted query may be stated as: $Q = \{(x1, x2, x3, x4) \mid (x1, x2, x3, x4) \in \text{Article} \wedge x2 = \text{"pc"}\}$.

Table 2.4: Article

A-ID	Name	Model	Price
123	Computer	IBM	3000
124	Intel-PC	Toshiba	5000
125	Notebook	Dell	4000
127	PC	Compaq	2500
128	Product	HP	3000
129	Monitor	ELSA	1000
135	Keyboard	ITT	80
136	Desktop	IBM	1000
140	MacPC	Mac	2000
141	Calculator	Siemens	1500

Table 2.5: Components

S-ID	M-ID
123	129
123	135
123	136
124	129
124	135
125	135
127	129
127	135
127	136
128	129
128	135
128	136
140	129
140	135
140	136
141	135

Analyzing O_1, it may be deduced that a pc is composed out of three parts: a desktop, a monitor and a keyboard. Assuming that all PC-objects are composed exactly out of these parts, which do not participate in the composition of any other object, enables the identification of PCs by means of their components. Thus, the set of terms {desktop, monitor, keyboard} and the term pc are considered semantically equivalent. By applying this rule to the query Q we get the following reformulated query:

$$Q' = \{(a_1, a_2, a_3, a_4) \mid (a_1, a_2, a_3, a_4) \in \text{Article} \wedge a_2 = \text{"pc"}\} \cup \{(a_1, a_2, a_3, a_4) \mid (a_1, a_2, a_3, a_4) \in \text{Article} \wedge [\exists\, y_1, y_2 \mid (y_1, y_2) \in \text{Component} \wedge a_1 = y_1 \wedge \exists\, (b_1, b_2, b_3, b_4) \in \text{Article} \wedge (y_2 = b_1 \wedge b_2 = \text{"monitor"})] \wedge [\exists\, z_1, z_2 \mid (z_1, z_2) \in \text{Component} \wedge a_1 = z_1 \wedge \exists\, (c_1, c_2, c_3, c_4) \in \text{Article} \wedge (z_2 = c_1 \wedge c_2 = \text{"keyboard"})] \wedge [\exists\, u_1, u_2 \mid (u_1, u_2) \in \text{Component} \wedge a_1 = u_1 \wedge [\exists\, d_1, d_2, d_3, d_4 \mid (d_1, d_2, d_3, d_4) \in \text{Article} \wedge u_2 = d_1 \wedge d_2 = \text{"desktop"})]\}$$

2.4.2. Query Reformulation using Profile Knowledge

The work of Kostadinov [2007] is concerned with two issues: (i) data personalization and (ii) query reformulation using profile knowledge. To deal with data personalization, Kostadinov proposes three metamodels: (i) a *profile metamodel* which is composed by five dimensions (domain of interest, personal data, quality data, security data and data delivery); (ii) a *context metamodel* which concerns information about the environment (location, time) and the interactions between users and the system and; (iii) a *preference metamodel* that organizes the user preferences and depends on the previous ones.

In his work, the problem of query reformulation has been addressed in mediator-based systems. Since personalization may occur in every step of a query life cycle, the work studies two query reformulation approaches based on algorithms for query enrichment and query rewriting, and proposes a new query reformulation approach. The idea is to introduce data personalization by performing query enrichment based on user profile and preferences, i.e., integrating elements of the user profile into the user's query (expressed on the virtual schema) so that it can be evaluated on the data sources.

The proposed query reformulation process is shown in Figure 2.6. It is composed by the following steps: query expansion, relevant data sources identification, relevant data sources combination and final enrichment. More precisely, the work deals with the first and the third steps. The second is done using the MiniCon Algorithm [Halevy and Pottinger 2001] and the fourth, using the Koutrika and Ioannidis algorithm [Koutrika and Ioannidis 2005].

Regarding the query expansion, the algorithm works as follows. Consider S_V a virtual schema, P_U an user profile interpretation over a query Q_U that has been submitted over S_V. To identify the meaningful virtual relations, the algorithm follows four steps: (i) select the virtual relations that are linked to user profile predicates; (ii) calculate the semantic distance between the selected virtual relations and the query; (iii) measure the contribution of the new predicates to the given query and; (iv) choose the virtual relations to be added for the query expansion.

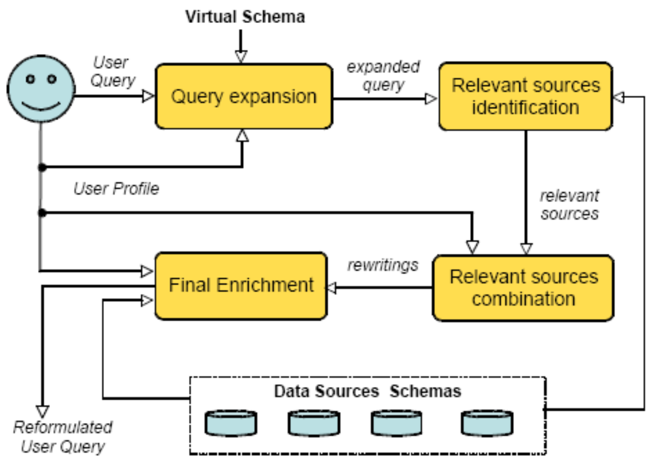


Figure 2.6 Kostadinov's Query Reformulation Process [Kostadinov 2007]

2.4.3. Query Reformulation in Piazza

Piazza [Halevy et al. 2005] is a PDMS where a formalism, named PPL (*Peer-Programing Language*, pronounced "People"), is defined for mediating between peer schemas. This formalism uses the GAV and LAV approaches to specify mappings. The semantics of query answering is specified by a query reformulation algorithm for PPL.

For the sake of simplicity, the authors assume that the peers employ the relational model, although in the implemented system, they share XML files and pose queries in a subset of XQuery[2]. Also, they assume that each peer defines its own

[2] http://www.w3.org/TR/xquery/

relational peer schema whose relations are called *peer relations*, so queries will be posed over these relations. Peers may also contribute with data to the system in the form of *stored relations* (analogous to data sources in a data integration system). As a result, queries will be reformulated in terms of stored relations. The set of stored relations is referred as the peer's stored schema. Metadata is stored in the Piazza catalog which is assumed to be acessible to all existing peers in the system.

There are two types of mappings: (i) *Storage descriptions*: describing the data within the stored relations (generally with respect to one or more peer relations), and (ii) *Peer Mappings*: between the schemas of the peers. In this sense, each peer contains a (possibly empty) set of *storage descriptions* that specify which data it actually stores by relating its stored relations to one or more peer relations. Formally, a storage description is of the form A: R = Q, where Q is a query over the schema of peer A and R is a stored relation at a peer. This description specifies that A stores in relation R the result of the query Q over its schema. PPL also provides storage descriptions of the form A: R $\subseteq$ Q. As a result, storage descriptions may be both *containment* (inclusion) or *equality* storage descriptions. For example, consider a storage description that relates the stored *students* relation at a peer UPenn to the peer relations:

$$\begin{aligned}\text{UPenn: students(Sid, name, advisor)} \subseteq\ & \text{UPenn: Student(Sid, name, \_)},\\ & \text{UPenn: Advisor(Sid, fid)},\\ & \text{UPenn: Faculty(fid, advisor, \_,\_)}\end{aligned}$$

This storage description says that UPenn: students stores a subset of the join of Student, Advisor and Faculty.

Peer mappings provide semantic links between the schemas of different peers and are classified into two types – *inclusion/equality mappings*, and *definitional* mappings. The first ones are defined in a similar way to the concepts of storage descriptions and are of the form $Q_1(A_1) = Q_2(A_2)$ or $Q_1(A_1) \subseteq Q_2(A_2)$, where Q_1 and Q_2 are conjunctive queries with the same arity and A_1 and A_2 are sets of peers. This kind of specification can accommodate both GAV and LAV-style mappings. The second kind of peer mappings are called *definitional mappings*. *Definitional mappings* are datalog rules whose head and body are both peer relations, i.e., the body cannot contain a query. These mappings are written as equality ones.

In order to process the queries in the peers, Piazza provides a query reformulation algorithm. The input of the algorithm is a set of peer mappings and storage descriptions and a query Q. The output is a query expression Q' that only refers to stored relations at the peers. The algorithm is considered sound and complete since Q' will always produce certain answers to Q. According to PPL semantics, the algorithm combines and interleaves unfolding (GAV) and rewriting (LAV) reformulation techniques – depending on the directionality of the mapping available.

2.4.4. Query Reformulation in an Ontology based PDMS

Xiao and Cruz [2006] describe an ontology-based PDMS (OPDMS), specifically focusing on the issue of query answering. In their work, local RDFS[3] ontologies are used to uniformly represent heterogeneous peer source schemas. In order to represent the semantic mappings among the peer ontologies, they use a mapping language named P2P Mapping Language (PML) which uses a meta-ontology called RDF Mapping Schema (RDFMS). Thus, the mapping information is stored in terms of instances of such meta-ontology. RDFMS provides one-to-one mappings such as *equivalent* (represented by EquivalentMap), *broader* (BroaderMap), and *narrower* (NarrowerMap). Regarding the case of one-to-many mappings, RDFMS defines *UnionMap* and *IntersectionMap* respectively for two types of logic combinations (i.e., *and* and *or*) of the elements on the multiple-element side.

In this approach, the process of query answering includes three steps: query execution, query rewriting, and answer integration. The user poses a query on a peer, which is first executed on that peer. Meanwhile, the query is also forwarded to each of the linked peers, where the query is reformulated into a new query that is executed locally and propagated further. Finally, answers from every peer are returned to the host peer, where they are integrated to produce the answer.

The work uses a first-order relation based method to interpret the inter-schema mappings. Actually, both the mappings and heterogeneous queries are interpreted by a set of first-order relations, so as to provide a unified environment for query rewriting. The query rewriting is dealt with as a function $Q_2 = f(Q_1,M)$, where Q_1 is the local query, M is the set of P2P mappings, and Q_2 is the resulting remote query. Besides the mappings, the query rewriting algorithm considers integrity constraints specified on local data sources.

2.4.5. Query Reformulation in "What to Ask to a Peer"

The goal of this approach is to present some results on the problem of exploiting the mappings between peers in order to answer queries posed to one given peer [Calvanese et al. 2004]. To this end, it focuses on a simplified setting based on two interoperating peers and investigate how to solve the "What-To-Ask" (WTA) problem: find a way to answer queries posed to a peer by relying only on the query answering service available at the queried peer and at the other peer. They study the WTA problem in a first-order logic (FOL) context and provide an algorithm to compute it.

[3] http://www.w3.org/TR/rdf-schema/

This work formalizes a knowledge-based peer as a tuple of the form $P = \langle K, V, M \rangle$, where: K is a knowledge base written in some subset of FOL; V is the exported fragment of K; and M is a set of mapping assertions. Two peers are considered, namely $P_\ell = \langle K_\ell, V_\ell, M_\ell \rangle$, called local peer, which is the peer to which the client is connected, and $P_r = \langle K_r, V_r, M_r \rangle$, called remote peer. This last peer does not contain mapping assertions. Clients pose their queries over the exported fragment V of a peer P.

Considering that, the reformulation algorithm first reformulates the client's query q into a set Q of conjunctive queries expressed over K_ℓ, in which it compiles the knowledge of the local knowledge base; then, according to the mapping M_ℓ, the algorithm reformulates the queries of Q into a set of queries that are accepted by the remote peer. For each query $q \in Q$, the algorithm checks if there exists an assertion stating a semantic relationship among classes and roles of K_ℓ that can be used to produce a new query to be added to the set Q. Three kinds of assertions are taken into account: (i) subsumption between classes, (ii) participation of classes in roles, and (iii) mandatory participation of classes in roles.

2.4.6. Query Reformulation in SomeRDFS

The SomeRDFS PDMS has been developed using a data model based on RDF on top of the SomeWhere infrastructure [Adjman et el. 2007]. SomeWhere is a PDMS where there are neither super-peers nor a central server. In this system, query reformulation is reduced to distributed reasoning over logical propositional theories by a propositional encoding of the distributed schemas.

In SomeRDFS, schemas are represented as ontologies, mappings are expressed in RDFS and data are represented in RDF. In this model, classes and properties can be defined as well as domain and range of properties can be typed. Mappings are defined as statements involving vocabularies of different peers and may be (i) an inclusion statement between classes or properties of two distinct peers or (ii) a typing statement of a property of a given peer with a class of another peer. Queries are conjunctive queries that may involve the vocabularies of several peers.

Query reformulation in SomeRDFS is reduced to consequence finding over logical propositional theories solved by DECA (*Decentralized Consequence finding Algorithm*) – the algorithm of SomeWhere, where each peer theory is a set of propositional clauses built from a set of propositional variables. Thus, the query reformulation algorithm of SomeRDFS, namely DeCA^{RDFS}, has been designed on top of DeCA. The strategy of DeCA^{RDFS} is to rewrite the user query's atoms independently with DeCA and then to combine their rewritings in order to generate some conjunctive rewritings of the user query w.r.t. a SomeRDFS PDMS. DeCA^{RDFS} guarantees that all the maximal conjunctive rewritings of the user query are generated.

2.4.7. Query Reformulation by Concept Approximation

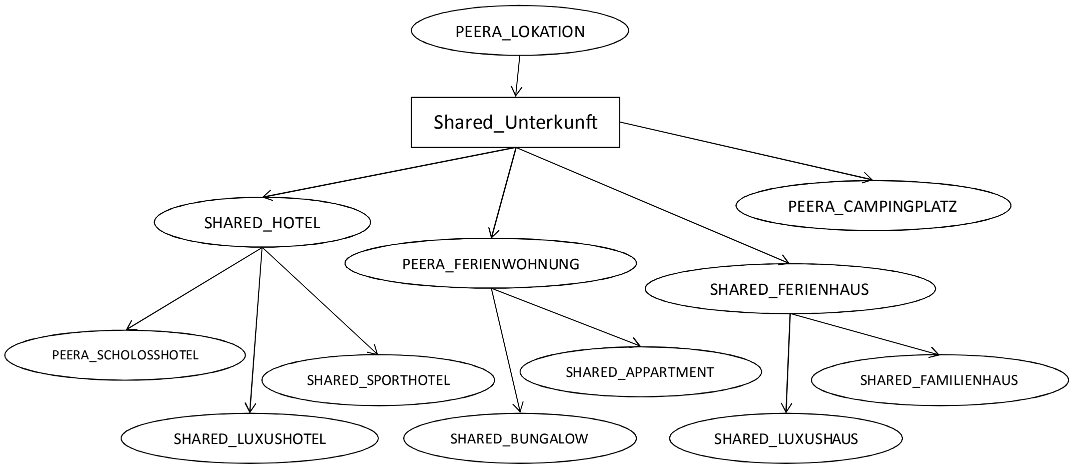
Stuckenschmidt [Stuckenschmidt 2002; Stuckenschmidt et al. 2005] has shown that query translation can be done in an approximate way using terminological reasoning and query relaxation. The idea is to compute approximate answers to conjunctive queries by transforming the query into a concept expression and estimating its closer position in a determined hierarchy of a remote peer. This hierarchy fixes the upper and lower bounds of a concept name.

They define the structure of a terminological knowledge base and its instantiation independent of a concrete language, using Description Logics. They formalize terminological queries as the following: conjuncts of a query are predicates that correspond to classes and relations of an ontology. They use a method for translating such conjunctive queries into concept expressions (in a Boolean model) that has been proposed by Horrocks and Tessaris [Horrocks and Tessaris 2000]. The idea is to treat the query as a concept expression in the ontology and classify it with respect to the concepts of the remote peer ontology. In this sense, the adaption of a query to remote peers can be done by rewriting the concept names in the query by their approximations (upper or lower) in the remote source. Thereby, instances of subsumed/subsuming concepts are returned as result.

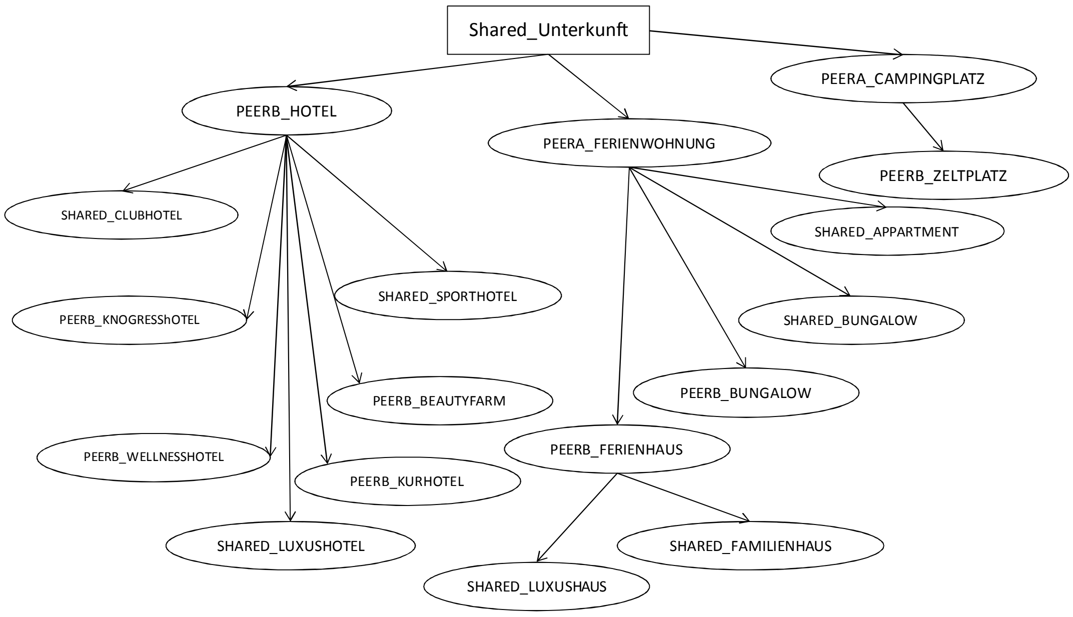
Since mappings between ontologies are usually sparse, objects that are meant to be an answer to a query may not be returned because their description does not match the query that is formulated using terms from a different ontology. To address such problem, they also provide query relaxation, i.e., the query is simplified by weakening constraints from the query expression that are responsible for that failure. The intuition underlying that is to start with the original query and generate queries where each is more general than the one before, i.e., each query following in the sequence returns all results of the previous one, but might return more results.

As an example, consider two overlapping concept hierarchies belonging to the domain of tourism (excerpts from them are depicted in Figure 2.7).

Regarding concept approximation, an example is provided through the concept “Ferien-Wohnung” (a flat used as accommodation during holidays). First, interesting approximations are searched in both peer hierarquies. In Peer A, sub-concepts of the example concept are: “Bungalow” and “Appartment”. In Peer B, there is also the concept “Ferienhaus” (house used during holiday) which falls under this category. The upper approximation is obtained through the general concept “Unterkunft” (accommodation) in both peers. After fixing the upper approximation, the method determines all instances of the general concept to be potential members of the example concept. Besides the members of the general concept, this also includes objects that are members of the concepts “Hotel” and “Campingplatz” (camp site) in the view of the answering peer B. The example does not report how they deal with the concepts of lower approximation. Most of results obtained through upper approximation are considered related to the example concept, because they are all accommodations used during holidays. However, hotels and camp sites are not the kind of answer the user would assume to get when asking for a flat, but may be considered approximate as well.



a) Peer A



b) Peer B

Figure 2.7 Hierarchies of two different peers on the same domain

As an example of query relaxation, suppose a query which aims at obtaining "the number of rooms of a hotel". Such query would be translated to a concept expression. But, since none of the ontologies except for the one the query is based on contains information about the number of rooms of a hotel, it is impossible to prove that a specific hotel is an answer to the query. As a result, they relax the query by removing the restriction on the number of rooms. Consequently, they get all hotels as potential answers. The authors argue that it is very likely that they could further improve the accuracy of the approximation by using logical reasoning for finding alternatives for class and relation names rather than removing them from the query.

2.4.8. Comparative Analysis

Table 2.6 summarizes the main features of the different query reformulation approaches that have been covered in this section. These approaches have been developed in different settings: most of them have been developed in distributed environments, excepting the work of Necib [2007], which has been implemented within single databases. In general, the works use the relational model to data sources, SQL and conjunctive queries to express queries, and some formalisms including First Order Logic and Description Logics to formalize their concepts and query reformulation approaches.

Regarding mappings/correspondences specification, most of them considers mappings between peer ontologies (Piazza considers mappings among peer schemas). The mappings considered are usually restricted to equivalence and subsumption (SomeRDFS also considers disjunction). As a result , the majority of them deal with query reformulation by means of query translation, without considering any kind of semantics. Exceptions to that are the work of Kostadinov [2007] and the work of Stuckenschmidt et al. [2005]. The former introduces data personalization inside a mediator-based system based on user preferences. The latter uses terminological reasoning by concept approximation and query relaxation to provide query translation. Also, the work of Necib [2007] uses semantic knowledge provided by a domain ontology. Thereby, these last three works have a different query reformulation approach by means of some kind of semantics usage.

Table 2.6: Comparative Analysis of Query Reformulation Approaches

Approach	Environment	Representation Model	Formalism	Query Language	Mapping/Correspondence Types	Semantics Usage	Reformulation Rules
[Necib 2007]	Single Databases	Relational	Term Rewriting Systems	SQL	Equivalence (between database schema and ontology)	Ontology as additional knowledge	Extension Rules (e.g. part-whole) Reduction Rules (e.g. sensitivity)
[Kostadinov 2007]	Mediator-based System	Relational	Conjunctive Query	SQL	LAV mappings	User Profiles	Enrichment Rules Translation rules using LAV approach
Piazza [Halevy et al. 2005]	PDMS	Relational and XML	Conjunctive Query	XQuery or Conjunctive Query	Equivalence, Inclusion and Definitional Mappings	Metadata in a Catalog	Translation Rules, using GAV/LAV approaches
OPDMS [Xiao and Cruz 2006]	PDMS	RDF	FOL (First Order Logic)	Conjunctive RQL Query	Equivalence, Broader, Narrower, Union and Intersection	Mapping Ontology	Translation Rules
WTA [Calvanese et al. 2004]	PDMS	Knowledge-based (First Order Logic - FOL)	FOL (First Order Logic)	FOL Query	Subsumption between classes, Participation of classes in roles, Mandatory participation of classes in roles	—	Translation Rules
SomeRDFS [Adjman et al. 2007]	PDMS/Semantic Web	RDF	DL (Description Logics) and FOL (First Order Logic)	FOL Query	Equivalence, Inclusion, Disjunction	—	Translation Rules
Concept Approximation [Stuckenschmidt et al. 2005]	Weakly-Structured Environments	Terminological Knowledge base using DL	DL (Description Logics)	Boolean Query	Equivalence, Specialization (Lower Approximation), Generalization (Upper Approximation)	Terminological reasoning and query relaxation	Concept Approximation in terms of Lower and Upper Bounds

2.5. Concluding Remarks

In this chapter, we provided an overview of query reformulation in Data Integration Systems and Peer Data Management Systems. We introduced query reformulation by using semantics, including terms expansion and query personalization. Finally, we presented a survey of some query reformulation approaches proposed in the literature.

Most query reformulation approaches have mainly concentrated on translating a source query to an exact target query by using traditional correspondences such as equivalence and, sometimes, subsumption. Nevertheless, dynamic distributed environments, such as PDMS, also need techniques to improve the relevance of query answers, considering that users are usually more interested in answers that fit their needs, or that are closer to what they define as relevant at query formulation time. In Chapter 5, we propose a new query reformulation approach that is targeted at fulfilling these requirements.

CHAPTER 3

"The meaning of things lies not in the things themselves,
but in our attitude towards them."
Antoine de Saint-Exupery

Semantic Issues

In a general sense, semantics is the study of meanings of the message underlying the words or underlying certain elements that need to be interpreted in a given task or situation. In dynamic distributed environments, semantics may be identified considering the user's perspective, the peers' perspective or even the query formulation. Since such environments perform services over data from existing heterogeneous sources, they must be able to deal with different categories of heterogeneity such as structural and semantic heterogeneity. As a result, metadata must be used to describe the content of the data sources in a comprehensive and uniform way. To provide a shared understanding of the terms that are being evaluated, a domain ontology may be used as a semantic reference. In addition, context may be used to improve the system's services.

In this chapter, we provide an overview of some semantic issues, particularly, *ontologies*, *context* and the *Description Logics* formalism. These issues are been increasingly used as a means for enhancing query answering in distributed environments. Thus, this chapter covers the following contents: Section 3.1 defines ontology and related notions, presenting its potential benefits in distributed settings' processes; Section 3.2 describes Description Logics, providing some background understanding of this formalism; Section 3.3 introduces the concept of context and how it can be used. Finally, Section 3.4 concludes the chapter with some considerations.

3.1. Ontology

In the last years, the term "ontology" has been increasingly used in diverse computational areas, including Artificial Intelligence, Databases and Information Retrieval. However, there are many views of what an ontology is supposed to denote.

The Webster dictionary online[4] defines the term ontology as "the metaphysical study of the nature of being and existence". In Artificial Intelligence, an ontology was initially defined as an "explicit specification of a conceptualization", where a conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose [Gruber 1993]. As an illustration, for a biologist, a conceptualization may include that animals can be classified in groups called species and that the animals belonging to species have similar eating habits. Based on these eating habits, the species can be subcategorized into herbivores, carnivores and omnivores [Borst 1997].

Studer and his group [Studer et al. 1998] go further and define ontology as a formal, explicit specification of a shared conceptualization, where *formal* means machine-readable, *explicit specification* entails that the concepts, properties, relations, constraints and axioms are explicitly defined, *shared* means consensual knowledge and *conceptualization* is an abstract model of some phenomenon in the world.

In summary, we can say that an ontology is a representation of a shared understanding of concepts in a particular domain of interest as agreed by a community. This representation must be clear, concise, and consistent for such community [Necib 2007], where a community can be a group of people or computer systems that interact with one another within a common domain of interest. In order to make up an ontology, we need to deal with a vocabulary of the basic terms, a precise specification of what those terms mean and how they relate to each other.

Typical real-world ontologies include taxonomies on the Web (e.g., Yahoo! categories) and top-level ones, i.e., ontologies with general concepts (e.g., SUMO - *Suggested Upper Merged Ontology*[5]). There are also domain ontologies which describe the vocabulary related to a given domain (e.g., *Health* or *Geography*) and are the most commonly found ones. Some examples of domain ontologies which have been constructed over the years are: UMLS[6], Gene Ontology[7] and Core Legal Ontology[8].

3.1.1. Basic Notions

Although there are different definitions of the term "ontology", some basic notions regarding its structure are shared by most approaches. Considering that an ontology can be represented by a hierarchy of concepts and a hierarchy of relations, we describe some of its elements and their meaning as follows [Noy and McGuinness 2001]:

[4] http://www.websters-online-dictionary.org/
[5] http://www.ontologyportal.org/
[6] http://www.nlm.nih.gov/research/umls/
[7] http://www.geneontology.org/
[8] http://www.estrellaproject.org/lkif-core/
[6] http://wordnet.princeton.edu/

- Concept (also known as class): A concept is a representation for a conceptual grouping of similar terms. For example, a Vehicle could be represented as a concept which would have many sub-concepts such as Car, Motorbike.
- Properties (also known as slots): A property is seen as a relation, since it is used to describe a relationship between two terms. The first term must be a concept that is the *Domain* of the relation and the second must be a concept that is the *Range* of the relation. For example, drives could be represented as a relation such that its domain is Person and its range is Vehicle. The range of a property may also be a *datatype* such as string, integer, real or boolean. A relation may have sub-relations. For example, firstName, lastName and title could be sub-relations of the relation designation.
- Instance: An object is an instance of a concept whether it is a member of the set denoted by that concept. For instance, Anna is an instance of Person.

The knowledge captured in ontologies can be used, among other things, to annotate data, generalize or specialize concepts, and infer entirely new (implicit) information. Some semantic relationships can be usually identified in an ontology:

- Subsumption: the relationship between a concept and its sub-concepts or between a property and its sub-properties is called an "is_a" (or specialization) relationship. This relation can be either declared in an ontology or inferred through transitivity.
- Disjointness: it means that two terms do not overlap. This relationship can also be declared or inferred.
- Sibling: this relationship occurs when two terms share a common ancestor (a common super-concept), which can be either a direct or an indirect parent.

Taking into account the structure of an ontology, we can determine the semantic distance of two concepts. In general, semantic distance denotes the degree of semantic association between concepts [Scriver 2006]. Considering the taxonomy (hierarchy of concepts) of an ontology, we provide some notions which are relevant when determining the semantic distance between concepts:

- Length: The length between two concepts C_1 and C_2 is the shortest allowable path connecting C_1 and C_2 (measured in nodes) in the taxonomy.
- Depth: The depth of a concept C in a taxonomy is the length of the path from the root to the concept C.
- Height: The height of a concept C in a taxonomy is the length of the path from the concept C to the deepest leaf. The height of a taxonomy is the height of the root.

As an example, consider the *Animal* taxonomy as depicted in Figure 3.1. Some illustrative obtained measures are presented as follows:

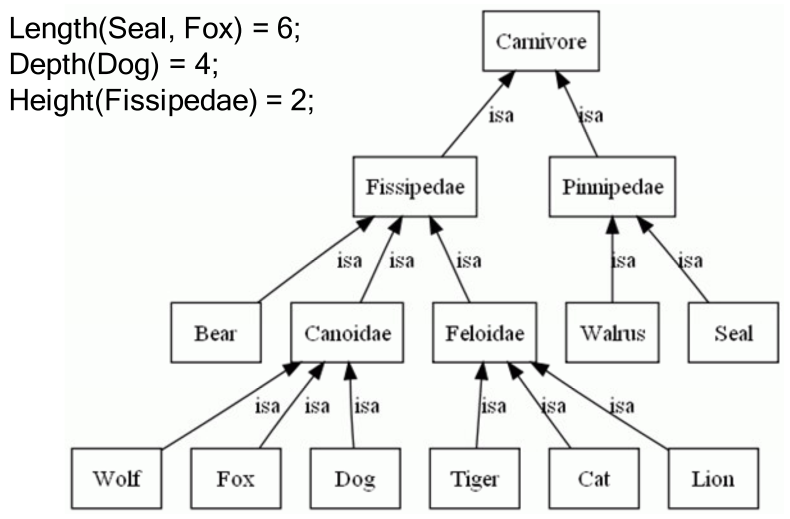


Figure 3.1 An excerpt from an Animal Taxonomy

3.1.2. Ontology Reasoning and Representation

An ontology provides the mechanisms for modeling a domain of interest and reasoning upon it, and has to be represented in terms of a well-defined language. According to Lenzerini and his group [2009], "reasoning over an ontology" means any mechanism that makes explicit facts that are represented implicitly in an ontology. Methods for reasoning about ontologies are required for several reasons, including [Lenzerini et al. 2009]: (i) validation, which means ensuring that the ontology is a good representation of the domain of discourse and (ii) analysis which provides the inference of new facts about the domain that are implicitly represented.

Ontologies are usually expressed in an ontology language. Since they are a knowledge representation technique based on Description Logics (DL) [Baader et al. 2003], it is common to find ontologies expressed in such formalism. Due to the use of DLs in this work, we provide a more complete presentation of their syntax and semantics in Section 3.2. Here we briefly discuss the Web Ontology Language – OWL[9] which is considered the standard language for the representation of ontologies on the Semantic Web [Berners-Lee et al. 2001] and which is itself based on Description Logics.

OWL is an XML-based markup language which allows the construction of the different elements of an ontology based on RDF graphs. In RDF, elements are represented using a triple pattern logic [Chamiel and Pagnucco 2008]. Each element in the RDF graph has to be a literal or an RDF resource. For example, the following tag

[9] http://www.w3.org/TR/owl-features/

constructs an OWL class (concept) named Food: <owl:Class rdf:ID="Food">. Here, the *subject* is a class definition, the *predicate* is the class id and the *object* is the string "Food". Properties and individuals can be created in the same manner. For example, the following tag constructs a new individual food <Food rdf:ID="Carrot">.

OWL is based on the $\mathcal{SH}$ family of Description Logics [Horrocks 2005], which besides the traditional boolean constructs and quantification, allows for enforcing roles (i.e., binary predicates) to be transitive, and for forming role hierarchies [Lenzerini et al. 2009]. OWL has three flavors of expressivity: OWL Lite, OWL DL and OWL Full.

3.1.3. Ontology in Distributed Environments

There is a growing interest in ontologies for enhancing data management in distributed environments. Due to the fact that ontologies provide good supports for understanding the meaning of data, they have been used in data integration systems to overcome problems caused by the heterogeneity of data and to optimize query answering among the distributed sources. In these settings, they have been used for some purposes, including [Xiao 2006]: (i) metadata representation, where metadata in each data source are represented by a local ontology; (ii) global conceptualization, providing a conceptual view over the schematically heterogeneous source schemas; (iii) support for high-level queries, where, given a global ontology, the user can formulate a query without specific knowledge of the different data sources.

In addition, mainly due to semantic heterogeneity, research on PDMS has also considered the use of ontologies as a way of providing a domain reference and describing data sources in a uniform notation. Considering a given knowledge domain, an agreement on its terminology can occur through the definition of a domain ontology which may be used as a semantic reference or background knowledge in order to enhance processes such as ontology matching.

Since ontologies are usually used to represent peer schemas in distributed environments, there are several forms of heterogeneity between them. These heterogeneities are classified as follows [Euzenat and Shvaiko 2007].

- Syntactic heterogeneity: occurs when two ontologies are not expressed in the same ontology language, e.g., when two ontologies are modeled by using different representation formalisms, such as OWL and F-logic.
- Terminological heterogeneity: occurs due to variations in names when referring to the same entities in different ontologies. This can be caused by the use of different natural languages, e.g., Paper vs. Articulo, different technical sublanguages, e.g., Paper vs. Memo, or the use of synonyms, e.g., Paper vs. Article.

- Conceptual heterogeneity: also called semantic heterogeneity, stands for the differences in modeling the same domain of interest. It can happen due to the use of different axioms for defining concepts or due to the use of totally different concepts, e.g., geometry axiomatised with points as primitive objects or geometry axiomatised with spheres as primitive objects. More specifically, it may occur due differences in [Benerecetti et al. 2001]:
    - Coverage: occurs when two ontologies describe different, possibly overlapping, regions of the world at the same level of detail and from a unique perspective. This is, e.g., the case of two partially overlapping geographic maps.
    - Granularity: occurs when two ontologies describe the same region of the world from the same perspective but at different levels of detail. This applies, e.g., to geographic maps with different scales, i.e., one displays buildings, while another depicts whole cities as points.
    - Perspective: also called difference in scope, occurs when two ontologies describe the same region of the world, at the same level of detail, but from a different perspective. This occurs, e.g., for maps with different purposes such as a political map and a geological map which do not display the same objects.
- Semiotic heterogeneity: also called pragmatic heterogeneity, is concerned with how entities are interpreted by people. Indeed, entities which have exactly the same semantic interpretation are often interpreted by humans with regard to the context of how they have been ultimately used.

In order to deal with such heterogeneities, matching processes are used. *Matching* is the process of finding relationships or correspondences between elements of different ontologies, and *alignment* is a set of correspondences between two or more ontologies, i.e., the output of the matching process [Euzenat and Shvaiko 2007]. Still, a *correspondence* is the relation holding between elements of different ontologies. These elements can be concepts, individuals, properties or formulas.

Nevertheless, in order to make an effective use of existing agreed ontologies (which may be used as domain ontologies), we have to verify if they are correct. This means verifying whether the modeled entities and properties in an ontology correctly represents entities in the world being modeled. To this end, a methodology for validating the ontological adequacy of taxonomic relationships, called *OntoClean,* has been presented by Guarino and Welty [2002]. As an example, DOLCE[10] is an upper ontology built on the principles of *OntoClean*.

[10] http://www.loa-cnr.it/DOLCE.html

OntoClean is based on general ontological notions drawn from philosophy, like *rigidity (essence)*, *identity*, and *unity*. These notions are described as follows [Fernández-López and Gómez-Pérez 2002]:

- *Rigidity*: This notion is defined based on the idea of essence. A property is essential to an individual if and only if it necessarily holds for that individual. Thus, a property is *rigid* (+R) if and only if it is necessarily essential to all its instances. A property is *non-rigid* (-R) if and only if it is not essential to some of its instances, and *anti-rigid* (~R) if and only if it is not essential to all its instances. For example, the concept person is usually considered rigid, since every person is essentially such, while the concept student is normally considered anti-rigid, since every student can possibly be a non-student a few years later.
- *Identity*: A property carries an identity criterion (+I) if and only if all its instances can be (re)identified by means of a suitable "sameness" relation. A property supplies an identity criterion (+O) if and only if such criterion is not inherited by any subsuming property. For example, person is usually considered a supplier of an identity criterion (for example the fingerprint), while student just inherits the identity criterion of person, without supplying any further identity criteria.
- *Unity*: an individual is a whole if and only if it is made by a set of parts unified by a relation R. For example, the enterprise Iberia is a whole because it is composed by a set of people that are linked by the relation having the same president. A property P is said to carry *unity* (+U) if there is a common unifying relation R such that all the instances of P are wholes under R. For example, the concept enterprise-with-president carries unity because every enterprise with president is made up people linked through the relation having the same president. A property carries *anti-unity* (~U) if all its instances can possibly be non-wholes. Properties that refer to amounts of matter, like gold, water, etc., are good examples of anti-unity.

In cleaned ontologies which hold such notions, these meta-properties are assigned to concepts. For example, the ontology depicted in Figure 3.2 is an example of a cleaned ontology. These meta-properties impose constraints on the subsumption relation, which can be used to check the ontological consistency of taxonomic links. For example, the class student cannot subsume the class person if the former is *anti-rigid* and the latter is *rigid*. For more details and a complete example of usage, we refer the work of Guarino and Welty [2004]. Next section, we describe the Description Logics formalism.

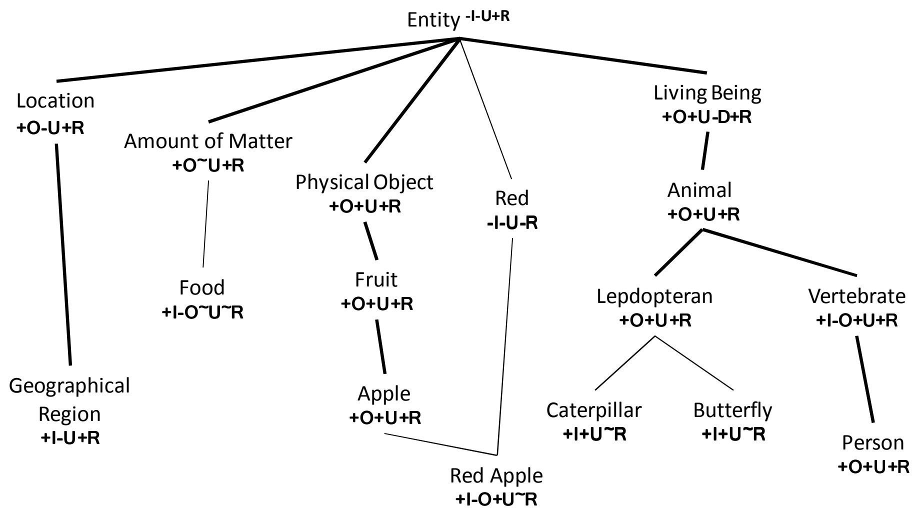


Figure 3.2 An excerpt from a cleaned Ontology [Guarino and Welty 2004]

3.2. Description Logics

Description Logics (DLs) are considered a family of knowledge representation formalisms that allow representation of domain knowledge and reasoning with it in a formally well understood way [Baader et al. 2003; Horrocks 2005]. They provide the specification of *concepts* (classes), *individuals* (instances) and *roles* (properties). Operators, such as negation (¬) or conjunction (⊓), can be used in order to build more complicated composite concepts. As an initial example, consider the following concept:

Female ⊓ ≥ 2hasChild ⊓ ∀haschild.Female

This axiom describes females who have at least two children and all of whose children are female.

By using a DL language, users can build a terminology of agreed terms and use a knowledge representation system to store and reason about such terminology. A terminology, also called a TBox, is a set of axioms that induce a concept taxonomy. The basic form of declaration in a TBox is a concept definition, that is, the definition of a new concept in terms of other previously defined concepts. For instance, a woman can also be defined as a female person by writing an equivalence declaration, as follows:

Woman ≡ Person ⊓ Female

Inclusion axioms of the form C ⊑ D, where C and D are arbitrary concept expressions, can also be defined. An example is given by the axiom Professor ⊑ Person, meaning that the concept Professor is a specialization of the concept Person and hence every instance of the concept Professor must also be an instance of the concept Person. Description Logics may also allow for a role hierarchy, also called an

RBox, where, e.g. the axiom hasSon $\sqsubseteq$ hasChild states that all pairs of individuals that are related through the role hasSon are also related through the role hasChild [Glimm 2007]. If a DL supports transitive roles, one can state that the role hasDescendent is transitive. In turn, a DL reasoner can deduce that, if hasDescendent(Anna, John) and hasDescendent(John, Carl) holds, then hasDescendent(Anna, Carl) also holds.

Assertions about individuals have the form Female(Anna), hasChild(Anna, John), or Anna $\neq$ Ana, and are called *concept assertions*, *role assertions*, and *inequality assertions*, respectively. A collection of assertions about individuals is called an *ABox*. Therefore, a TBox, RBox (if supported by the DL language), and ABox together constitute a knowledge base (that may be implemented as an ontology).

3.2.1. Description Logics Languages

Particular DL languages are mainly characterized by: (i) a set of constructors for building complex concepts and roles from simpler ones and (ii) a set of axioms for asserting facts about concepts, roles and individuals. In this section, we present the $\mathcal{ALC}$ language.

The name $\mathcal{ALC}$ stands for "Attributive concept Language with Complement". It is obtained from $\mathcal{AL}$ by adding the complement operator ($\neg$). Formally, $\mathcal{ALC}$ syntax is defined as follows [Baader et al. 2007]: Let N_C be a set of concept names and N_R be a set of role names. The sets of $\mathcal{ALC}$-concept descriptions are the smallest ones such that:

1. $\top$, $\bot$, and every concept name $A \in N_C$ is an $\mathcal{ALC}$-concept.

2. If C and D are $\mathcal{ALC}$-concepts and $r \in N_R$, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$ and $\exists r.C$ are $\mathcal{ALC}$-concepts.

The $\mathcal{ALC}$ constructors are: conjunction, disjunction, negation, existential restriction and value restriction.

One important application of DLs is being used as the formal foundation for ontology languages. OWL is an example of DL based ontology language, as explained in Section 3.1.2. In particular, $\mathcal{ALC}$ has been extended with several features that are important in an ontology language, including number restrictions, inverse roles, transitive roles, subroles, concrete domains, and nominals. With number restrictions, it is possible to describe the number of relationships of a particular type that individuals can participate in, e.g., we may say that a person is married to at most one other individual: Person $\sqsubseteq$ $\leq$1married. With qualified number restrictions, we can additionally describe the type of individuals that are counted by a given number restriction. We can define a HappyMan including the fact that instances of HappyMan have at least two children who are doctors, such as [Baader et al. 2007]: $\mathsf{HappyMan} \equiv \mathsf{Human} \sqcap \neg\mathsf{Female} \sqcap (\exists\mathsf{married.Doctor}) \sqcap (\forall\mathsf{hasChild.}(\mathsf{Doctor} \sqcup \mathsf{Professor})) \sqcap {\geq}2\mathsf{hasChild.Doctor} \sqcap {\leq}4\mathsf{hasChild}$. Also, with inverse roles, transitive roles, and subroles, we can, in addition to hasChild, use its inverse hasParent, specify that hasAncestor is transitive, and specify that hasParent is a subrole of hasAncestor.

The name given to a particular DL usually reflects its expressiveness, with letters indicating the provided constructors. The letter $\mathcal{S}$ is often used as an abbreviation for the "basic" DL consisting of $\mathcal{ALC}$ extended with transitive roles. The letter $\mathcal{H}$ represents subroles (role **H**ierarchies), $\mathcal{O}$ represents nominals (n**O**minals), $\mathcal{I}$ represents inverse roles (**I**nverse), $\mathcal{N}$ represent number restrictions (**N**umber), and $\mathcal{Q}$ represent qualified number restrictions (**Q**ualified). The integration of a concrete domain/datatype is indicated by appending its name in parenthesis, but sometimes a "generic" $\mathcal{D}$ is used to express that. Composing such representative identifications, we have the so-called extended DL $\mathcal{SHOIN(D)}$ which corresponds to the OWL DL ontology language and includes all these mentioned constructors [Glimm 2007].

3.2.2. Distributed Description Logics

Distributed Description Logics (DDL), introduced by Borgida and Serafini [2003] are a natural generalization of the DL framework designed to formalize multiple ontologies interconnected by semantic mappings [Homola 2007]. For instance, suppose we have two information sources IS_1 and IS_2, each using some (potentially different) description logic to describe its contents. We can establish connections between them, e.g., $\mathsf{GradStudent} \sqsubseteq_{int} \mathsf{Student}$ would indicate that every graduate student in IS_2 is also a student in the overlapping part of the world described by IS_1.

In DDL, a *distributed T-Box* consists of local T-Boxes T_i, described using ordinary DLs, and *bridge-rules* relating them. In order to support directionality, it is necessary to define two sets of bridge rules B_{12} and B_{21} from IS_1 to IS_2, and vice-versa. A bridge rule from i to j is an expression of the following two forms [Ghidini and Serafini 2006]: (i) $i{:}x \xrightarrow{\sqsubseteq} j{:}y$, named *into bridge rule* or (ii) $i{:}x \xrightarrow{\sqsupseteq} j{:}y$, denoted as *onto bridge rule,* where x and y are either two concepts, or two roles, or two individuals of DL_i and DL_j, respectively. In this sense, the into-bridge rule $i{:}x \xrightarrow{\sqsubseteq} j{:}y$ states that, the concept x in i is less general than its local concept y. Similarly, the onto-bridge rule $i{:}x \xrightarrow{\sqsupseteq} j{:}y$ expresses the fact that, x in i is more general than y in j. Hence, bridge rules from i to j represent the possibility of j's ontology to translate (under some approximation) the concepts of foreign i's ontology into its internal model [Ghidini and Serafini 2006].

Example

To illustrate DDLs semantics and syntaxes, we consider the following DDL system that contains two peers with their ontologies describing their schemas, depicted in Table 3.1 and Table 3.2 (and graphically in Figures 3.3 and 3.4). The schemas are concerned with an "Education" domain and include concepts such as Professor and its Publications. Each ontology is formalized as a T-Box (including an R-Box) and an A-Box. Between both ontologies, we have a set of bridge-rules (B_{12} and B_{21}). For the sake of simplicity, we consider two concepts from each ontology and a primitive (root) concept called *thing*. Furthermore, we consider three kinds of bridge rules (into, onto and equivalent), where an equivalence bridge rule is defined as the conjunction of into and onto bridge rule.

Table 3.1 Ontologies O_1 and O_2 (TBox and ABox)

O_1 (TBox)	Teacher ⊑ Thing
	Publication ⊑ Thing
	Teacher(∀name ⊓ ∃hasPublication)
	Publication(∀number ⊓ ∃year ⊓ ∃isPublishedBy)

(ABox)	Teacher(John, 323);
	Publication(323, 2005, John)

O_2 (TBox)	Professor ⊑ Thing
	Pub ⊑ Thing
	Professor((∀first_name ⊓ ∀last_name) ⊓ ∃isAuthorOf)
	Pub((∀num ⊓ ∃year ⊓ ∃hasAuthor)

(ABox)	Professor((Anna, Gomes), 245);
	Pub(245, 2003, (Anna, Gomes))

Table 3.2 Bridge-Rules between O_1 and O_2

B_{12}	O_1.Teacher $\xrightarrow{\sqsubseteq}$ O_2.Professor O_1.Teacher(name) $\xrightarrow{\sqsubseteq}$ O_2.Professor(first_name ⊓ last_name) O_1.Teacher(hasPublication) $\xrightarrow{\sqsubseteq}$ O_2.Professor(is_Author_of) O_1.Publication $\xrightarrow{\equiv}$ O_2.Pub O_1.Publication(number) $\xrightarrow{\equiv}$ O_2.Pub(num) O_1.Publication(isPublishedBy) $\xrightarrow{\equiv}$ O_2.Pub(hasAuthor)
B_{21}	O_2.Professor $\xrightarrow{\sqsupseteq}$ O_1.Teacher O_2.Professor(first_name ⊓ last_name) $\xrightarrow{\sqsupseteq}$ O_1.Teacher(name) O_2.Professor(isAuthorOf) $\xrightarrow{\sqsupseteq}$ O_1.Teacher(hasPublication) O_2.Pub $\xrightarrow{\equiv}$ O_1.Publication O_2.Pub(num) $\xrightarrow{\equiv}$ O_1.Publication(number) O_2.Pub(hasAuthor) $\xrightarrow{\equiv}$ O_1.Publication(isPublishedBy)

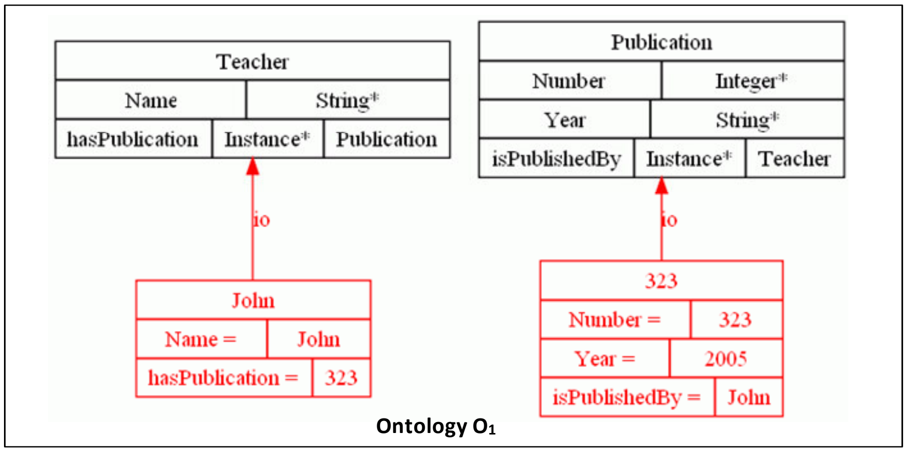


Figure 3.3 Ontology O_1 (using OntoViz[11] Notation)

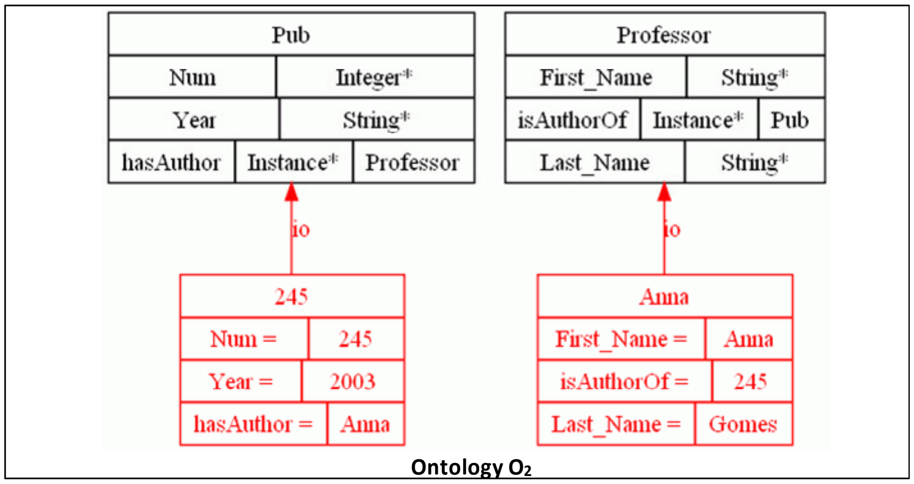


Figure 3.4 Ontology O_2

Another kind of semantic knowledge is context [Dey 2001]. Next section, we introduce such concept with related relevant notions and applications.

3.3. Context

The notion of *context,* which firstly emerged in Psychology and Philosophy [Chalmers 2004], has recently become an active field of research in areas related to Computer Science, such as *Ubiquitous Computing (UC)*, *Human-Computer Interaction* (HCI) and *Data Integration (DI).* Early work considered context to be related to the user´s location, nearby people and the resources that could be accessed [Schilit et al. 1994]. Other work has also tried to predict users' needs [Cai et al. 2003], helping in their activities, as well as adapting the system's behavior according to some situational

[11] http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz

circumstances that change over time. In works regarding Data Integration, context has been mainly used to represent different understanding of data and schema elements [Kashyap and Sheth 1996; Goh 1997; Belian 2008].

Context is usually concerned with some specific situation, usually perceived as a set of variables that may be of interest for an agent [Bolchini et al. 2007]. Context may also be understood as the circumstantial elements that make a situation unique and comprehensible [Dey 2001]. More abstractly, Vieira [2008] makes a distinction between the concepts of contextual element (CE) and context. The former is any piece of data or information that enables to characterize an entity in a domain. The latter is the set of instantiated contextual elements that are necessary to support a task at hand.

Bazire and Brézillon [2005] have collected a set of approximately 150 definitions of context coming from different domains. The two main conclusions drawn from their work are: (i) the context acts like a *set of constraints* that influence the behavior of a system embedded in a given task and; (ii) the definition of context depends on the *field of knowledge* that it belongs to. As a result, the standpoints from which the notion of context is considered are different: in UC, the context is specifically analyzed in terms of its physical sources and parameters; in HCI, the context is mainly taken into account through the history of the user-application dialogues; in DI, context information is employed to improve schema integration and query answering.

Although there are diverse definitions of what context means, most of researchers agree in some points [Vieira et al. 2006]: (i) context only exists when related to another entity (e.g., task, class); (ii) context is defined as a set of items (e.g., properties, rules) associated to an entity; and (iii) each item is considered as part of a context only if it is useful to support the problem at hand. For example, the proposition "it is raining" is defined as part of the context in a *traffic jam support system*, since rain has implications in visibility, speed and in the traffic. However, the same proposition is not considered contextual information in a *museum guide system.*

3.3.1. Context-Sensitive Systems

Context-sensitive computing deals with the ability of computer systems to obtain contextual knowledge in order to improve services or tasks. As a consequence, it creates a new generation of applications – called *context-sensitive systems (CSS)* - in which the user-application interaction or a common service provided by the system (e.g., query answering) is enhanced by perceiving/sensing the surrounding context. An interesting CSS example is the *Dynamic Tour Guide* (DTG) [Kramer et al. 2005]. DTG is a mobile agent enabling a personalized spontaneous guided tour. It plans an individual tour, selecting attractions, providing navigational guidance and offering environmental information. To this aim, it gathers all available information (e.g., location, personal interests, current time, walking speed) and filters them. As a result, the system adapts the tour according to this set of contextual information.

Despite different context views, one issue is a consensus: developing CSS implies in managing contextual information. A CSS should include a context management service which would be responsible for handling contextual information. This functionality may be developed as a framework, a middleware or a set of web services in a modular way, so other applications or services can interact with it.

Based on our analysis of the available CSS state-of-the-art, we have selected a set of tasks usually accomplished by a context management service. In the following, we briefly discuss each one.

- *Acquisition.* The quality of context-aware services is dependent on the quality of information collected from the context sources. Context data may be captured in four ways: (i) from physical or hardware sensors (e.g., GPS, microphones); (ii) from logical sensors – intelligent agents or services that are able to collect context; (iii) from explicit input – some information may be provided explicitly by the user (e.g., preferences) or (iv) from static sources - by user profiles or by information stored in databases.
- *Representation.* A challenge to be faced in context management is the fact that there is not still a standard model for representing contextual information. On the other hand, it is common sense that context representation is therefore becoming a necessity in most application domains. Current research has worked with a considerable number of context representation techniques, such as Contextual Graphs [Brézillon 2005], Ontologies [Souza et al. 2008], Topic Maps [Power 2003]. Due to its relevance to our work, we provide more details regarding context representation in Section 3.3.2.
- *Reasoning*. As soon as the context data is acquired, it should be interpreted to provide for the appropriate system reaction or adaptation. Thus, reasoning mechanisms may be used to process contextual information, i.e., to deduce high-level implicit context from low-level explicit context.
- *Storage*. Almost every piece of contextual information is supposed to be stored in a way that enables later recovery. Sometimes, however, if the CSS reasoning only depends on the current value of sensors, then the context storage is not necessary and may be taken away.
- *Sharing*. A scenario with lots of users and applications entails the necessity of context-sharing. It also implies on defining privacy and security.

- *Reusability*. Context should be reused in future, saving new effort. Besides, users would not be required to answer questions or to define preferences more than once. Reusability would also avoid continuous acquisition.
- *Evolution.* Context evolution is hard to predict. It is essential that some mechanism may be used to plan context changing when applications also need change.
- *Accountability*. Context sensitiveness implies allowing users to make informed decisions based on context. The context management service should provide feedback to users and control to them in cases of conflict of interests [Bellotti and Edwards 2001].

Benefits such as *adaptation*, *personalization*, and *awareness (i.e.,* pro-activity*)*, are expected from a CSS [Vieira et al. 2006]. Adaptation means to adjust a service or information according to available contextual elements. It includes operations such as filtering of information or invocation of additional services. Personalization means to adapt an application to different people, such that they perceive the application differently at the same time, according to each person's preferences or skills. Aawareness is the service which delivers or pushes information to a client without explicit request. This service works autonomously as a background process, informing the user as configured.

3.3.2. Context Representation

In order to allow context usage, it is crucial to define how context will be represented. Some issues should be considered when evaluating techniques to represent context: (i) the model must be portable; (ii) it should have validation tools for edition, type checking and conversion between formats; (iii) formality is welcome since it eases definition, reasoning and reusability; and (iv) it must allow reasoning.

Current research has worked with a considerable number of context representation techniques [Strang and Linnhoff-Popien 2004], such as Contextual Graphs [Brézillon 2003], Topic Maps [Power 2003] and Ontologies [Wang et al. 2004; Souza et al. 2006]. A contextual graph is an acyclic directed graph and allows a context-based representation for operational processes by taking into account the working environment [Brézillon 2003]. Topic maps are an attempt to connect pieces of data into a graph which represent the relationship between them while providing a lightweight way of navigating the information [Power 2003]. Shared ontologies, as explained in Section 3.1, are fundamental for reusing knowledge, serving as a means for integrating problem-solving, domain representation and knowledge acquisition modules [Wang et al. 2004].

According to the issues pointed out above and to some recent works [Wang et al. 2004; Souza et al. 2008], ontologies seem to be one of the best options for context representation. There are several advantages for developing ontology-based context models [Wang et al. 2004; Souza et al. 2008], namely: to provide knowledge sharing (services are supposed to deal with the same set of concepts), to enable knowledge reuse, to define semantics independently from data representation, and finally to enable the use of existing inference engines. On the other hand, a drawback related to ontologies [Vieira 2008], is that the tools and standards for manipulating ontologies are still immature and hard to use. In addition, using ontologies requires care since reasoning over them may impact the application performance.

3.3.3. Context in Databases and Data Integration

Almost every statement we make is imprecise and hence meaningful only if understood with reference to an underlying context [Goh 1997]. As an illustration, imagine a database which stores the following: Street('Epitacio Pessoa', 3000, 'Good'). How can these values be interpreted? When trying to figure out the sentence's meaning, some possibilities may be considered: (i) Street(name, length, conservation level), or (ii) Street(name, width, cleanliness level). This means that each database maintains its own assumptions about the data it stores in an independent way. In this sense, context may be used to specify the assumptions made in database design to understand the underlying semantics. Nevertheless, there has been little work on integrating context into distributed environments that make use of database technologies.

Distinguishing useful information from noise (i.e., not relevant) is not a trivial task, neither in a single database nor in a distributed environment. An example of *context-based data tailoring* approach is the work of Bolchini and her group [2008]. They have used context to define context-aware data views over large information systems. In their work, they propose the definition of a context-guided methodology to support the designer in identifying, for a given application scenario, the contexts and the correspondingly interesting subsets of data. Such methodology is composed by three basic elements: (i) a context model, capturing all the aspects – the so-called dimensions, that allows the implicit representation of the possible application contexts; (ii) a strategy for identifying, for each dimension, a relevant portion of the entire data schema, i.e., partial views; and (iii) a suite of operators for combining these views to derive the final view(s) associated with each context. Through this methodology, they allow the tailoring of the data, providing a personalized subset of the available information.

Context may also be used in a broader way to improve data integration processes, such as *query answering* and *schema reconciliation*. In a schema reconciliation process, usually some tasks are considered [Belian 2008]: i) the preprocessing routine that translates the schemas into a common format; ii) the schema comparison which establishes the meaning of schema elements producing inter-schema mappings; and iii) the merging and restructuring tasks which group corresponding elements to generate the integrated schema. In this process, element names can have different meanings depending on the context in which they are related. Hence, contextual elements may improve the semantic interpretation of an entity by restricting or modifying the meaning of an element according to a specific context [Souza et al. 2008]. Works dealing with conflicts resolution (i.e., semantic and/or schematic conflicts) were the first ones to use contextual information in data integration settings. Examples of these are the works of Kashyap and Sheth [1996], Goh [1997], Ram and Park [2004] and Belian [2008].

Regarding query answering, context is any information that may influence the result given to users in response to their queries [Yu et al. 2005]. A context-based query answering process in a dynamic distributed environment is usually accomplished by the following steps: (i) query submission and analysis; (ii) relevant data sources' identification; (iii) query reformulation according to semantic mappings; (iv) query execution and results' integration; and (v) result presentation. Applying context reasoning to query answering enriches the complete process as well as provides what has been called *context-sensitive queries* - those whose results depend on the context at the time of their submission [Stefanidis et al. 2005]. In this sense, when a user poses a query, all the surrounding contextual elements will be analyzed to avoid ambiguity, indicating the data that are really relevant to the user's specific situation. Besides, specific data conflicts arise mostly when query answers are assembled to produce a final result [Goh 1997]. Therefore, user profile, query model and interface, data sources' availability and semantic correspondences are examples of contextual elements that may be used to contextualize queries, providing users with more useful results.

An example of context-based query answering approach is the work of Stefanidis and his group [2005]. In Stefanidis et al. [2005], they investigate the use of context in relational database management systems by means of a preference database system that supports context-sensitive queries. In this work, context is modeled as a finite set of special-purpose attributes, called *context parameters*. Examples of context parameters are location, weather and the type of computing device in use. A *context state* is an assignment of values to context parameters. Thus, users express their preferences on specific database instances based on a single context parameter. Then, such basic preferences are combined to compute *aggregate preferences* that include more than one context parameter. The dependencies between context-dependent preferences and database relations are stored through data cubes, and the queries over them are processed using OLAP techniques, what allows for the manipulation of the captured context data at various levels of abstraction. For instance, in the case of a context parameter representing location, preferences can be expressed for example at the level of a city, the level of a country or both.

3.4. Concluding Remarks

Nowadays, dynamic distributed settings, such as PDMS, face challenging problems in dealing with the huge amount of data and the variety of its format. As a consequence, they not only need additional supports for manipulating data but also for understanding its meaning. To provide meaning and assist tasks such as query answering, semantic knowledge in the form of ontologies and context has proven to be helpful. Moreover, there is an increasing interest on combining context and ontology to define such semantics.

Ontologies, that make explicit the usually implicit data modeling assumptions regarding semantics of the data sources, play important roles in the reconciliation of semantic differences across data sources [Wache et al. 2001]. Also, carrying semantics for particular domains, ontologies are largely used for representing domain knowledge, and can be used as background knowledge in processes like ontology matching or query answering. On the other hand, context may be employed as a way to improve decision-making over heterogeneity reconciliation in data integration processes since it helps to understand the data schema semantics as well as the data content semantics. Furthermore, it can improve query answering capabilities, providing users with more meaningful answers according to the context acquired at query submission time.

It is possible to find formal and informal approaches defining ontology and context models. Both models must be represented in terms of a well-defined language, and, once such a representation is available, there ought to be well-founded methods for reasoning upon it. To this end, some formalisms such as XML-related languages and Description Logics (DLs) may be used. Among the formers, the most prominent is OWL, which has been developed by the W3C and maintains compatibility with other pre-existing languages like RDF. OWL is intended to be the standard Semantic Web ontology language. DLs are a family of logic-based knowledge representation formalisms designed to represent and reason about the knowledge of a given domain. In fact, they are the OWL theoretical underlying formalism.

Using semantics from an underlying ontology and from the gathered context might enhance query answers in a dynamic distributed environment. More precisely, query reformulation should take into account such semantics in order to produce both enriched and exact queries. A proposed solution for this problem is presented in the next chapters.

CHAPTER 4

"Strength does not come from physical capacity.
It comes from an indomitable will."
Gandhi

Using Ontologies in Data Management

Data management solutions have been continuously evolving during the last years in order to answer users' needs and face new technology challenges. To help matters, semantic knowledge in the form of ontologies has proven to be a helpful support for the techniques used for managing data [Necib and Freytag 2005; Xiao and Cruz 2006]. As such, ontologies are considered a key technology used to describe the semantics of data at different sources, helping to overcome problems of semantic interoperability and data heterogeneity, and thus assisting query answering over the distributed data sources. In our work, we use ontologies in a threefold manner: (i) as background knowledge in order to identify semantic correspondences between matching ontologies; (ii) as a mechanism to represent and store contextual information and (iii) as a means for defining Ontology-based PDMS and representing peer conceptual schemas.

In this sense, this chapter presents the way we use ontologies in our approach. To this end, Section 4.1 presents our approach for identifying semantic correspondences between ontologies; Section 4.2 presents CODI – a context ontology to represent and store contextual elements. Ontology-based PDMS are defined in Section 4.3. Finally, Section 4.4 concludes the chapter with some remarks.

4.1. Using Domain Ontologies to Identify Semantic Correspondences

In ontology-based distributed environments, several ontologies are usually developed with meaningful content overlapping among them. This means that when two ontologies overlap, they can be linked together in order to enable exchange of their underlying knowledge. Nevertheless, ontologies are developed by different people or systems, with diverse goals and design assumptions. These assumptions have the effect of creating several forms of heterogeneity between them, even between those on the same domain [Euzenat and Shvaiko 2007]. Reconciling such ontologies is still a relevant research issue, mainly in distributed settings.

The common approach for supporting ontology reconciliation in such settings is based on ontology matching techniques, what provides the definition of semantic relationships between elements belonging to the different ontologies, called in our work as *correspondences*. A simple example of correspondence is the one stating that the concept Professor in one ontology is equivalent to the concept Professeur in other ontology. Correspondences among ontologies can be used for various tasks, including ontology merging, query answering, or for navigation on the Semantic Web.

Since traditional approaches to ontology matching mainly rely on linguistic and/or structural techniques, and these techniques are not so precise, some works have considered the use of additional descriptions, called background knowledge [Sabou et al. 2006; Reynaud and Safar 2007]. The use of background knowledge, through ontologies or thesaurus, can enhance the correspondences identification by extending the ones commonly found. In fact, existing matchers have shown that concepts from two matching ontologies are rarely precisely equivalent, but rather have some semantic overlap. Thereby, finding such degree of semantic overlap (or not, in case of disjointness) becomes more useful for tasks such as query answering.

We consider that there are four main activities connected to the problem of correspondences in a dynamic distributed environment composed by ontologies [extended from Haase and Wang 2007]:

i. Identifying correspondences between the overlapping ontologies;
ii. Representing these correspondences in an appropriate formalism;
iii. Using the correspondences for a given task (e.g., query answering, data integration, ontology merging); and
iv. Maintaining correspondences according to ontologies evolution.

We focus on the i, ii and iii above mentioned issues. The correspondences' maintenance problem is out of our scope. Regarding issue i, we focus on identifying semantic correspondences between two given ontologies, taking into account a domain ontology as background knowledge. Regarding issue ii, we use the $\mathcal{ALC}$-DL notation in order to formalize the correspondences, since it is a formalism which can be instantiated and reused in real environments, according to their specific requirements. Finally, concerning issue iii, we enhance query reformulation by using semantics derived from the set of identified semantic correspondences between peers. This latter issue will be dealt with by presenting our query reformulation approach in Chapter 5.

Next section, we present our approach to deal with issues i and ii. The goal of our approach is to overcome the limitations of traditional ones by using domain ontologies as background knowledge. We go one step further as, besides the common correspondences (e.g., equivalence and subsumption), we also identify other semantic ones (e.g., disjointness and closeness), providing various and semantically-rich degrees of similarity between ontology elements.

4.1.1. Correspondences Specification

In our approach, ontologies are used as uniform conceptual representation of data sources schemas. These data sources, which are abstractly called *peers*, are grouped within the same knowledge domain (e.g., *Education* or *Health*) and an ontology describing the domain is available to be used as background knowledge. Correspondences between peers' ontologies are established to provide a common understanding of their data sources. We consider that correspondences are determined between pairs of peers which have been semantically related according to a clustering process. Ontologies belonging to such peers overlap at some semantic degree. Therefore, we define our setting as a distributed one composed by a set of peer ontologies interconnected with correspondences. For the sake of simplicity, in this chapter, we call peer ontologies by *ontologies* (O).

Correspondences are considered first-class objects, since they are stored independently from their relating ontologies. In our approach, three aspects are considered in defining correspondences:

- Directionality: A correspondence can be unidirectional or bidirectional. In the former, we specify how to map elements in a target ontology using elements from a source ontology in a way that is not commonly invertible. The latter works both ways, i.e., an element in a target ontology is expressed using elements of a source one and vice-versa. In our setting, we consider unidirectional correspondences. This means that if C is a unidirectional correspondence between two concepts i and j, C_{ij} is not the inverse of C_{ji}.
- Homogeneity/heterogeneity: homogeneous correspondences allow mapping concepts into concepts and properties into properties (attributes and relationships). Heterogeneous correspondences are the ones which provide correspondences between properties and concepts, i.e., a concept in an ontology O_i and a property in an ontology O_j or vice-versa. Such correspondences have been considered necessary to express the semantic relationships between two ontologies, when the information represented as a concept in the former is represented as a relationship in the latter, or vice versa [Ghidini and Serafini 2006]. As an example, we may consider a first

ontology O_i which has the concept Family and the property spouseIn. Family represents the set of families, and spouseIn relates a Human with the family in which s/he is one of the spouses. Another ontology O_j contains the property spouse, which represents the relationship of marriage between two Humans. When integrating O_i and O_j we may state that every family in O_i can be mapped into a married couple in O_j, or in other words, that the concept Family can be mapped into the property spouse. In our specification, correspondences may be heterogeneous, although currently in our implemented solution, they are stated as homogeneous.

- Cardinality: It may be necessary to map an element (concept or property) from one ontology to a number of different elements in another ontology. Hence, we can have 1:1 or 1:n correspondences. In our specification, we restrict the cardinality of the correspondences to be 1:1.

We say that $\{C\} = \{C_{ij}\}_{i \neq j}$ refers to the set of correspondences between a source ontology (O_i) with a target ontology (O_j). Since terminological normalization is a pre-matching step in which the initial representation of two ontologies are transformed into a common format suitable for similarity computation, we consider that both ontologies O_i and O_j have been converted to a uniform representation format. In other words, element names from O_i and O_j have been adjusted to become compatible with the element names found in the Domain Ontology.

Domain Ontologies (DO) contain concepts and properties of a particular knowledge domain. In our setting, we consider DO as reliable references that are made available on the Web. Particularly, we use them in order to bridge the conceptual differences or similarities between two overlapping ontologies. In this sense, first concepts and properties from the two matching ontologies are mapped to equivalent concepts/properties in the DO and then their semantic correspondence is inferred based on the existing semantic relationship between the DO elements. More specifically, when comparing two concepts x and y having equivalent concepts k and z (respectively in the DO), we compare k and z, and, if they are semantically related, infer that x and y are semantically related as well. Using such knowledge, we can identify what kind of semantic relationship exists between k and z and consequently specify what type of correspondence may be established between x and y (from the matching ontologies). Figure 4.1 shows an overview of our approach for specifying the semantics of the correspondences between two given ontologies O_1 and O_2. In this overview, $O_1{:}x \equiv DO{:}k$ and $O_2{:}y \equiv DO{:}z$. Since k is subsumed by z in the DO, we infer that the same relationship occurs between x and y. Then, we conclude that $O_1{:}x$ is subsumed by $O_2{:}y$, denoted by $O_1{:}x \xrightarrow{\sqsubseteq} O_2{:}y$.

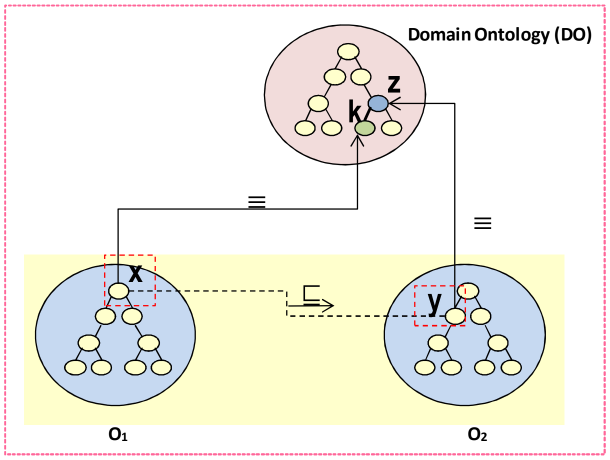


Figure 4.1 Using a DO to Specify Semantic Correspondences between Ontologies

In order to specify the correspondences, we take into account four aspects:

i. The semantic knowledge found in the DO;

ii. Whether the ontologies' concepts share super-concepts in the DO;

iii. If these super-concepts are different from the root; and

iv. The depth of concepts measured in nodes.

Next, we present the definition of semantic correspondences together with the set of rules that identify their types. The notation we use is based on Distributed Description Logics (DDL) [Borgida and Serafini 2003]. Since our approach is not concerned with proposing new algorithms for DL or DDL, we rely on existing equivalence and subsumption ones [Baader et al. 2003] as the basis for our definitions.

Definition 1 - *Semantic Correspondence*. A semantic correspondence is defined as one of the following expressions:

1. $O_1{:}x \xrightarrow{\equiv} O_2{:}y$, an *isEquivalentTo* correspondence

2. $O_1{:}x \xrightarrow{\sqsubseteq} O_2{:}y$, an *isSubConceptOf* correspondence

3. $O_1{:}x \xrightarrow{\sqsupseteq} O_2{:}y$, an *isSuperConceptOf* correspondence

4. $O_1{:}x \xrightarrow{\trianglerighteq} O_2{:}y$, an *isPartOf* correspondence

5. $O_1{:}x \xrightarrow{\trianglelefteq} O_2{:}y$, an *isWholeOf* correspondence

6. $O_1{:}x \xrightarrow{\approx} O_2{:}y$, an *isCloseTo* correspondence

7. $O_1{:}x \xrightarrow{\perp} O_2{:}y$, an *isDisjointWith* correspondence

where x and y are elements (concepts/properties) belonging to the matching ontologies O_1 and O_2.

Next, we describe each one of the existing correspondences types. To make definitions clearer, we provide examples using an illustrative scenario which is concerned with electronic devices, including computers and their components. In Figure 4.2, we present the DO, while in Figure 4.3, we depict the matching ontologies.

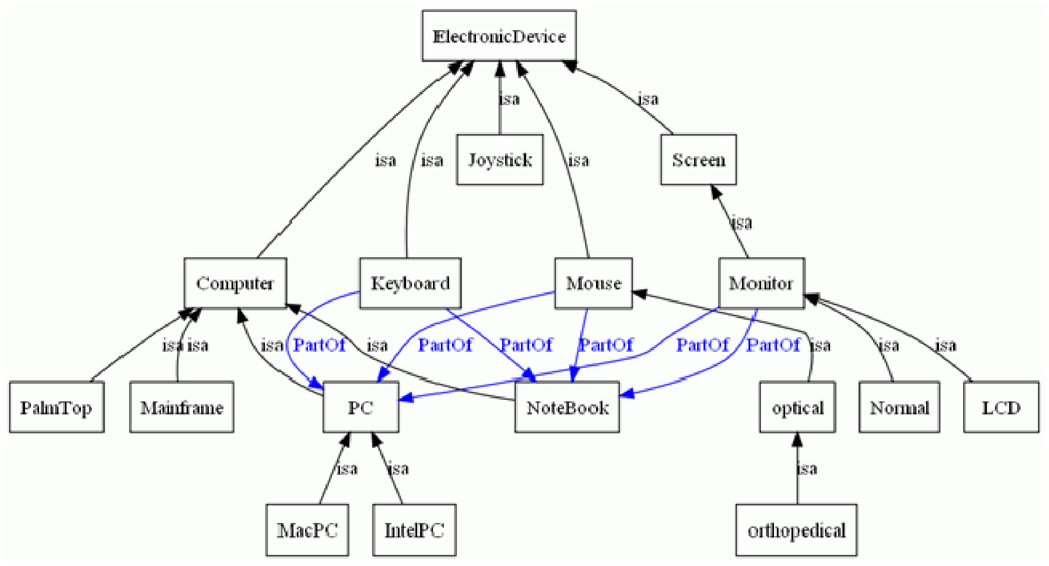


Figure 4.2 An Illustrative Domain Ontology

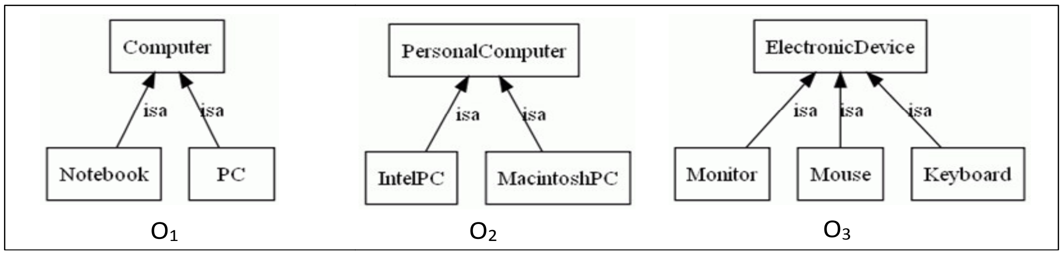


Figure 4.3 Matching Ontologies

Equivalence

The correspondence *isEquivalentTo* has already been defined in several ways [Baader et al. 2003]. In our approach, using the domain ontology, if O_1:x points to a concept/role k in the DO and O_2:y points to the same concept k, we can infer that both concepts or properties describe the same real world concept/property (See Figure 4.4). The equivalence correspondence is defined as follows.

Definition 1.1 - *isEquivalentTo* Correspondence. An element O_1:x *isEquivalentTo* O_2:y if O_1:x ≡ DO:k and O_2:y ≡ DO:k. This correspondence is represented by O_1:x $\xrightarrow{\equiv}$ O_2:y.

Considering the illustrative scenario, as an example of this correspondence identification, we have:

O_1:PC ≡ DO:PC and
O_2:PersonalComputer ≡ DO:PC

Then O_1:PC $\xrightarrow{\equiv}$ O_2:PersonalComputer

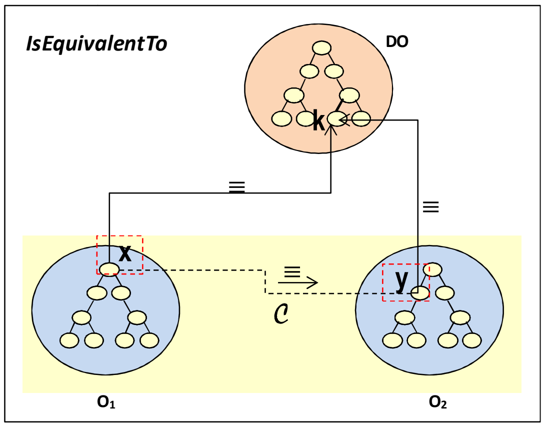


Figure 4.4 Specifying the *isEquivalentTo* Correspondence

Specialization and Generalization

The semantics of *isSubConceptOf* and *isSuperConceptOf* has also been defined in some works (e.g., [Ghidini and Serafini 2006]). In our work, we redefine both using the domain ontology, as depicted in Figure 4.5. In this sense, the *isSubConceptOf* correspondence denoted as O_1:x $\xrightarrow{\sqsubseteq}$ O_2:y states that the concept (or property) x in O_1 is less general than its related concept (or property) y in O_2. On the other hand, the *isSuperConceptOf* correspondence denoted as O_1:x $\xrightarrow{\sqsupseteq}$ O_2:y expresses the fact that x in O_1 is more general than y in O_2. We provide these definitions as follows.

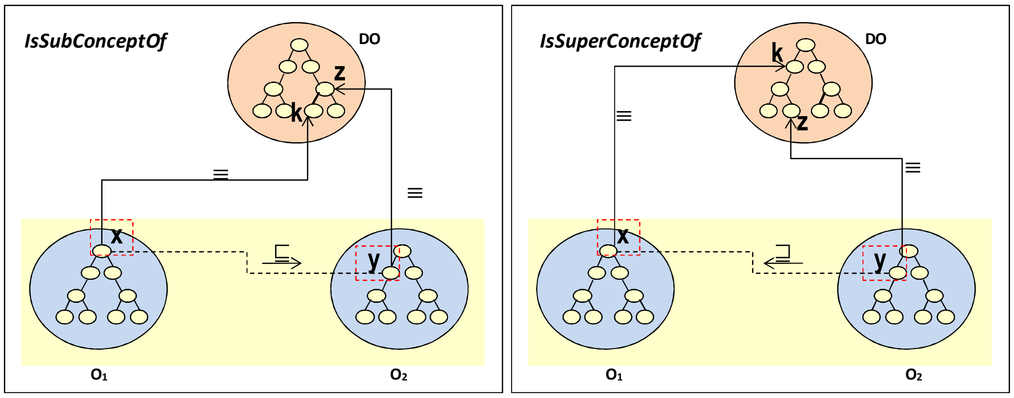


Figure 4.5 Specifying the *isSubConceptOf* and *isSuperConceptOf* Correspondences

Definition 1.2 - *isSubConceptOf* Correspondence. An element O_1:x *isSubConceptOf* O_2:y if O_1:x ≡ DO:k and O_2:y ≡ DO:z and DO:k ⊑ DO:z. Such correspondence is represented by O_1:x $\xrightarrow{\sqsubseteq}$ O_2:y.

An example of such correspondence is:

O_2:PersonalComputer ≡ DO:PC and
O_3:ElectronicDevice ≡ DO:ElectronicDevice and

DO.PC $\sqsubseteq$ DO:ElectronicDevice

Then O_2.PersonalComputer $\xrightarrow{\sqsubseteq}$ O_3.ElectronicDevice

Definition 1.3 - *isSuperConceptOf* Correspondence. An element O_1:x *isSuperConceptOf* O_2:y if O_1:x $\equiv$ DO:k and O_2:y $\equiv$ DO:z and DO:k $\sqsupseteq$ DO:z. This correspondence is represented by O_1:x $\xrightarrow{\sqsupseteq}$ O_2:y.

An example is:

O_1:Computer $\equiv$ DO:Computer and

O_2:PersonalComputer $\equiv$ DO:PC and

DO:Computer $\sqsupseteq$ DO:PC

Then O_1.Computer $\xrightarrow{\sqsupseteq}$ O_2.PersonalComputer

Using the transitivity property, we can infer that O_1.Computer $\xrightarrow{\sqsupseteq}$ O_2.IntelPC and O_1.Computer $\xrightarrow{\sqsupseteq}$ O_2.MacintoshPC.

Aggregation

In many applications of information sharing, relating aggregated objects with their components or the inverse (i.e., components with the whole) may be of key importance, particularly in query expansion. Defining *part-whole* correspondences may be used to enrich queries in order to provide users with another level of concept approximation. For instance, when querying for team, the team's participants can also be provided as additional answers.

We have defined both kinds of aggregation correspondences, as shown in Figure 4.6. The correspondence *isPartOf* states that the concept x in O_1 is a part or component of the related concept y in O_2, and the correspondence *isWholeOf* expresses the fact that x in O_1 is an aggregate of y in O_2, i.e., x in O_1 is composed by y in O_2. Both definitions are provided in the following.

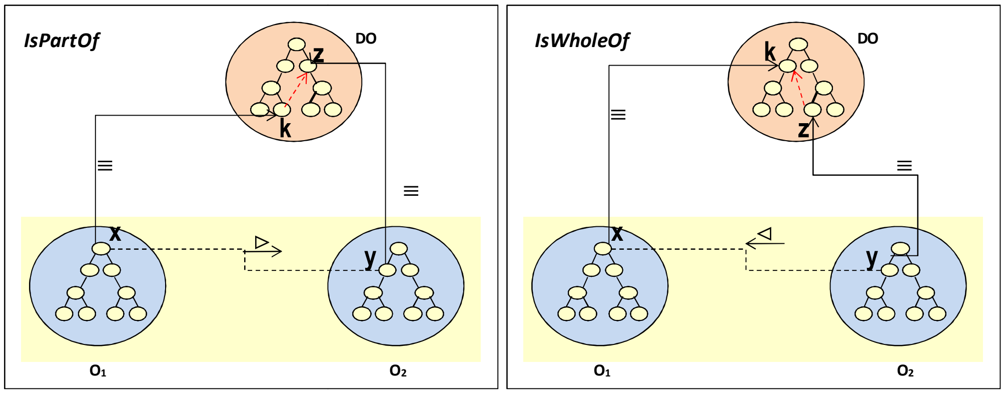


Figure 4.6 Specifying the *isPartOf* and *isWholeOf* Correspondences

Definition 1.4 - *isPartOf* Correspondence. An element O_1:x *isPartOf* O_2:y if O_1:x ≡ DO:k and O_2:y ≡ DO:z and DO:k ⊳ DO:z (isPartOf). This correspondence is represented by O_1:x $\xrightarrow{\rhd}$ O_2:y.

As an example, consider:

O_3:Keyboard ≡ DO:keyboard and
O_1:PC ≡ DO.PC and
DO:keyboard ⊳ DO:PC
Then: O_3.Keyboard $\xrightarrow{\rhd}$ O_1.PC

Definition 1.5 - *isWholeOf* Correspondence. An element O_1:x *isWholeOf* O_2:y if O_1:x ≡ DO:k and O_2:y ≡ DO:z and DO:k ⊲ DO:z (isWholeOf). This correspondence is represented by O_1:x $\xrightarrow{\lhd}$ O_2:y.

An example is in the following:

O_1:Notebook ≡ DO.Notebook and
O_3.Monitor ≡ DO.Monitor and
DO.Notebook ⊲ DO.Monitor
Then: O_1:Notebook $\xrightarrow{\lhd}$ O_3.Monitor

Closeness

In order to enrich queries and provide users with meaningful and related answers, sometimes we need to add semantically *close* concepts to the query. In our approach, two concepts of relating ontologies are *close* if they are perceived as belonging together to a common relevant context or meaning, i.e., two concepts are under the same real world concept (the same ancestor in the domain ontology).

Considering the domain ontology, two sibling concepts usually overlap in some degree. If they do not overlap, they must be explicitly made disjoint by the use of disjoint axioms. Thereby, sibling concepts which are not stated as disjoint may be close. In this sense, reflecting on some existing domain ontologies, examples of semantically close concepts organized under the same ancestor category are notebook, palmtop and mainframe which are sub-concepts of computers, or magazine and book which are sub-concepts of reading material. However, being siblings that are not disjoint is not the only sufficient condition to be set as close concepts.

Another important aspect concerns the intended meaning of the concepts that make up a domain ontology. According to the *OntoClean* methodology [Guarino and Welty 2002], discussed in Section 3.1.3, a concept must be labeled as *rigid*, *non-rigid* or *anti-rigid*, meaning the degree of essentiality such concept has in the ontology. We consider that the common ancestor of two close concepts must be labeled as *rigid*, i.e., a concept carries a rigid property if, for all its instances, such property is rigid. Such property is intended to guarantee that the common ancestor of two close concepts is an important concept in such conceptualization.

Other aspects are concerned with the depth of the common ancestor in the domain ontology and the depth of close concepts in relation to their ancestor. The former is important to guarantee that the ancestor is not a very general concept in the DO, and the latter aims to guarantee a degree of proximity of each concept in relation to its common ancestor. These aspects are indeed completely dependent on the domain ontologies' granularity and size. They are verified by the use of two thresholds: (i) *thresholdRoot* which provides a limit for the position of the common ancestor in relation to the root; and (ii) *thresholdCommonAncestor* which provides a limit for the position of each matching concept in relation to the common ancestor.

In this light, we consider that two concepts k and z are close if:

i. They share a common ancestor in the DO;
ii. This common ancestor is not the root ($\top$) and it is labeled as *rigid*;
iii. The concepts do not hold any subsumption nor disjointness relationship between themselves; and
iv. The measured depths are evaluated to true, according to the referred thresholds.

Implicitly, taking into account these conditions, we infer the semantic path between the concepts, thus, verifying the degree of closeness between them. The closeness correspondence is specified as follows (see Figure 4.7).

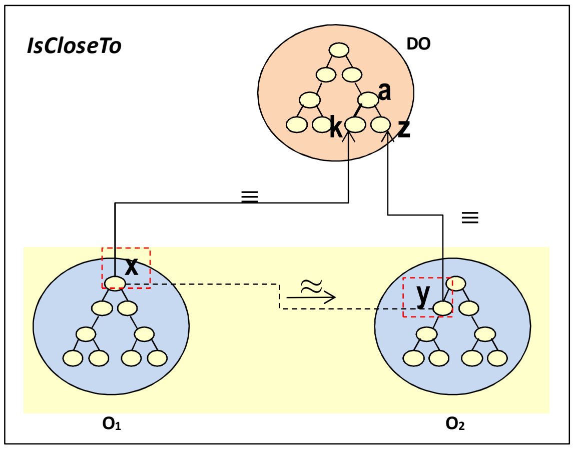


Figure 4.7 Specifying the *isCloseTo* Correspondence

Definition 1.6 - *isCloseTo* Correspondence. An element O_1:x *isCloseTo* O_2:y if (O_1:x ≡ DO:k and O_2:y ≡ DO:z) and (DO:k ⊑ DO:a and DO:z ⊑ DO:a) and DO:a ≠

$\top$ and DO:a *isRigid* and (depth(DO:a, DO:$\top$) $\geqslant$ tresholdRoot) and $\neg$(DO:k $\perp$ DO:z) and (depth(DO:k,DO:a) $\leqslant$ thresholdCommonAncestor and depth(DO:z,DO:a) $\leqslant$ thresholdCommonAncestor). This correspondence is represented by O_1:x $\overset{\approx}{\rightarrow}$ O_2:y.

Where $\neg$(DO:k $\perp$ DO:z) means that k and z are not disjoint.

Assuming that DO:Computer has been labeled as *rigid*, thresholdRoot has been set to 1, and thresholdCommonAncestor has been set to 3, we provide an example as follows.

O_1:Notebook $\equiv$ DO:Notebook and
O_2:MacintoshPC $\equiv$ DO:MacPC and
DO:Notebook $\sqsupseteq$ DO:Computer and
DO:MacPC $\sqsupseteq$ DO:Computer and
DO:Computer $\neq \top$ and
DO:Computer *isRigid* and
depth(DO:Computer, DO:$\top$) = 1 and
depth(DO.Notebook,DO.Computer) = 1 and
depth(DO.MacPC, DO.Computer) = 2
Then: O_1.Notebook $\overset{\approx}{\rightarrow}$ O_2.MacintoshPC.

Disjointness

Recently, several ontology editors have allowed to explicitly specify whether two concepts are disjoint. In description logics, two concepts are considered as disjoint if their taxonomic overlap (i.e., the set of common individuals) is empty [Volker et al. 2007]. In other words, concepts are disjoint if they cannot have any instances in common. For example, the Red wine and the White wine classes are disjoint: no wine can be simultaneously red and white. Specifying that classes are disjoint enables the system to validate the ontology better [Noy and McGuinness 2001]. Thus, if we declare the Red wine and the White wine concepts to be disjoint and later create a concept that is a subclass of both, the system can indicate that there is a modeling error. Nevertheless, it is very likely that ontology engineers sometimes forget to introduce disjointness axioms, simply because they are not aware of the fact that classes which are not explicitly declared to be disjoint will be considered as overlapping.

Considering that, we have defined the *isDisjointWith* correspondence in order to identify the strongest dissimilarity between peer ontology elements. In terms of matching, this correspondence is not so important, but regarding query answering and query reformulation, it is essential, mainly when we have negation over concepts in queries. Thus, the correspondence *isDisjointWith* states that two concepts O_1:x and O_2:y are disjoint if their DO corresponding concepts k and z, respectively, are disjoint, i.e., in the domain ontology k and z have been defined as disjoint. This means that x in O_1 does not overlap with y in O_2. Such type of correspondence is depicted in Figure 4.8 and defined as follows.

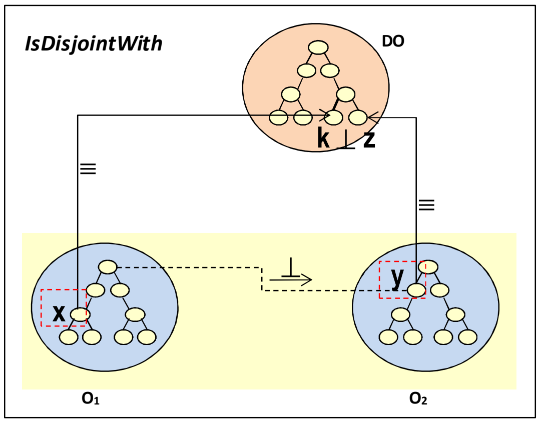


Figure 4.8 Specifying the *isDisjointWith* Correspondence

Definition 1.7 - *isDisjointWith* Correspondence. An element O_1:x *isDisjointWith* O_2:y if O_1:x ≡ DO:k and O_2:y ≡ DO:z and DO:k ⊥ DO:z. This is represented by O_1:x $\xrightarrow{\perp}$ O_2:y.

As an illustration of disjointness correspondence, consider now the domain ontology depicted in Figure 4.9. Such ontology concerns *organisms*, such as animals, minerals and plants. In this DO, disjoint axioms have been stated between some elements, including Animal ⊥ Mineral, Animal ⊥ Plant, Bird ⊥ Mammal, Mammal ⊥ Amphibian, and Bird ⊥ Amphibian. Using the transitivity property, for example, we can also infer that Animal ⊥ Salt, and Animal ⊥ MineralWater.

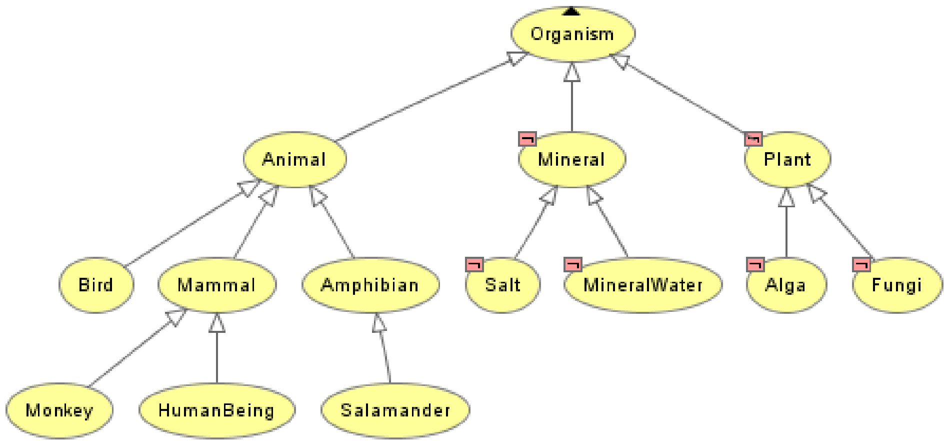


Figure 4.9 A Domain Ontology about Organisms

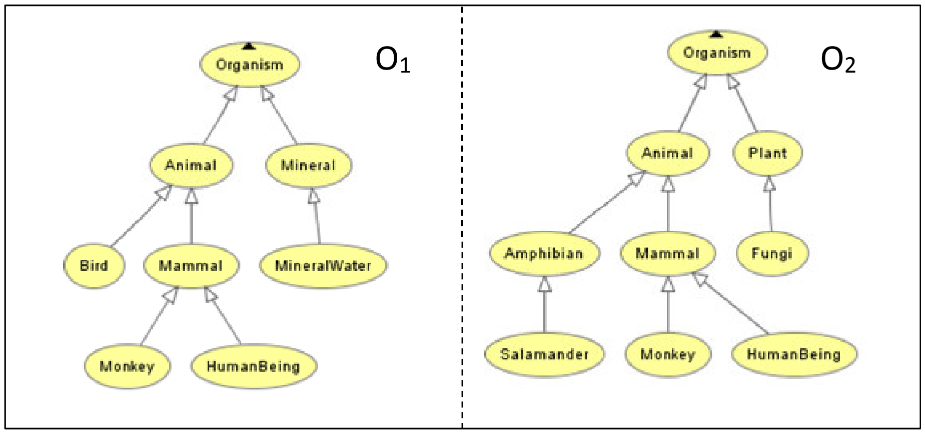


Figure 4.10 Matching Ontologies

Suppose that the ontologies shortly presented in Figure 4.10 are to be semantically matched. Considering the disjoint axioms, we can infer the following disjoint correspondence:

O_1:Animal $\equiv$ DO:Animal and
O_2:Plant $\equiv$ DO:Plant and
DO:Animal $\perp$ DO.Plant
Then O_1:Animal $\xrightarrow{\perp}$ O_2:Plant
In the same way, other disjoint correspondences can be identified, including:

O_1:Mineral $\xrightarrow{\perp}$ O_2:Plant
O_1:Bird $\xrightarrow{\perp}$ O_2:Mammal
O_1:Mammal $\xrightarrow{\perp}$ O_2:Amphibian

4.1.2. A More Complete Example

We demonstrate our approach more accurately using a real and practical example. In this current scenario, we consider a setting composed by two peers P_1 and P_2. The peers store and share data about conferences whose works (e.g., papers) are submitted electronically. Each peer is described by an ontology – O_1 and O_2, as shown in Figure 4.11. In addition, we have considered as background knowledge a Domain Ontology (DO) depicted in Figure 4.12.

In order to identify the semantic correspondences between O_1 and O_2, first, we found out the equivalences between concepts of O_1 and concepts in the DO, and the equivalences between concepts of O_2 with their related ones in the DO. Then, the set of described rules was applied. As a result, the set of semantic correspondences between O_1 and O_2 was identified. We present this resulting set in Table 4.1.

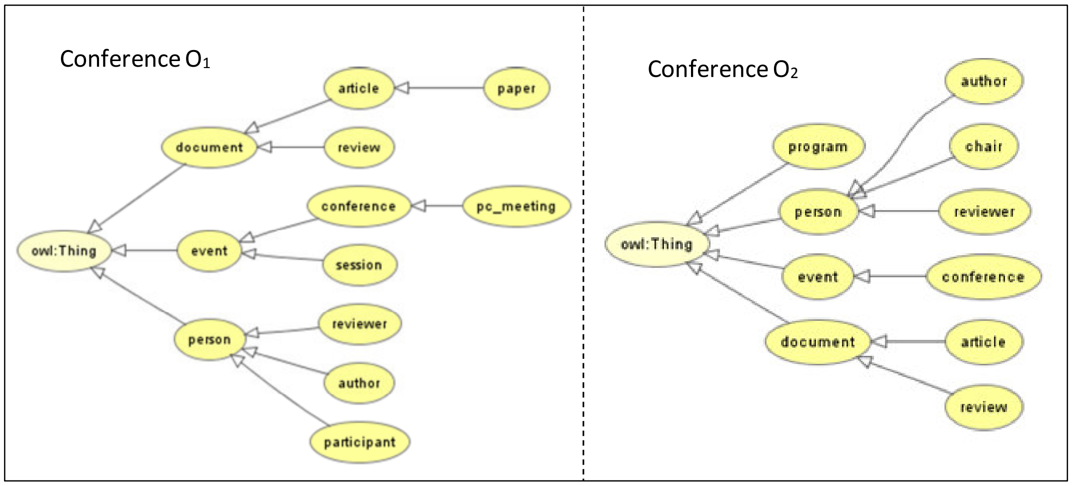


Figure 4.11 Excerpts from Conference Ontologies O_1 and O_2

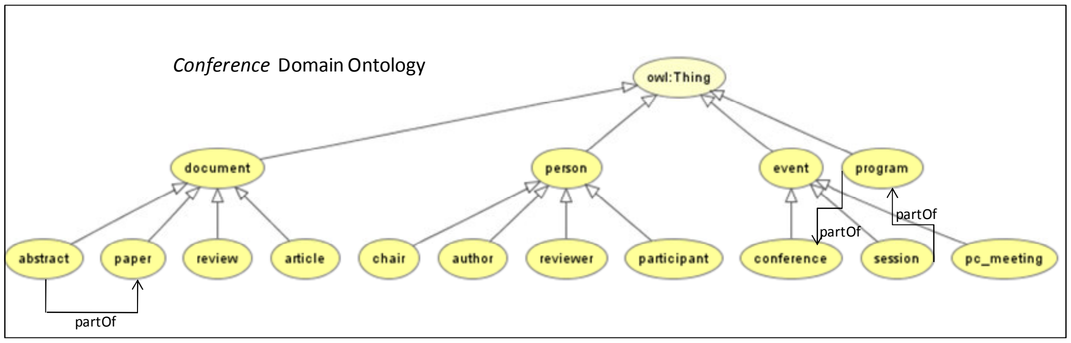


Figure 4.12 Excerpt from Conference Domain Ontology

Table 4.1. Semantic correspondences between O_1 and O_2

O_1 Concept	Correspondence Type	O_2 Concept
article	isDisjointWith	review
article	isEquivalentTo	article
article	isSubConceptOf	document
author	isCloseTo	chair
author	isCloseTo	reviewer
author	isEquivalentTo	author
author	isSubConceptOf	person
conference	isEquivalentTo	conference
conference	isSubConceptOf	event
conference	isWholeOf	program
document	isDisjointWith	event
document	isDisjointWith	person
document	isDisjointWith	program
document	isEquivalentTo	document
document	isSuperConceptOf	article
document	isSuperConceptOf	review
event	isDisjointWith	document
event	isDisjointWith	person

O_1 Concept	Correspondence Type	O_2 Concept
event	isDisjointWith	program
event	isEquivalentTo	event
event	isSuperConceptOf	conference
paper	**isCloseTo**	**article**
paper	**isCloseTo**	**review**
paper	**isSubConceptOf**	**document**
participant	**isCloseTo**	**author**
participant	**isCloseTo**	**chair**
participant	**isCloseTo**	**reviewer**
participant	**isSubConceptOf**	**person**
pc_meeting	**isDisjointWith**	**conference**
pc_meeting	**isSubConceptOf**	**event**
person	isDisjointWith	document
person	isDisjointWith	event
person	isDisjointWith	program
person	isEquivalentTo	person
person	isSuperConceptOf	author
person	isSuperConceptOf	chair
person	isSuperConceptOf	reviewer
review	isDisjointWith	article
review	isEquivalentTo	review
review	isSubConceptOf	document
reviewer	isCloseTo	author
reviewer	isCloseTo	chair
reviewer	isEquivalentTo	reviewer
reviewer	isSubConceptOf	person
session	**isDisjointWith**	**conference**
session	**isPartOf**	**program**
session	**isSubConceptOf**	**event**

In this resulting set, we can see, for instance, the equivalence correspondence between conference in O_1 and O_2. Equivalence is an example of a commonly identified correspondence type in traditional ontology matching approaches. On the other hand, we can see that, taking into account the semantics underlying the DO, we can identify other unusual correspondences. For example, participant has been identified as *close* to author, chair and reviewer; person has been identified as *superconcept* of author, chair and reviewer; review as *disjoint* with article and session as *part* of program.

Particularly, analyzing some identified correspondences in Table 4.1, we verify that some concepts in O_1 do not have corresponding equivalent ones in O_2. The concepts are: paper, participant, pc-meeting and session (they are highlighted in Table 4.1). For these cases, the unusual identified correspondences make a difference since they provide levels of semantic relationship which somehow approximate the concepts with other ones in O_2. Thereby, the existence of these semantic correspondences avoids producing an empty reformulation and further enhances the query reformulation process with some kind of enrichment. In Chapter 5, we will provide details regarding how such strategy is accomplished.

Therefore, we give evidence that the use of background knowledge really allows producing semantically richer correspondences between two ontologies, providing different degrees of semantic overlapping between them. Using such background knowledge, we can obtain kinds of correspondences which would not be possible, if we have considered only syntactic or linguistic criteria. Moreover, the semantics underlying these correspondences may effectively contribute to enhance query reformulation, and query answering as a whole, providing an enrichment of query original terms and, consequently, a set of resulting expanded answers to users.

4.1.3. Comparing Existing Approaches with Ours

A few semantic-based approaches have considered the use of background knowledge as a way to improve the determination of correspondences between two ontologies. Aleksovski and his group [Aleksovski et al. 2006] present a matching case where the source and the target ontology are of poor semantics (flat lists). They use the DICE ontology as background knowledge to provide descriptions of the properties of the concepts involved. The work described by Reynaud and Safar [2007] makes use of WordNet and implements a system named *TaxoMap*. This system performs a two-step process: a sub-tree is first extracted from WordNet, corresponding to the senses assumed to be relevant to the domain of the involved ontologies. Second, mappings are identified in this sub-tree, and the correspondences between the ontologies are identified. The work of Sabou et al. [2006], differently, uses online available ontologies as background knowledge. The idea is that these ontologies can be selected dynamically (e.g., using Swoogle[12]), thus circumventing the need for an a priori, manual ontology selection. *S-Match* is a semantic-based matching tool [Giunchiglia et al. 2004] which takes two trees, and for any pair of nodes from the two trees, it computes the strongest semantic relation holding between the concepts of the two nodes. For this, it uses relations between synsets in WordNet, and the structure of the tree. *CTXMatch* [Serafini et al. 2006] is an algorithm for discovering semantic mappings across hierarchical classifications (HCs) using logical deduction. It takes two inputs and, for each pair of concepts, returns their semantic relation.

Like our approach, most of the mentioned works use some kind of background knowledge in order to figure out correspondences between ontologies, excepting CTXMatch. However, the correspondences are usually restricted to equivalence

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

(CTXMatch consider subsumption and disjointness). We go one step further in our process as we also identify other types of semantic correspondences (e.g., closeness and aggregation), providing various and semantically-rich degrees of similarity between ontology elements. Moreover, to the best of our knowledge, closeness is a type of semantic correspondence that is not found in any related work.

4.1.4. Considerations

In our approach, since the two ontologies being mapped may be defined at different levels of granularity, our correspondence identification technique deals with a degree of flexibility capable of accommodating a variety of scenarios. This means that our approach is able to match two ontologies with different levels of details, i.e., they may differ in terms of size, partition of concepts or conceptual organization.

Another important remark is that our correspondences identification approach is part of a general semantic matching process which makes use of the domain ontology to complement linguistic and structural matching techniques. In this matching process, besides identifying semantic correspondences, the objective is also to calculate the overall similarity between the two matching ontologies. To this latter end, the semantic correspondences are used in conjunction with linguistic and structural correspondences to produce a more accurate semantic similarity measure (SSM) between the ontologies. The SSM is used to identify semantically related peers in our working PDMS (which will be described in Section 6.1) to cluster them and enhance query answering. The strategy for calculating the SSM is being object of study in another work [Pires 2009].

4.2. Using an Ontology to Represent Context

Considering a dynamic distributed environment, the semantics and control information surrounding the running processes (e.g., queries) are rather important to produce results with relevance according to users' needs and environment's capabilities. These may be obtained using contextual information, as discussed in Chapter 3.

We are able to understand context when identifying how humans use it in practice. Humans seem to be able to build complex contexts instinctively [Mills and Goossenaerts 2005]: first context is recognized and understood; then the relevant set of items (e.g., location, interests) required to deal with that context is automatically assembled. In our work, we define context as follows.

Definition 2 - *Context*. *Context* is a set of elements surrounding a domain entity of interest which are considered relevant in a specific situation during some time interval.

The domain entity of interest may be a person, a procedure, a file, a set of data or even a semantic correspondence. Furthermore, we use the term contextual element (CE) referring to pieces of data, information or knowledge that can be used to define the *Context*, in accordance with the definition provided by Vieira [2008].

When a user poses a query, all the surrounding contextual elements will be analyzed in order to denote the data that is relevant to the user's specific situation. According to the query and user context, the reformulated queries may be enriched, i.e., relevant concepts or properties may be added in order to obtain more complete reformulations, and, consequently, expanded query answers. User preferences, peers' availability, or even semantic correspondences between peers' ontologies are examples of contextual elements that are used to execute more contextualized queries in such a way that users will be provided with more relevant results.

In order to store and use context, an important issue is how to represent its elements, as discussed in Section 3.3.4. A challenge to be faced is the fact that there is not a standard model for representing it yet. Context ontologies have been considered an interesting approach because they enable sharing and reusability and may be used by different reasoning mechanisms [Wang et al. 2004; Souza et al. 2006]. Hence, in this work, we have designed an ontology, named CODI, to represent and store contextual information [Souza et al. 2008]. This ontology aims to assist the common tasks of a generic data integration process such as query answering or schema integration [Belian 2008]. We present CODI in the following.

4.2.1. CODI – A Context Ontology for Data Integration

CODI (**C**ontextual **O**ntology for **D**ata **I**ntegration) is an ontology for representing context according to some Data Integration (DI) and PDMS issues discussed in Chapter 2 [Souza et al. 2008]. In order to establish the relevant contextual elements (CEs), at first, we have identified the domain entities that we needed to work with. A domain entity is anything in the real world that is relevant to describe the domain (e.g., data sources, users and applications) [Vieira 2008]. In our work, we consider that CEs are used to characterize a given domain entity. Therefore, we determined six main domain entities around which we consider the CEs: *user*, *environment*, *data*, *procedure*, *association* and *application*. To figure out these domain entities and their related CEs, our approach has been guided by a participatory and incremental design methodology. The ontology was developed during a series of face-to-face meetings between DI and PDMS experts who are concerned with issues such as schema reconciling, query answering, connectivity and reasoning. Furthermore, we have also examined systematically in the literature some DI and PDMS systems and related problems. As a result, we draw the domain entities' concepts, their properties and more specifically the related contextual elements that would be relevant to deal with.

We present the domain entities' taxonomy as well as some contextual elements relevant to them in Figure 4.13. CODI is indeed a conjunction of those domain entities and the CEs which are related to them. More precisely, next we describe each one of the domain entities with their CEs.

- *User.* One of the key factors for accurate access to information is users' context. The CEs that make up users' context are concerned with their profile (interests, role, group), location, region, query interface type and preferences concerning the way they desire query answering and query reformulation (Figure 4.14). For instance, according to the user's query interface type, the system may define the way query answers result will be presented.

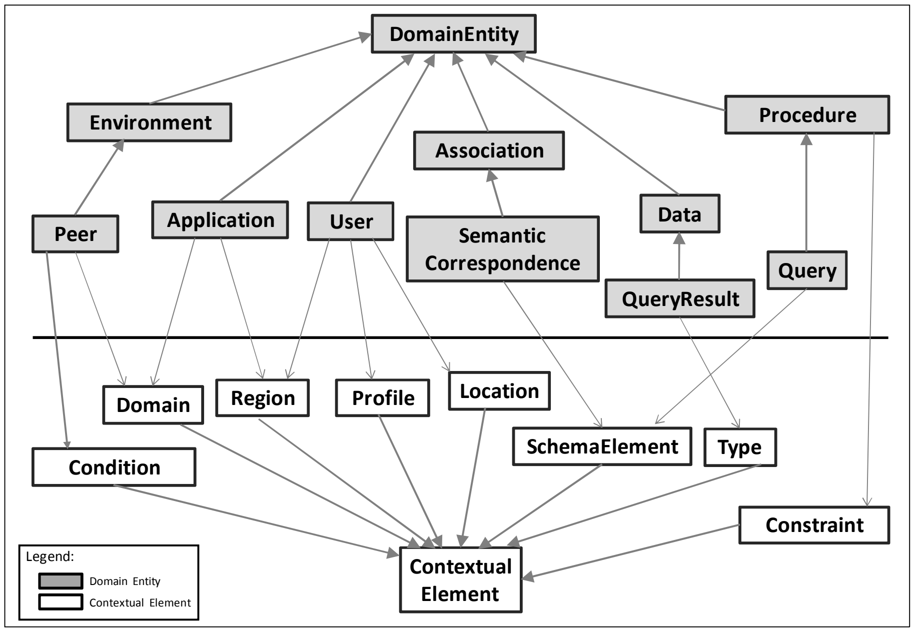


Figure 4.13 The Domain Entities' taxonomy and some CODI's CEs

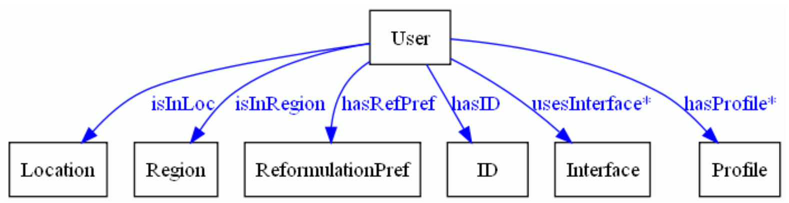


Figure 4.14 CEs for the User domain entity

- *Association.* Associations are important to characterize relationships between elements. In our approach, two kinds of associations are dealt with: mapping expressions and semantic correspondences (Figure 4.15). The formers are used in a data integration system (or in a PDMS mediation-level) in order to allow query reformulation and answers integration (mapping expressions will be better explained in Section 6.1.3). The latter ones are rather essential to query reformulation between a pair of peers, considering any distributed query answering environment.

- *Procedure*. A procedure is an ordered collection of actions [Brézillon 2003]. In our setting, a procedure is mainly characterized by queries and their execution processes (Figure 4.16). A procedure has CEs including constraint and goal, and it is usually composed by steps. In particular, besides the inherited Procedure's CEs, a query has its own CEs, such as its model, the schema elements which are necessary to work with and the operators which are to be executed. Those CEs will be acquired at query formulation time.

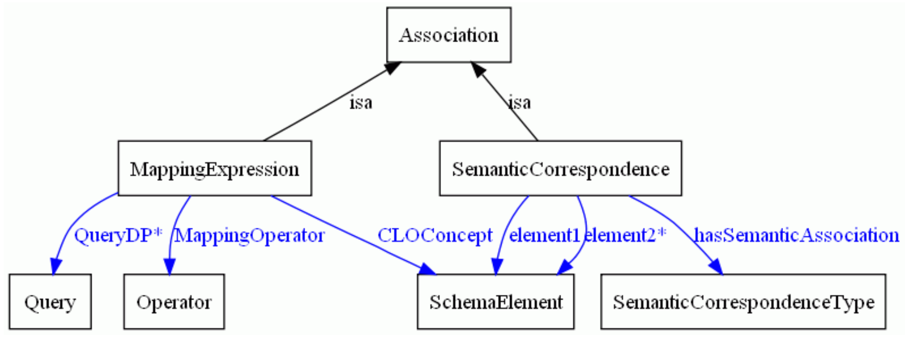


Figure 4.15 CEs for the Association domain entity

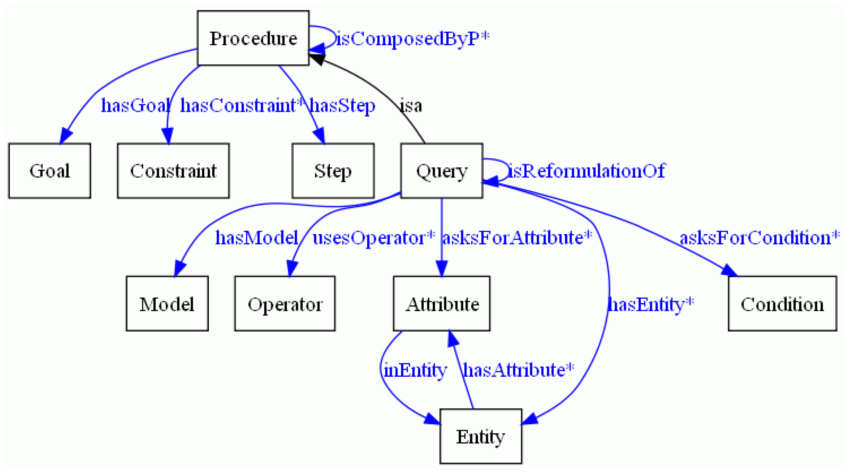


Figure 4.16 CEs for the Procedure domain entity

- *Application*. Each application has its particular features (Figure 4.17). Thereby, the use of information may vary regarding to different levels of granularity, vocabularies and/or scopes of a domain. An important CE to application regards its domain. Each domain has a vocabulary, usually represented by a domain ontology and its specific terms.

- *Data*. Data CEs are classified into Schema Element Content and Query Result (Figure 4.18). A Schema element content is related to its schema element. Schema element constitutes one of the main concepts both to query answering and schema integration, since it is possible to infer semantic associations from its meaning (achieved when identifying its corresponding concept in the domain ontology). The Query Result represents results of individual queries as well as the final result obtained from the integration of several individual query results.

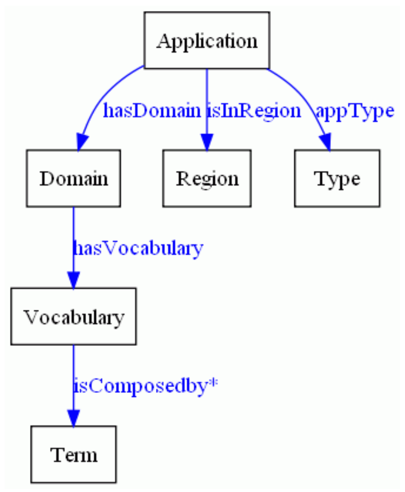


Figure 4.17 CEs for the Application domain entity

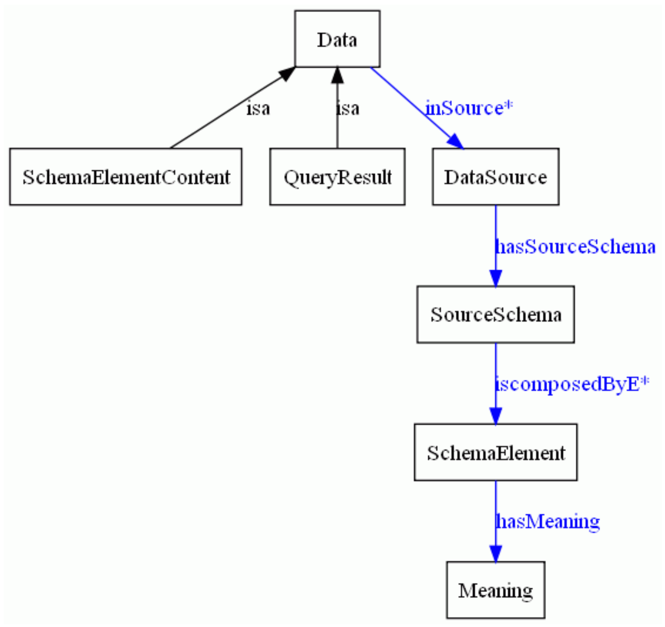


Figure 4.18 CEs for the Data domain entity

- *Environment*. Concerns the environment where the user interacts and the application is executed. In CODI, it may be a Data Integration System or a

PDMS, as shown in Figure 4.19. In both cases, we are dealing with dynamic and autonomous data sources that may join and leave the network at any time. Environment CEs must be acquired on the fly (e.g., data source availability). In this sense, Data Integration Systems, PDMS, data sources, peer and source schemas are the domain entities from which the CEs will be acquired. In general, the main environment CEs are: Type, Region, Platform and Condition. Depending on the system, other specific elements may be added or refined.

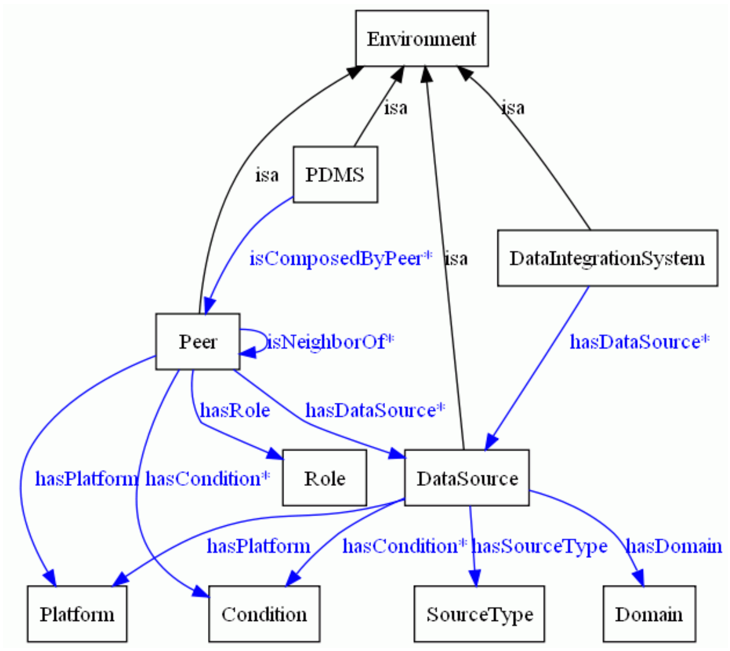


Figure 4.19 CEs for the Environment domain entity

Contextual Elements can either be explicit or implicit. An explicit CE is obtained from static sources, such as a profile. An implicit one is perceived in the dynamic environment or is derived through some reasoning process. For example, considering a geospatial application, a spatial relationship (e.g., touch, cross, distance [Egenhofer 1991]) is inferred through the analysis of two objects locations. Still, the scale a user is working with may be identified through his/her application parameters. Another illustration concerns the presentation of query results. A query's result set may contain different data representations, e.g., different unit formats that are used in the distributed data sources. Thus, depending on the context of query submission, a specific unit may be chosen and a conversion and merging process may be performed automatically.

4.2.2. CODI in Practice

In this section, we present an example of query answering in order to illustrate the applicability of CODI. To this end, we consider a geospatial application concerning the *Brazilian Hydrographic System* which has been developed in a PDMS environment. For the sake of simplicity, we only consider two peers A and B, which store geospatial data sources with the ontologies representing their schemas, as depicted in Figure 4.20. Peer A is at scale of 1:1000’000, while peer B is more detailed and is at scale of 1:250’000. Peer A contains three concepts – Lake, StreamofWater and Town which inherit some characteristics from Geographical_entity. The three concepts have a geometry attribute. Peer B contains Lake, River and City which are sub-concepts of Basic_geo_entity. These concepts have a shape attribute.

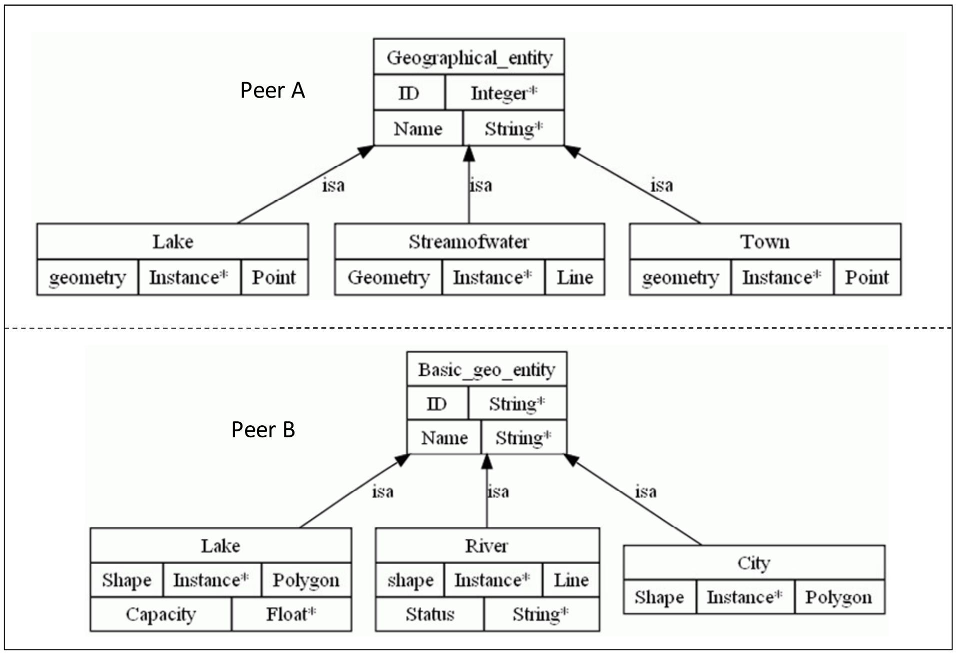


Figure 4.20 Ontologies for Peers A and B

In this scenario, the ontologies are not normalized (i.e., neither terminologically nor syntactically normalized), thus we consider some conflicts which arise due to the heterogeneity of the peers. The semantic conflicts related to schema level are: (i) different entity names – Geographical_entity vs. Basic_geo_entity, StreamofWater vs. River and Town vs. City; (ii) different attribute names – geometry vs. shape; and also different data types – integer vs. string (GID) and point vs. polygon (lake, and town and city). These conflicts are resolved in schema reconciling time, when correspondences are identified. Other relevant conflicts (found in query answering) are the instance level ones. Here we have different scales 1:1000’000 (Peer A) vs. 1:250’000 (Peer B) and the multi-representation problem, since lake is represented by a point in Peer A and by a polygon in Peer B. For simplicity, both peers are considered to be vector.

We present CODI's usage for a context-based query execution process performed by the following steps: query submission, query analysis, relevant peers' establishment, query reformulation, query execution and answers integration, and result presentation. We provide views of CODI's instantiation which have been produced using OntoViz[13]. In this format, instances are associated with their concepts through the *io* relationship and subtypes are associated with their supertypes through the *isa* relationship. The diagrams presented next are fragments from the overall ontology, and do not show neither the whole class hierarchy nor the complete set of instances.

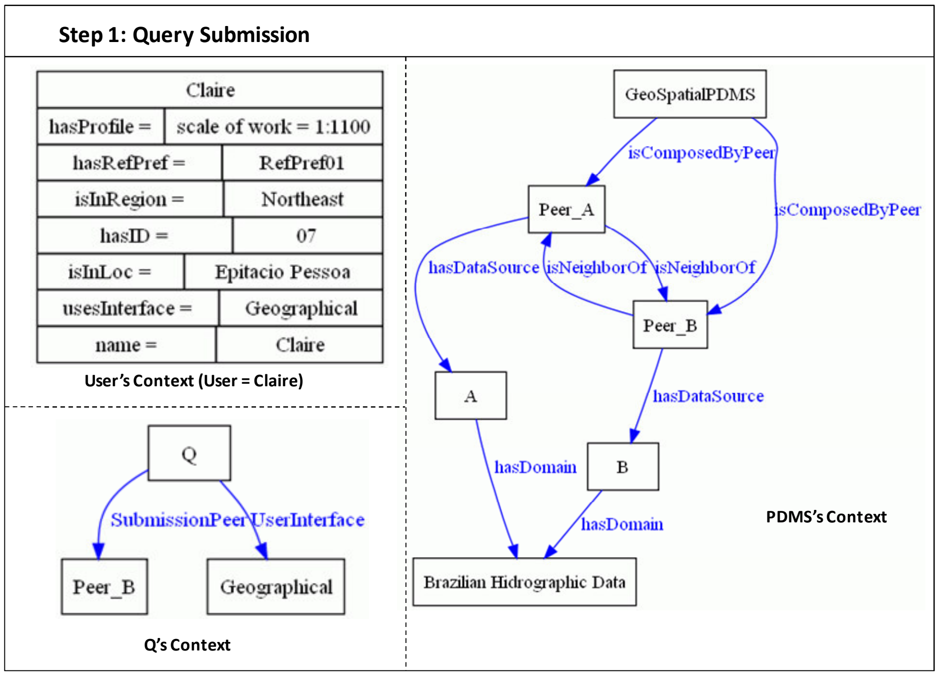
Assuming that the correspondences between the peer ontologies have already been generated, we focus on executing a given query. Suppose that a user poses the following spatial query Q: "SELECT R.Name, C.Name FROM River R, City C WHERE Cross(R.Shape,C.Shape)=1;". The topological spatial operator Cross(geometry$_1$, geometry$_2$) is a Boolean operation which returns true if a geometry$_1$ intersects with another geometry$_2$ [Egenhofer 1991]. Thus, Q's submission is done in Peer B and means: "*For all the rivers, find the cities through which they pass*".

At submission time (Step 1), context concerned with the user, the query and the environment are acquired or perceived as depicted in Figure 4.21. In this case, the user profile, her preferences, location and the kind of interface she is using are CEs that are gathered. Also, information about the environment, i.e., the PDMS, such as the composing peers and data sources as well as their domain are important information that will be dealt with when the relevant peers are set. To this end, we have to know for example which peers are available, if they have a common knowledge domain and the existing elements in each peer's schema. Besides, as context of the query, it is observed where it has been submitted and what kind of interface has been used.

In Step 2, the query is completely analyzed (Figure 4.22). The required entities, spatial operators, attributes, constraints and conditions are gathered to identify the semantics of the query. As a result, this semantics will be taken into account to verify which peers are relevant for such query and how it can be reformulated in these peers.

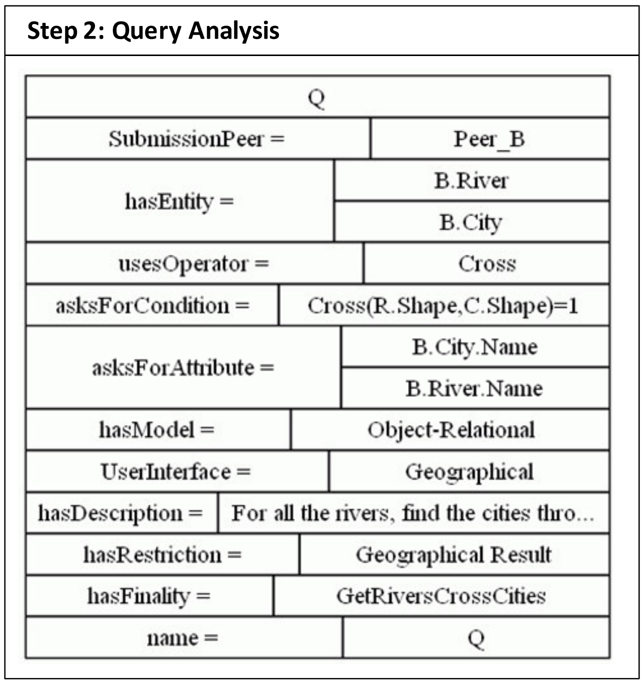
Next, in Step 3, the peers that are considered relevant (in our example, Peer A) are also observed and their context acquired and used (Figure 4.23). For instance, we have to see if such peers are available for query reformulation and if they can execute the spatial operator that has been required, since not all DBMS are able to execute properly the set of existing spatial operators.

13 a Protégé plug-in



Claire	
hasProfile =	scale of work = 1:1100
hasRefPref =	RefPref01
isInRegion =	Northeast
hasID =	07
isInLoc =	Epitacio Pessoa
usesInterface =	Geographical
name =	Claire

Figure 4.21 CEs at Query Submission Time


Step 2: Query Analysis

Q	
SubmissionPeer =	Peer_B
hasEntity =	B.River
	B.City
usesOperator =	Cross
asksForCondition =	Cross(R.Shape,C.Shape)=1
asksForAttribute =	B.City.Name
	B.River.Name
hasModel =	Object-Relational
UserInterface =	Geographical
hasDescription =	For all the rivers, find the cities thro...
hasRestriction =	Geographical Result
hasFinality =	GetRiversCrossCities
name =	Q

Figure 4.22 CEs at Query Analysis Time

Next step is reformulating query Q to a reformulation that is compatible with each relevant peers' schemas. In this example, Peer A is relevant, so the process takes into account the correspondences between Peer B and Peer A and reformulates Q into another query QRef. Since we have not presented our complete query reformulation approach yet (Chapter 5), we restrict this current example to only equivalence correspondences. Thus, Figure 4.24 depicts some correspondences which, for us, are treated as contextual information and are used to allow query reformulation. Figure 4.25 presents the context of the reformulated query QRef in Peer A.

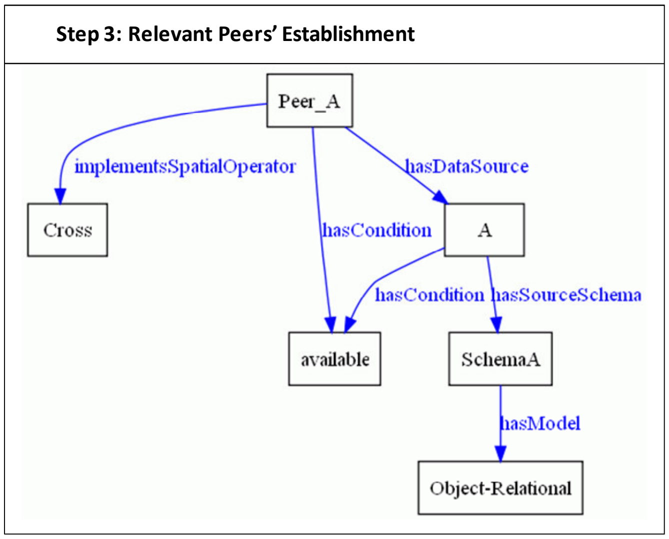
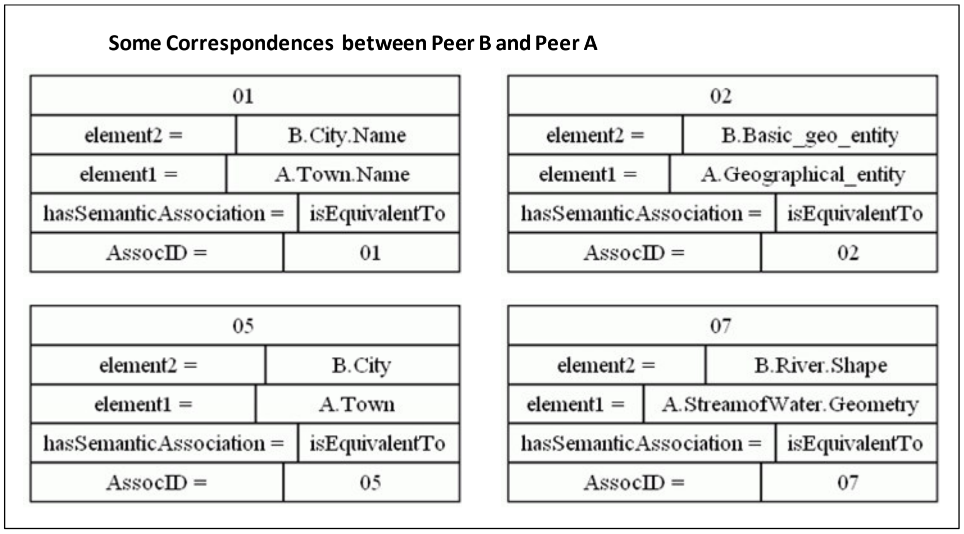


Figure 4.23 CEs at Relevant Peer's Establishment Time


Some Correspondences between Peer B and Peer A

01	
element2 =	B.City.Name
element1 =	A.Town.Name
hasSemanticAssociation =	isEquivalentTo
AssocID =	01

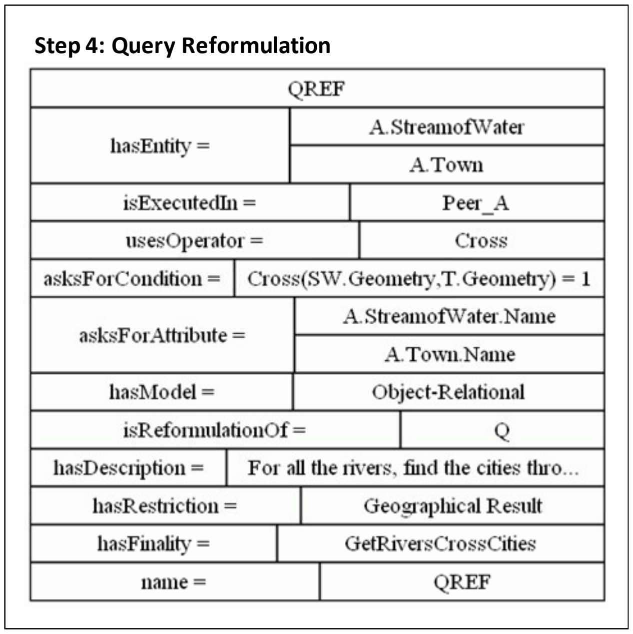
02	
element2 =	B.Basic_geo_entity
element1 =	A.Geographical_entity
hasSemanticAssociation =	isEquivalentTo
AssocID =	02

05	
element2 =	B.City
element1 =	A.Town
hasSemanticAssociation =	isEquivalentTo
AssocID =	05

07	
element2 =	B.River.Shape
element1 =	A.StreamofWater.Geometry
hasSemanticAssociation =	isEquivalentTo
AssocID =	07

Figure 4.24 Some Correspondences between Ontologies B and A

It is important to note that, in this example, query Q will be executed both in Peer B (submission peer) and in Peer A (through a reformulation). In Step 5, when the executed queries results are assembled to produce the final answer, the system analyzes other CEs such as multi-representation and scales difference. Considering that the formulating scale is about 1:100'000, this means that the user is working with a more detailed view of the themes. Thus the graphical result will be taken from Peer B whose scale of origin is closer and whose City's geometric representation (polygon) is more adequate to that level of detail. Therefore, since the user interface is able to present geographical results, the final result (Step 6) will be depicted to the user both graphically and textually (e.g., in the map and in a table format). Sometimes, in a PDMS, the final result may be produced from the answers obtained in several peers if they return complementary information, for example, when some attributes are present in one peer but are absent in another.



Step 4: Query Reformulation

QREF	
hasEntity =	A.StreamofWater
	A.Town
isExecutedIn =	Peer_A
usesOperator =	Cross
asksForCondition =	Cross(SW.Geometry,T.Geometry) = 1
asksForAttribute =	A.StreamofWater.Name
	A.Town.Name
hasModel =	Object-Relational
isReformulationOf =	Q
hasDescription =	For all the rivers, find the cities thro...
hasRestriction =	Geographical Result
hasFinality =	GetRiversCrossCities
name =	QREF

Figure 4.25 CEs at Query Reformulation Time

This is a brief description of how the use of CODI can help to enhance query answering. All information from the geospatial integration world that is to be reasoned over may be dealt with as context. Consequently, from explicit CEs, gathered from the peers, from the correspondences and from the query formulation, the system can infer and derive other implicit CEs. Since the environment (PDMS) is highly dynamic and, for each submitted query, the whole query execution process instantiation may change, the context around the query (its semantics), the peers (availability), correspondences (may be of different types, as our semantic correspondences) and the user (preferences) are essential information that have to be dealt with. In our work, such information is treated as context.

By using context, the system is able to adapt and react to different users' queries and needs. Without context, query answering would be limited by not dealing with some information that can just be acquired on the fly. As an illustration, in our example, the kind of the interface where the query has been submitted and the working scale can only be acquired in such given time. Another example concerns the user preferences: not all user preferences are relevant all the time, and only those that are semantically close to the current query should be used, disregarding those ones that are out of context. We can think in the same way for the other domain entities: environment, application, data, procedure and associations.

4.2.3. Considerations

CODI aims to structure entities and their CEs in such a way that they may be used for diverse DI and PDMS processes. The idea is that CODI may be used by developers to identify, model and represent contextual information in their applications.

Our query reformulation approach uses the concept of context as a way to enhance the overall process. To analyze the semantics around a submitted query, we propose to take into account the context of the user, of the query and of the environment. The users' context is acquired through a set of contextual variables allowing them to express their preferences regarding the reformulation process. These variables express the kind of approximation the user wants in order to guide the reformulation process, e.g., they will state that the user wants answers that are more general than the concepts of the initial query. The context of the query is derived from the corresponding submitted query and will consist of concepts and operators. Finally, the context of the environment will be captured using some parameters describing the way the queries will be routed among the peers. The use of context in our query reformulation approach will be described in Section 5.3.

4.3. Using Ontologies to Define OPDMS

Data management in dynamic distributed environments such as PDMS is a challenging and difficult problem considering the excessive number of peers, their autonomous nature and the heterogeneity of their schemas. PDMS perform their services over data from existing heterogeneous sources. As a result, they must be able to deal with different categories of heterogeneity: (i) structural heterogeneity, involving different data models; (ii) syntactical heterogeneity, concerning to different languages and data representations; (iii) system heterogeneity, related to hardware and operating systems; and (iv) semantic heterogeneity, involving different concepts and their interpretations [Wache et al. 2001]. As discussed in Section 3.1, the provision of ontologies enables PDMS interoperability at different levels of abstraction.

Xiao [2006] has introduced the concept of OPDMS through two important issues: (i) ontologies are used in local sources as a uniform conceptual metadata representation; and (ii) ontology mappings are established between peers to allow query answering. We argue that ontologies may be used in a broader way to enhance PDMS services. Moreover, ontologies may be used as a way to enrich PDMS services and provide users with more complete results. Considering that, in this work, we propose an extension to the OPDMS description [Pires et al. 2008], as follows.

Definition 3 - ***OPDMS***. An OPDMS is a PDMS which is conceived for supporting dynamic ontology-based knowledge sharing, and this knowledge must be employed to improve its services.

Essentially, an OPDMS, as a typical PDMS, keeps the properties of all PDMS such as autonomy, flexibility, single global schema absence, data location and query answering. Nevertheless, an OPDMS addresses data management issues mostly using ontologies. Furthermore, we consider that an OPDMS is a semantic-based PDMS as well, since it deals with semantic issues through knowledge obtained from ontologies.

Based on our analysis of the state-of-the-art on PDMS, we have identified a set of high-level requirements that an OPDMS should fulfill [Pires et al. 2008]. Next, we briefly discuss each one:

i. *Exported schema representation*: peer's metadata should be mapped onto an ontological description, using a common and standard model;
ii. *Global conceptualization*: a global ontology may be used to provide a high-level view over the heterogeneous peer schemas;
iii. *Support for correspondences identification*: an ontology may also be used to assist the identification of correspondences between peer ontologies (i.e., peer schemas);
iv. *Support for query answering*: query answering in a PDMS may use a global ontology in a twofold way: a) as a high-level view of the sources; and b) as a terms' reference for query reformulation between peers. The former is concerned with query formulation, i.e., the user can formulate a query using the global ontology without specific knowledge of the different data sources stored in the peers. The latter is concerned with query reformulation, i.e., the query is reformulated into a target query over other connected peers, according to the defined correspondences among them;
v. *Semantic Index*: a semantic index can be built according to the main terms or categories referring to a set of ontologies. Such index must enable efficient location of peers;
vi. *Semantic matchmaking capabilities*: a semantic matching component is needed for matching ontologies in order to find out which concepts match in different ontologies and (possibly) at which level. Such capability can be used for the organization of peers in the network and the definition of correspondences between peers.

A system should take into account the previous requirements not only to be considered an OPDMS, but also to take advantage of using ontologies for semantic enrichment. In order to fulfill those requirements, Pires [Pires 2007] has proposed an OPDMS architecture which is used in this work as our running setting. Such architecture will be described in Chapter 6.

4.4. Concluding Remarks

In this chapter, we described the way we use ontologies in order to: (i) identify semantic correspondences between ontologies; (ii) represent and store context; and (iii) extend the definition of an OPDMS. Regarding the first usage, we presented an approach which identifies, besides the traditional types of correspondences (equivalence and subsumption), some other ones (e.g., closeness and disjointness). To this end, we make use of background knowledge by means of domain ontologies. As we have seen, such knowledge is useful in the reconciliation process, mainly when the context of interpretation of the involved matching concepts is precisely known (considering that the ontologies belong to the same domain).

Concerning issue (ii), representing context using an ontology brings various benefits. It provides concept subsumption, concept consistency and instance checking (including object properties checking). Efficient implementation of these operations allows a distributed setting (e.g., a PDMS) to organize knowledge, maintain its consistency, answer queries considering contextual elements and recognize conditions that trigger rule firings.

Finally, OPDMS are the result of blending the benefits of PDMS with the employment of the richer semantics obtained using ontologies. In this sense, we have proposed an extension to the OPDMS definition through the identification of high-level requirements that an OPDMS should fulfill. The idea is that ontologies are essentially employed for the definition and application of semantics in all services of a PDMS.

CHAPTER 5

"Everyone has a voice deep down inside them; a voice that says, "shine".
Some have loud voices; some have quiet ones.
Some people's voice is so quiet they never get to hear it.
Some of the people that get to hear it choose not to listen.
Unfortunately, only a few decide to act on this voice –
the stars, the lifters who influence others to listen themselves."
Aaron Betesta

The *SemRef* Approach

Sometimes, query answers which are not an exact match, but which are a close match to the requirements specified at query submission time, can still serve the purpose of users, depending on their preferences and on the dynamicity of the environment. Considering that, the main contribution of this thesis is to bring together the concept of *query reformulation* and the concept of *query enrichment* in dynamic distributed environments. To this end, we use semantics derived from the correspondences among the ontologies which represent peer schemas as well as semantics obtained from the context of the user, of the query and of the environment. The purpose underlying our approach, named *SemRef*, is to enhance query reformulation by using these kinds of semantics in such a way that we can provide users with a *set of expanded answers*. *Exact* and *enriched* query reformulations will be produced as a means to obtain this set of answers. In this chapter, we present important definitions regarding the use of semantics. Furthermore, we present the algorithms underlying our approach which may be instantiated in any ontology-based distributed environment.

This chapter covers the following contents: Section 5.1 introduces an overview of the *SemRef* approach; Section 5.2 defines query enrichment and set of expanded answers; Section 5.3 discusses how context is used in our work; Section 5.4 presents the *SemRef* algorithm; Section 5.5 provides an example showing *SemRef* in practice; Section 5.6 compares existing query reformulation approaches with ours. Finally, Section 5.7 concludes the chapter with some remarks.

5.1. Overview of the Approach

The environments we are considering in our work are characterized by having a diversity of perspectives, dynamic data, and the possibility of intermittent participation. As we have stated in Chapter 1, they are generally composed by a set of autonomous and heterogeneous *peers* which are associated by means of correspondences. To help matters, ontologies have been considered as a basis for making explicit the content of these data sources and, consequently, as a means for promoting information integration. In this perspective, the crucial point we address is how to reformulate queries among the peers in such a way that the resulting set of answers closely expresses what the users intended to obtain, taking into account what kind of data the sources may contribute with and the dynamicity of the system.

Some aspects should be addressed when dealing with query reformulation. First, retrieving relevant information wherever it may be or querying overlapping sources of information should be transparent and useful for users. By *transparent*, we mean that users are not supposed to be aware about where obtained data are stored, or even how they have been integrated. By *useful*, we mean that resulting query answers should be in conformance with users' preferences, since they usually want to obtain additional related information stored in other peers that are not possible to get using only their local data. On the other hand, it is not useful for users when they do not receive any answer at all. A second aspect is that concepts from a source peer do not always have exact corresponding concepts in a target one, what may result in an empty reformulation and, possibly, no answer to the user. In this sense, if it is not possible to produce an exact answer to a given query or if users define that it is relevant for them to receive semantically related answers, it may be better to produce an approximate or enriched answer than to produce no answer at all. Therefore, regarding the former aspect, we argue that user preferences and the current status of the environment should be taken into account at query reformulation time; regarding the latter, the original query should be adapted to bridge the gap between the two sets of concepts, using not only equivalence correspondences but also other ones that can approximate and/or enrich the queries.

In this light, we present a query reformulation approach, named *SemRef*, which uses semantics as a way to better deal with the afore mentioned aspects. Thus, in order to capture user preferences, query semantics and environmental parameters, we use contextual information. We accomplish query reformulation and adaptation by means of query enrichment. To this end, besides equivalence, we use other correspondences which go beyond the ones commonly found, namely: specialization, generalization, aggregation, disjointness and closeness, as defined in Section 4.1. By using this set of semantic correspondences, we are able to produce two different kinds of query reformulations [Souza et al. 2007; Souza et al. 2009]:

iii. an *exact* one, considering only equivalence correspondences; and
iv. an *enriched* one, resulting from the set of the other correspondences.

In order to present more clearly the problem of reformulating and, at the same time, enriching queries in dynamic distributed systems, we make some simplifying assumptions. First of all we will only consider two peers P_1 and P_2 that want to communicate. Then we assume that there are only two ontologies involved, a source one named O_1 (at submission peer P_1) and a target one named O_2 (at P_2), as well as a set of semantic correspondences between them $\{Co_{12}\}$. We further assume that both ontologies are encoded on the same language and belong to the same knowledge domain with considerable overlapping content between them. Figure 5.1 illustrates this setting.

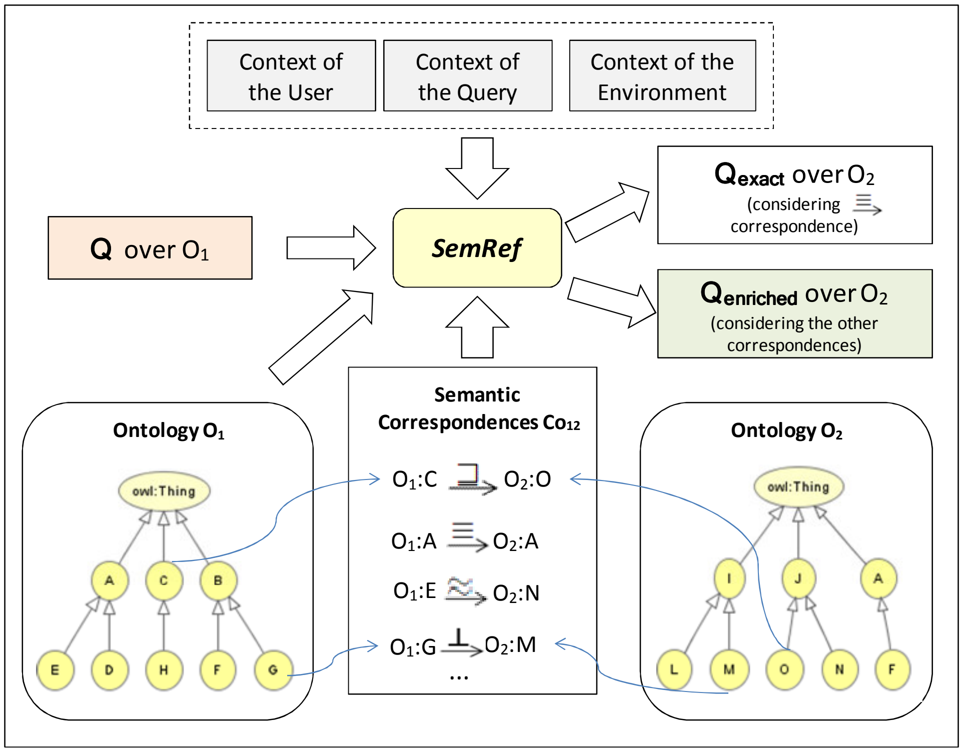


Figure 5.1 Query Reformulation Setting

Regarding this simplified setting, we define our problem as follows: given an ontology O_1 (at peer P_1), a user query Q expressed in terms of the concepts of O_1, a target ontology O_2, our goal is to find reformulated queries of Q expressed in terms of the concepts of O_2 in such a way that these reformulated queries not only include the best possible one, i.e., exact (considering equivalence correspondences) but also the ones provided by other semantic correspondences between the ontologies. The reasons underlying that are twofold:

i. We consider that answers which are not an exact match, but which are a *close match* to the requirements specified in the query, can still serve the purpose of users, if these answers are in conformance with their preferences;
ii. We want to provide users with a set of *expanded answers*, in the light of the differences between the existing sets of concepts in the peers, and taking into account the context surrounding the query. This set of expanded answers will be obtained by executing *exact* (i.e., best) and *enriched* reformulations.

Furthermore, *SemRef* uses the concept of context, as a way to enhance the overall query reformulation process and to deal with information that can only be acquired on the fly. In order to analyze the semantics around a submitted query, we propose to take into account the context of the user, of the query and of the environment. The users' context is acquired through a set of contextual variables allowing them to express their preferences regarding the reformulation process, i.e., the degree of approximation users want. The context of the query is derived from the submitted query and the way it will be reformulated. Finally, the context of the environment will be captured using some parameters defined by users and taking into account the availability of peers.

5.2. Enriching Queries and Producing Expanded Answers

Although a considerable effort has been employed in recent years to provide reformulation techniques which enrich user queries before their execution, this has been often limited to a single data source setting. We argue that query enrichment may be even more crucial in a dynamic distributed environment, where queries are usually interpreted according to targeted peers, and users are generally provided with limited answers in response to such queries.

In this work, we consider *query enrichment* as both a combination of query expansion and query personalization. Regarding the former, whenever we employ any kind of extra knowledge to adapt or expand the query, we perform query enrichment. Regarding the latter, whenever query enrichment is carried out taking into account the user profile or user preferences, we perform query personalization. In this sense, we define query enrichment as follows.

Definition 4 - *Query Enrichment*. Query enrichment is the process of analyzing an initial query expression in order to find out some extra semantic knowledge that can be added, so its resolution will provide expanded answers.

When identifying semantically related concepts to query terms, we say that these concepts are candidates for query enrichment, where a term can be added or even removed from the original query. The process of query enrichment, considering some kind of refinement by using additional knowledge, is not only concerned with adding terms, but perhaps with removing them. We say that query enrichment may occur by:

i. Substituting some terms with other ones that are semantically equivalent; or
ii. Expanding user queries by adding synonym or semantically related terms; or
iii. Reducing the scope of queries by removing some of its terms, in order to optimize the query and reduce the number of incorrect or redundant answers.

We present an example of each one of the previous possibilities. To this end, consider the SQL query Q = "Select name from Professor, Lecturer;". By using additional knowledge, for example an ontology or semantic correspondences, we verify that Professor is equivalent to Teacher, and a close concept of Educator; and Lecturer is equivalent to Instructor and a sub-concept of Educator. Therefore, analyzing the described options i, ii and iii, we would perform the following reformulations:

i. Substituting original terms by equivalent ones, we have: Q_1 = "Select name from Teacher, Instructor;"
ii. Expanding original terms by other semantic ones, we have: Q_1' = "Select name from Teacher, Educator, Instructor, Educator;"
iii. There is a redundancy in query Q_1', i.e., the concept Educator has been used as a semantically related one of both original concepts Professor and Lecturer, what resulted in its repetition. We can optimize the final reformulated query by removing one of these repetitions. Thus, the final reformulated query would be: Q' = "Select name from Teacher, Educator, Instructor;".

In our approach, we work with query enrichment by means of items i and ii, in such a way that the original query is expanded with additional semantically related terms at query reformulation time.

Considering a dynamic distributed environment, there are, at least, three context-dependent factors which influence the answer to a given user query: (i) the current status of the network, mainly in terms of the available peers; (ii) the semantics underlying the submitted query and (iii) the users' preferences regarding the query answering process. As a result, we can see that the context has an impact on the quality and content of the produced answer, and thereby, we cannot guarantee that the complete set of answers will always be returned to users. On the other hand, considering these contextual elements, we can enrich queries, and thus produce a set of answers that match users' preferences while hold environmental conditions. Moreover, by applying query enrichment at query reformulation time, we can guarantee that *a set of expanded answers* will be returned to users, defined as follows.

Definition 5 – *Set of Expanded Answers*. *Expanded answers* are query answers provided by the available peers, by using the set of existing semantic correspondences, in such a way that these answers are semantically related to the user's preferences when formulating the original query.

This notion is context-dependent, relying on many aspects: what the user wants, the status of the network, its connectivity, the subject and goal of the query, and other circumstances. It means that in dynamic distributed settings, answers to user requests in most cases are not supposed to be complete, as they usually are in other integration approaches. However, the answers shall be as close as possible to users' needs, and they shall reflect the current status of the environment. The set of expanded answers should conform to users' preferences, as we will explain in the next Section. The goal of our work is to produce exact and enriched query reformulations as a means to provide users with such set of expanded answers.

5.3. Using Context to Enhance Query Reformulation

In our setting, for each submitted query, the whole query execution process instantiation may change completely, thus the context around the query (its semantics), the peers (availability), semantic correspondences (their different types) and the user (preferences) are essential information that have to be dealt with. In our approach, we make use of three types of context:

i. the context of the users, represented by the set of preferences that they define;
ii. the context of the query, acquired from the identification of its semantics (including concepts, properties and operators) and its query reformulation mode; and
iii. the context of the environment, where we identify the relevant peers to send the reformulated queries.

The context of the users is acquired when they initialize a query session. At this moment, they may state their preferences concerning the reformulation policy. Since the exact reformulation of a given query Q will always be produced (it is the default option), these preferences involve setting four variables which specify what should be considered when Q is also to be enriched. Enriching variables are defined as follows:

- *Approximate*: includes concepts that are close to the ones of Q;

- *Specialize*: includes concepts that are sub-concepts of some concepts of Q;
- *Generalize*: includes concepts that are super-concepts of some concepts of Q; and
- *Compose*: includes concepts that are part-of or whole-of some concepts of Q.

If all variables are set to false, this indicates that the user wants only the exact reformulation (considering equivalence correspondences). However, if at least one of them is set to true, it means that the algorithm (which will be presented in next Section) should consider such preference when producing an enriched reformulation. For example, if *approximate* is set to TRUE, the algorithm will verify *closeness* correspondences in order to reformulate the original query. These variables help to guide the execution of the query reformulation algorithm. They may be defined for some period of time (e.g., for 30 queries or for the whole query session), i.e., they are not related to the execution of the algorithm for a single query, since it seems not realistic to force the user to specify the values of these variables for each query. Nevertheless, whenever users want, they can redefine the variables.

The context of the query is obtained in a twofold way: (i) through the analysis of its semantics, i.e., its essential features and goal, and (ii) through the query reformulation mode, which is defined by the user, at query submission time. In the former, the query's required concepts, properties and operators are identified and dealt with in the query reformulation algorithm. The latter is concerned with the way the reformulation algorithm will operate. There are two reformulation modes:

- *Restricted*: it is the default option. In this case, the priority is to produce an exact reformulation, although if this reformulation results empty, then an enriched reformulation may be provided in place of the empty exact one; and
- *Expanded*: in this option, both exact and enriched reformulations are to be produced.

The context of the environment is acquired at two different moments: (i) at Query Session Configuration time, when users define another variable, named *Path_Length*, where they delimitate the number of subsequent reformulations (forwardings) in the set of relevant peers and; (ii) at Query Submission time, when the system identifies in which peer the query has been submitted (submission peer's identification), and the system also establishes the context of the submission peer neighbors (peer's availability, whether or not it can apply the required operators). Based on this set of contextual elements, the system defines where to route and reformulate the queries.

Most contextual information that is used in this work is acquired at query submission time. Some are gathered from the users' preferences, i.e., the way they expect the reformulation algorithm operates. Others are inferred on the fly according to the environment's capabilities. In this current version of the work, we have already represented both kinds of context using CODI (for more details, we refer the reader to Section 4.2). However, at the moment, we only deal with the context acquired from the user preferences (*enriching variables*, *path_length* and *query reformulation mode*). In this light, taking into account the users' perspective at query submission time, we provide an overview of the steps users are supposed to go over when they set their preferences regarding the query reformulation process. This overview is presented by the UML[14] activity diagram depicted in Figure 5.2.

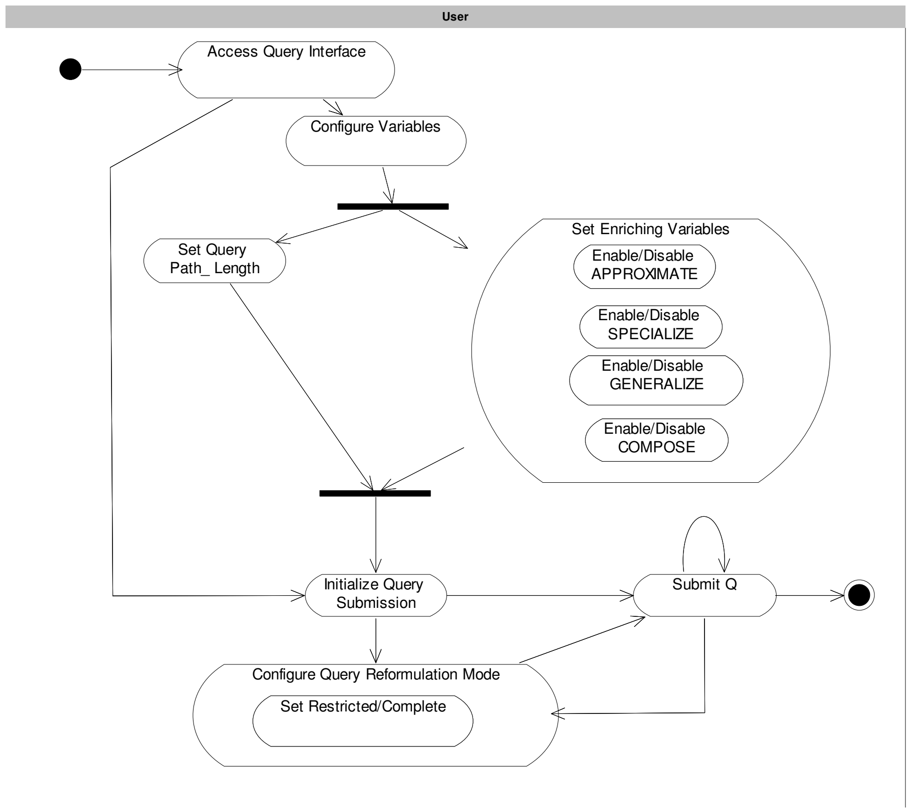


Figure 5.2 Activity Diagram for User's Query Submission

As an illustration of the query submission process, suppose a user "Anna" wants to query the system. She accesses the query submission module interface. Initially, she does not want any kind of enrichment, so she lets *enriching variables* disabled, i.e., she skips variables configuration. She also starts to submit queries. In this case, queries will be reformulated considering default configuration, i.e., restricted reformulation mode and no enrichment option. After submitting 20 queries, she is a bit disappointed and verifies that some queries have no answers. At this moment, she enables two enriching

[14] Unified Modeling Language

variables, namely, approximation and specialization. She submits 10 queries, still in restricted mode. However, enabling the two refereed variables, she has allowed the algorithm to produce an *enriched* reformulation in place of *empty exact* reformulations. As a result, she receives answers from the majority of submitted queries. At end, she decides to choose the expanded reformulation mode. By doing so, she is able to receive both exact and close answers, i.e., expanded answers, resulting from exact and enriched reformulations.

Next section, we will present the *SemRef* algorithm. The considered contextual information will help to guide the generation of diverse types of reformulations.

5.4. The *SemRef* Algorithm

Query reformulation is considered the most important aspect of query answering in a distributed environment, since it is crucial for the system's ability to answer user queries. The goal of our approach is to enhance the query reformulation capabilities using semantics derived from the correspondences among peer ontologies, according to the contextual information which has been obtained [Souza et al. 2009]. In this section, we present the Semantic Query Reformulation algorithm, namely *SemRef*. To this end, considering the query reformulation scenario presented in Figure 5.1, we make the following assumptions:

i. There are two peers – a submission peer P_1 and a target peer P_2;
ii. Each peer is described by an ontology O_1 and O_2, respectively;
iii. The peers (and their ontologies) are within the same knowledge domain;
iv. An ontology describing the domain is available (DO) and
v. The set of semantic correspondences $Co[O_1,O_2]$ between the considered two peer ontologies has been generated.

Our query reformulation approach has been encoded in $\mathcal{ALC}$-DL [Baader et al. 2007]. The basic elements in $\mathcal{ALC}$-DL are concepts, roles and individuals. Concepts (class of individuals) and roles (relations between individuals) are either primitive (named concepts or roles) or complex (recursively defined with constructors and other concepts or roles). The supported constructors are: $\neg C$ (negation), $C \sqcap D$ (conjunction), $C \sqcup D$ (disjunction), $\forall R.C$ (universal restriction) and $\exists R.C$ (limited existential restriction) where C and D are concepts and R is a role. We use these definitions in order to describe the peers' ontologies, their semantic correspondences and the submitted and reformulated queries.

Peer ontologies (or ontologies, in general) are composed of axioms asserting truth about a knowledge domain. In our approach, we formalize an ontology as a triple

$O = \langle C, R, I \rangle$, where C is a set of $\mathcal{ALC}$-concepts, R is a set of $\mathcal{ALC}$-role definitions and I is a set of individuals or instances.

A query Q over a peer ontology O_i is a concept expression, $Q = C$, where C is an $\mathcal{ALC}$ concept. However, since an $\mathcal{ALC}$ concept may be an atomic concept or a complex concept including roles, quantifiers, conjunctions or disjunctions, we generalize a query formula Q as a disjunction of queries which are themselves conjunctions of $\mathcal{ALC}$ concepts $C_1, ..., C_N$ where $n \geq 1$, as follows.

Definition 6 - *Query*. Q is a query expressed over P_i's ontology, having the following form: $Q = Q_1 \sqcup Q_2 \sqcup ... \sqcup Q_M$, where $Q_i = C_1 \sqcap C_2 \sqcap ... \sqcap C_N$, and where each C_j is an atomic concept, a negated atomic concept of a quantified atomic concept (in other terms: C_j, $\neg C_j$, $\forall R.C_j$ or $\exists R.C_j$).

We state that all submitted and/or reformulated queries follow this general query formula. Indeed, the class of formulas we are considering here is the so-called disjunctive normal forms (or DNFs) [Tonin and Bittencourt 2000], i.e., disjunctions of conjunctions of $\mathcal{ALC}$ concepts. Therefore, well-formed query formulas may include a disjunction of n conjunctions, a conjunctive query expression (which can be transformed to a disjunction with one element) or, simply, an $\mathcal{ALC}$ concept such as C, $\neg C$, $\forall R.C$ or $\exists R.C$, $\top$, or $\bot$.

Supposing a peer ontology concerning an academic research center, with concepts such as Teacher, GraduateStudent, Student and Researcher, we provide some examples of queries following Definition 6:

$Q_1 = \neg Teacher$, which asks for all non-teacher people belonging to a research center.

$Q_2 = GraduateStudent$, which asks for the existing graduate students.

$Q_3 = [Teacher \sqcap Researcher] \sqcup [Student \sqcap Researcher]$, which asks for people who are teachers and researchers or students that are also researchers.

Considering the user's perspective when posing queries such the ones provided before, we argue that users should be aware that they may need not only exact answers, but also those answers that meet or complement their initial intention. Moreover, they may prefer an alternative answer to their query than not receiving any answer at all. Thus, if it is not possible to produce an exact answer to a given query or if users define that it is relevant for them to receive semantically related answers, it may be better to produce an approximate or enriched answer than to produce no answer at all. In this sense, we consider that a query formulated in terms of a source peer ontology may be reformulated exactly or approximately into a query using terms of a target peer ontology, according to the set of semantic correspondences between them. Regarding the defined query formula, the query reformulations are produced according to the following definitions:

Definition 7 - *Exact Reformulation.* A reformulation Q’ of a query Q is said to be exact (denoted as Q_{exact}) if each concept (or property) C’ of Q’ is related to a concept (or property) C of Q by a Co correspondence, where Co ∈ $\{\xrightarrow{\equiv}\}$ (the set of *equivalence* correspondences).

Definition 8 - *Enriched Reformulation.* A reformulation Q’ of a query Q is said to be enriched ($Q_{enriched}$) if each concept (or property) C’ of Q’ is related to a concept (or property) C of Q by a Co correspondence, where Co ∈ $\{\xrightarrow{\sqsubseteq}, \xrightarrow{\sqsupseteq}, \xrightarrow{\approx}, \xrightarrow{\triangleright}, \xrightarrow{\triangleleft}, \xrightarrow{\perp}\}$ (the set of *specialization*, *generalization*, *closeness*, *part-of*, *whole-of* and *disjointness* correspondences).

Exact reformulations are always produced by *SemRef* Algorithm, although sometimes they may result empty (this happens when there is no equivalence correspondence between concepts in the submitted query and concepts in the target ontology). On the other hand, enriched reformulations will be produced in two situations:

i. If the user requires enrichment by defining enriching variables (*approximate*, *specialize*, *generalize* and/or *compose*) and *expanded* query reformulation mode; or
ii. If the user has set the *enriching* variables (at least one of them), the reformulation mode was defined as *restricted*, but the produced exact reformulation resulted empty.

The possible reformulation modes and resulting query reformulations are described in Table 5.1. This table also presents the various possibilities concerning the combination of choices that a user may define and the kind of reformulated queries that *SemRef* will be able to produce.

For instance, consider that now user “Anna” sets the specialize and generalize variables to TRUE, the query reformulation mode to expanded, and then submits query Q = C. The *SemRef* algorithm will produce both *exact* and *enriched* reformulations of Q. As a result, Anna will receive from the target peer a set of expanded answers that comprise instances of equivalent, sub-concepts and/or super-concepts of C. Later, Anna comes back and now changes the reformulation mode to restricted, although she does not change the enriching variables. She submits query Q_1 = C_1. *SemRef* tries to produce an exact reformulation of Q_1, but, due to the fact that there is no corresponding equivalent concept of C_1 in the target peer, Q_{exact} results empty. Since enriching variables specialize and generalize are still set, the algorithm will produce an enriched reformulation $Q_{enriched}$ of Q_1 in place of the empty Q_{exact}.

Table 5.1 User Preferences and Produced Reformulations

Enriching Variables *Approximate* *Compose* *Specialize* *Generalize*	**Mode**		**Produced Reformulated Queries**
	Expanded	*Restricted*	
At least one is TRUE	TRUE	FALSE	Exact Enriched
All are FALSE	TRUE	FALSE	Exact
At least one is TRUE	FALSE	TRUE	Exact Enriched, if Exact is EMPTY
All are FALSE	FALSE	TRUE	Exact

In this light, the *SemRef* algorithm receives as input a given query Q, submitted in a peer P_1, the target peer P_2, $Co[O_1, O_2]$ (the set of semantic correspondences between O_1 and O_2), and the context that has been set by the user (enriching variables and query reformulation mode values). As output, it produces one or two reformulated queries (Q_{exact} and/or $Q_{enriched}$), according to the possibilities shown in Table 5.1. A high level view of *SemRef* is sketched in Figure 5.3. The complete *SemRef* algorithm is detailed in Figure 5.4.

```
SemRef(Q, P1, P2, Co[O1,O2], MODE, REF_VAR, Qexact, Qenriched)
Input: Q, P1, P2, Co[O1,O2], MODE, REF_VAR
Output: Qexact, Qenriched

1.  For each conjunctive query $Q_k$ in Q
2.       Find exact reformulation $Q_{k\_exact}$ of $Q_k$
3.       If (one of APPROXIMATE, COMPOSE, SPECIALIZE, GENERALIZE is TRUE)
4.            Then
5.                 Find enriched reformulation $Q_{k\_enriched}$ of $Q_k$
6.  End For;
7.  If (at least one of $Q_{k\_exact} \neq \varnothing$)
8.       Then
9.            Build final exact reformulation $Q_{exact}$ of Q
10.      Else $Q_{exact} \leftarrow \varnothing$
11. If ((MODE is expanded) or (MODE is restricted and $Q_{exact}$ is empty)) and
12.      (at least one of $Q_{k\_enriched} \neq \varnothing$)
13.      Then
14.           Build final enriched reformulation $Q_{enriched}$ of Q
15.      Else $Q_{enriched} \leftarrow \varnothing$
16. End SemRef;
```

Figure 5.3 High Level View of the *SemRef* Algorithm

```
SemRef (Q, P_1, P_2, Co[O_1,O_2], MODE, REF_VAR, Q_exact, Q_enriched)

For each Q_k in Q        /* for each conjunctive query in Q */

    B ← TRUE         /* B will be used to stop the search if some concept of Q_k has no
                     correspondent concept in  P_2 */

    While (there is still a concept C_j in Q_k to process) and (B=TRUE)
        S_1C_j ← ∅          /* set of concepts that are equivalent to C_j */
        S_2C_j ← ∅          /* set of concepts related to C_j by other kind of correspondence,
                             except disjointness */
        Neg_S_2C_j← ∅     /* set of concepts related to C_j by disjointness correspondence */

        For each equivalence assertion between C_j and a concept C' /* C' is in O_2 */
            Add C' to S_1C_j
        End For;

        For each other kind of assertion involving C_j   /* different from equivalence */
          If SPECIALIZE = TRUE
            Then
                If there is a concept C' in P' such that C' ⊑ C_j   /* subConceptOf */
                Then
                    Add C' to S_2C_j
                End If;
            End If;

          If APPROXIMATE=TRUE
            Then
                If there is a concept C' in P' such that C' ≈ C_j   /* closeTo */
                Then
                    Add C' to S_2C_j
                End If;
            End If;

           If GENERALIZE= TRUE
            Then
                If there is a concept C' in P' such that C' ⊒ C_j  /* superConceptOf */
                Then
                    Add C' to S_2C_j
                End If;
            End If;

           If COMPOSE= TRUE
            Then
                If there is a concept C' in P' such that C' ▹ C_j or C' ◃ C_j   /* related through
                a part of  or a whole of correspondence*/
                Then
                    Add C' to S_2C_j
                End If;
            End If;

            If Cj is negated
            Then
                If there is a concept C' in P' such that C'⊥ C_j  /*  they are disjoint */
                Then
                Add C' to Neg_S_2C_j
                    BNeg ← TRUE
                End If;
            End If;

          End If;
        End For; /* End of the loop related to the assertions different from ≡ */
```

```
            If ($S_1C_j = \varnothing$ and $S_2C_j = \varnothing$ and $Neg\_S_2C_j = \varnothing$)
                  Then
                        B ← FALSE   /* there is no correspondence between $C_j$ and concepts of $P_2$ */
                  End If;
            End While;   /* End of the loop processing concepts */

            $B_1$ ← TRUE;
            If any $S_1C_j = \varnothing$
              Then $B_1$ ← FALSE
            End If;     /* Checking if there was an empty set concerning exact correspondences */

            $B_2$ ← TRUE;
            If any $S_2C_j = \varnothing$
              Then $B_2$ ← FALSE
            End If;   /* Checking if there was an empty set concerning enriching correspondences*/

            If $B_1$ = TRUE
                  /* if there were exact correspondences and no resulting empty set, then we can build the exact
                  reformulation for the current Q */
              Then
                 $Q_{k\_exact}$ ← Build_Exact_Reformulation ($Q_k$, $S_1C_1$, $S_1C_2$, …, $S_1C_p$)
              Else
                 $Q_{k\_exact}$ ← $\varnothing$
              End If;

              If $B_2$ = TRUE or BNeg = TRUE
                  /* if there were enriching correspondences and no resulting empty set, then we can build the
                  enriched reformulation for the current Q */
             Then
                  $Q_{k\_enriched}$ ← Build_Enriched_Reformulation ($Q_k$, $S_2C_1$, … $S_2C_p$, $Neg\_S_2C_1$, … $Neg\_S_2C_p$)
             Else
                  $Q_{k\_enriched}$ ← $\varnothing$
             End If;

End For;   /* End of the loop processing the conjunctive queries $Q_k$ */

If (at least one of $Q_{k\_exact} \neq \varnothing$)    /* at least one of $Q_k$'s exact reformulations is not empty  */
   Then
        $Q_{exact}$ ← Build_Final_Exact_Reformulation (Q, $Q_{1\_exact}$, …, $Q_{m\_exact}$)
   Else
        $Q_{exact}$ ← $\varnothing$
End If;

If ((MODE is expanded) or (MODE is restricted and $Q_{exact}$ is empty)) and
     (at least one of $Q_{k\_enriched} \neq \varnothing$)

       /* If MODE is expanded or MODE is restricted and $Q_{exact}$ is empty; and
       at least one of $Q_k$'s enriched reformulations is not empty  */

   Then
         $Q_{enriched}$ ← Build_Final_Enriched_Reformulation (Q, $Q_{1\_enriched}$,…, $Q_{m\_enriched}$)
   Else
         $Q_{enriched}$ ← $\varnothing$
   End If;

End_SemRef;
```

Figure 5.4 The *SemRef* Algorithm

More precisely, in order to obtain the reformulations, the algorithm performs the following tasks:

I. It receives query Q as a disjunction of conjunctions of $\mathcal{ALC}$ concepts, i.e., $Q = Q_1 \sqcup Q_2 \sqcup ... \sqcup Q_M$. For each conjunctive query Q_k in Q, while there are concepts C_j in Q_k to process, it adds corresponding concepts (according to existing semantic correspondences) to three kinds of sets:
   - S_1C_j: the set of concepts that are equivalent to C_j;
   - S_2C_j: the set of concepts related to C_j by other kind of correspondence (closeness, specialization, generalization, part-of and whole-of). This set is produced if the reformulation mode is expanded or it is restricted and S_1C_j is empty; and
   - $Neg\_S_2C_j$: if there is a negation over C_j, *SemRef* searches for disjointness correspondences in order to directly get the opposite concept. In this case, the concept is added to $Neg\_S_2C_j$ set. If there is no disjoint correspondence, a variable BNeg is set to TRUE and later in the algorithm, the negation is done over the corresponding concept found through the set of other semantic correspondences (equivalence, specialization, generalization, part-of, whole-of or closeness).

II. After processing all the concepts of a conjunctive query, *SemRef* verifies if there were exact correspondences and if the conjunction did not fail (i.e., all the existing concepts in the conjunction had corresponding ones). If so, it builds the exact reformulation for the current conjunctive query Q_k.

III. In the same way, if there were enriching correspondences and the conjunction did not fail, then *SemRef* builds the enriched reformulation for the current conjunctive query Q_k.

IV. Finally, after processing all the conjunctive queries Q_k of Q, *SemRef* produces the final Q_{exact}, as the disjunction of the resulting exact conjunctions and the final $Q_{enriched}$ as the disjunction of the resulting enriched conjunctions.

We have defined some complementary functions which are called by *SemRef* algorithm. The *Build_Exact_Reformulation* function is responsible for building a given reformulation for one conjunction, considering the resulting set of corresponding concepts obtained from equivalence. Similarly, we have the *Build_Enriched_Reformulation* function which shows how we build a given reformulation for one conjunction, taking into account the resulting sets of corresponding concepts obtained from the specialization, generalization, closeness, part-of, whole-of and disjointness. In quite the same way, the *Build_Final_Exact_Reformulation* and *Build_Final_Enriched_Reformulation* consider the already produced Q_{m_exact} and $Q_{m\_enriched}$ to generate the final disjunction of them, respectively Q_{exact} and $Q_{enriched}$. These functions are presented in Appendix A.

In the following, we show the main properties of the *SemRef* Algorithm: (i) *termination*, meaning that it always terminates; (ii) *soundness* or correctness, meaning that every produced query reformulation is a "correct" reformulation solution; and (iii) completeness, implying that it always gives a solution when there is one. In order to present the proofs, we need to state some hypotheses:

- The original submitted query Q is posed at a peer P_1, by means of an ontology O_1. It follows the general formula (Definition 6), i.e., $Q = Q_1 \sqcup Q_2 \sqcup \ldots \sqcup Q_n$, where Q_i is a conjunctive query $C_1 \sqcap C_2 \sqcap \ldots \sqcap C_m$, and C_j is a concept of the initial ontology O_1; and
- $R(Q, O_1, O_2)$ is the set of reformulated queries returned by the *SemRef* algorithm for a given query Q posed over the ontology O_1; each query Q' in $R(Q, O_1, O_2)$ is expressed over the target ontology O_2.

The following theorem assures that *SemRef* is able to produce both *exact* and *enriched* reformulations of a submitted query Q and that these produced reformulations are correct. In addition, we prove that the algorithm terminates.

Theorem 1 (Soundness of *SemRef*). Let Q be a query in ontology O_1, then

- If Q is reformulated into Q' over ontology O_2 by equivalence correspondence, then Q' is an exact reformulation of Q.
- If Q is reformulated into Q' over ontology O_2 by specialization, generalization, aggregation, closeness and/or disjointness correspondences, then Q' is an enriched reformulation of Q.
- Each reformulated query Q' in $R(Q, O_1, O_2)$, returned by the *SemRef* algorithm, is a correct reformulation.

Proof. We need to prove that every reformulation Q' in $R(Q, O_1, O_2)$ is a correct reformulation of Q and is either an exact or an enriched reformulation of Q. To this end, suppose that Q' is a query, such that Q' is in $R(Q, O_1, O_2)$, but Q' is not a correct reformulation of Q (i.e., it is neither an exact nor an enriched reformulation). If Q' in $R(Q, O_1, O_2)$ is not a correct reformulation of Q, then, there is at least a concept C' in Q' such that for each concept C in Q the following assertions hold:

- $\neg(C' \xrightarrow{\equiv} C)$
- $\neg(C' \xrightarrow{\sqsubseteq} C)$
- $\neg(C' \xrightarrow{\sqsupseteq} C)$
- $\neg(C' \xrightarrow{\approx} C)$
- $\neg(C' \xrightarrow{\rhd} C)$

- $\neg(C' \xrightarrow{\triangleleft} C)$
- $\neg(C' \xrightarrow{\perp} C)$

If C’ is not related to C using any of the seven kinds of semantic correspondences, then $C' \notin S_1C$, $C' \notin S_2C$ and $C' \notin Neg\_S_2C$. The two functions *Build_Exact_Reformulation* and *Build_Enriched_Reformulation* are called with arguments S_1C, and S_2C and Neg_S_2C, without C’ in any of the three sets. As a consequence, C’ is not in the resulting set of reformulated queries, what is *contradictory* to our hypothesis.

Therefore, if, for each concept C’ in any query Q’ of $R(Q, O_1, O_2)$, there is a concept C in O_1 such that one of the seven semantic correspondences hold, then each Q’ of $R(Q, O_1, O_2)$ is a correct reformulation of Q, i.e., Q’ is either an exact or enriched reformulation of Q, and the *SemRef* algorithm is sound.

Termination of the *SemRef* algorithm is immediately implied by the fact that the number of reformulated queries that can be generated by the algorithm is finite, i.e., it produces zero, one or two reformulations (Q_{exact} and/or $Q_{enriched}$) of a submitted query Q.

□

The above theorem highlights the crucial role played by the semantic correspondences in the generation of query reformulations. Moreover, it shows that no wrong reformulation is returned by *SemRef*.

The following theorem presents the condition for the algorithm to be complete.

Theorem 2 (Completeness of *SemRef*). Given an original submitted query Q, the *SemRef* algorithm is able to find all the existing solutions (reformulations) for Q in $R(Q, O_1, O_2)$.

Proof. Suppose now that Q’ is a query over O_2 such that Q’ is not in $R(Q, O_1, O_2)$, but Q’ is a correct reformulation of Q. If a query Q’ is not in the resulting set of queries $R(Q, O_1, O_2)$ then Q’ is not an exact reformulation of Q and Q’ is not an enriched reformulation of Q. However, if Q’ is neither an exact nor an enriched reformulation of Q, then there is at least a concept C’ in Q’ such that $C' \notin S_1C$, and $C' \notin S_2C$ and $C' \notin Neg\_S_2C$, i.e., C’ is not related to any concept C of the initial query using any of the seven semantic correspondences. Thus, the query Q’ is not a correct reformulation, what is *contradictory* to our hypothesis.

As a conclusion, if a query Q’ is not in the resulting set of reformulated queries, then there is a concept C’ in this query which is not semantically related to any concept of the initial query; therefore, Q' is not a correct reformulation of Q. If all the correct reformulations of Q are in R(Q, O_1, O_2), the *SemRef* algorithm is complete.

□

This theorem means that no existing solution is missing in the reformulation result set.

As a brief illustration of *SemRef*'s execution, consider two peers with their corresponding ontologies and the following semantic correspondences between them: O_1:Animal $\xrightarrow{\perp}$ O_2:Plant; O_1:Animal $\xrightarrow{\perp}$ O_2:Mineral; and O_1:Animal $\xrightarrow{\approx}$ O_2:Beast. Now, assume that the query Q = ¬Animal was submitted in P_1. The *SemRef* algorithm starts by initializing three important sets: S_1C_1, S_2C_1 and Neg_S_2C_j. The first set will receive all the concepts that match Animal according to *isEquivalentTo* correspondences. Since there is no such correspondences, this set results empty (S_1C_1 = { }). The second set will receive the concepts resulting from the *subConceptOf*, *superConceptof*, *isCloseTo*, *isPartOf* and *isWholeOf* correspondences, if the corresponding enriching variables (*generalize, specialize*, *compose* and *approximate*) have been set to TRUE by the user. Assuming that all four variables have been set to TRUE and the reformulation mode has been defined to EXPANDED, then S_2C_1 is set to {Beast} (according to the closeness correspondence between O_1:Animal and O_2:Beast). The third set will receive animal's disjoint concepts, since there is a negation over the concept Animal. Thus, Neg_S_2C_1 = {Mineral, Plant}. Then, since the query is composed by only one concept, *SemRef* verifies the described sets and tries to build the exact and enriched reformulations. As S_1C_1 is empty, there is no exact reformulation for this given query. However, S_2C_1 and Neg_S_1C_1 are not empty, so the algorithm builds the enriched reformulation providing the negation over the concept Beast (from the closeness correspondence) and the union of this negated concept with the other found ones, taking into account the disjoint correspondences. As a result, $Q_{enriched}$ is set to [¬Beast ⊔ Mineral ⊔ Plant].

5.5. A More Complete Example

Our example scenario is composed by two peers P_1 and P_2 which belong to the "Education" knowledge domain. In this scenario, peers have complementary data about academic people and their works (e.g., research) from different institutions. Thus, it is very likely that a query may obtain a more complete result according to such diverse data sources. Each peer has got one ontology – O_1 (*Semiport.owl*) and O_2 (*UnivBench.owl*), respectively describing their schemas. In addition, we have considered as background knowledge a public domain ontology (DO) named *UnivCSCMO.owl*. For the sake of space, we present excerpts from them using OWLViz (a Protégé plug-in) in Figure 5.5 and Figure 5.6. The complete taxonomies are depicted in Appendix B[15].

In order to identify the semantic correspondences between O_1 and O_2, first, our matching tool found out the equivalences between concepts of O_1 and concepts in the domain ontology *UnivCSCMO*, and the equivalences between concepts of O_2 and *UnivCSCMO* as well. Then, the set of rules described in Section 4.1 was applied. As a result, the set of semantic correspondences between O_1 and O_2 was identified. Since the correspondences are unidirectional, first we present a fragment of the correspondences' set concerning the concept FullProfessor (from O_1) with its respective related concepts in O_2, in Table 5.2.

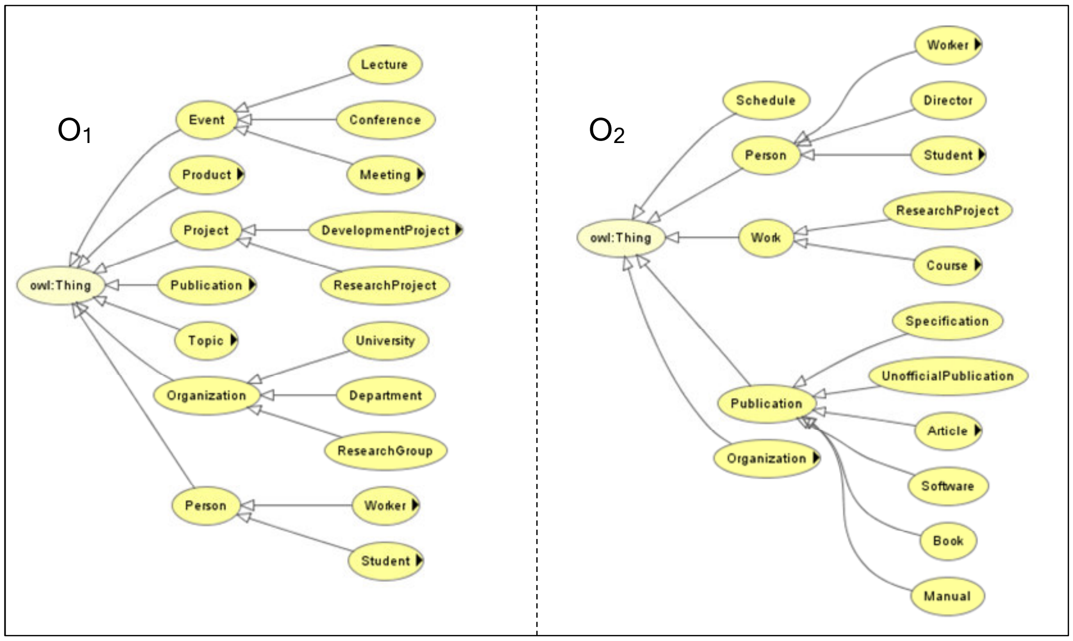


Figure 5.5 Excerpts from the Ontologies of P_1 and P_2

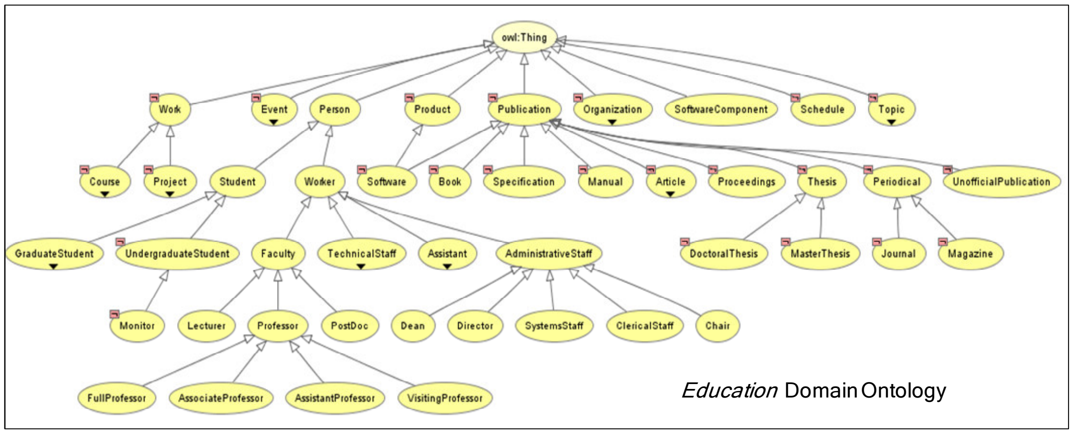


Figure 5.6 Excerpt from the *Education* Domain Ontology

[15] The complete ontologies are available at http://www.cin.ufpe.br/~speed/ontologies/Ontologies.html

Table 5.2. Some Semantic Correspondences between O_1 and O_2

Co_{12} for O_1:FullProfessor
O_1:FullProfessor $\xrightarrow{\equiv}$ O_2:FullProfessor
O_1:FullProfessor $\xrightarrow{\sqsubseteq}$ O_2:Professor
O_1:FullProfessor $\xrightarrow{\approx}$ O_2:VisitingProfessor
O_1:FullProfessor $\xrightarrow{\perp}$ O_2:AssociateProfessor
O_1:FullProfessor $\xrightarrow{\rhd}$ O_2:Course
O_1:FullProfessor $\xrightarrow{\rhd}$ O_2:ResearchProject

In this illustrative set, we can see the equivalence correspondence between FullProfessor in O_1 and O_2. This is the most commonly identified correspondence' type in traditional query reformulation approaches. We can also see that, taking into account the semantics underlying the DO, through the existing relationships, we can identify other unusual correspondences. In this fragment, FullProfessor has been identified as: (i) close to VisitingProfessor; (ii) subconcept of Professor; (iii) disjoint with AssociateProfessor; and; (iv) part of Course and of Research Project. We are able to determine such correspondences using the domain ontology knowledge.

Now considering the opposite direction (between O_2 and O_1), another illustration concerns the concept O_2:Course which has no direct corresponding concept in O_1 (i.e., no equivalent concept). Nevertheless, using the set of correspondence rules, we can determine five unusual semantic ones: O_2:Course $\xrightarrow{\lhd}$ O_1:UndergraduateStudent, O_2:Course $\xrightarrow{\lhd}$ O_1:GraduateStudent, O_2:Course $\xrightarrow{\lhd}$ O_1:AssistantProfessor, O_2:Course $\xrightarrow{\lhd}$ O_1:FullProfessor and O_2:Course $\xrightarrow{\perp}$ O_1:Project. Thus, at query reformulation time, these correspondences may provide a kind of semantic information which will somehow approximate O_2:Course of their semantically related concepts (or not, in case of disjointness), making possible a query reformulation enrichment, if the user enables this option. In summary, using the domain ontology, we can obtain kinds of correspondences which would not be possible, if we have considered only syntactic or linguistic criteria. Moreover, using these unusual semantic correspondences, *SemRef* can produce a larger set of query reformulations.

At this moment, using this scenario, we provide three query reformulation examples, presenting the *SemRef* main steps in practice. First, assume that the query Q = FullProfessor was submitted in P_1 and consider the set of correspondences shown in Table 5.2. The *SemRef* algorithm starts by initializing the sets S_1C_1, S_2C_1 and Neg_S_2C_1. The first set receives the concepts that match FullProfessor according to the equivalence correspondence, i.e., S_1C_1 = {FullProfessor}. The second set receives the concepts resulting from the other correspondences (except disjointness), according to the enriching variables definition. Assuming that the four variables have been set to TRUE, and the reformulation mode has been defined to EXPANDED, then S_2C_1 is set to {VisitingProfessor, Professor, Course, ResearchProject}. The third set would receive disjoint concepts, if there was a negation over the concept FullProfessor. Since the query is composed by only one concept and there is no negation over such concept, the algorithm verifies that both sets (S_1C_1 and S_2C_1) are not empty and consequently builds the exact and enriched reformulations. The final exact and enriched reformulations are the following:

- Q_{exact} = [FullProfessor]
- $Q_{enriched}$ = [VisitingProfessor ⊔ Professor ⊔ Course ⊔ ResearchProject].

The second query reformulation example regards the query Q = ¬TechnicalStaff ⊔ Lecturer submitted in P_1, as well. In order to explain such reformulation, consider the set of existing semantic correspondences concerning the concepts O_1:TechnicalStaff and O_1:Lecturer, presented in Table 5.3. In addition, we assume that the user has set to TRUE the four enriching variables (generalize, specialize, compose and approximate), although a RESTRICTED reformulation mode has been chosen.

Table 5.3. Some other Semantic Correspondences between O_1 and O_2

O_1 Concept	Semantic Correspondences
TechnicalStaff	O_1:TechnicalStaff $\xrightarrow{\sqsubseteq}$ O_2:Worker O_1:TechnicalStaff $\xrightarrow{\approx}$ O_2:Assistant O_1:TechnicalStaff $\xrightarrow{\approx}$ O_2:Faculty O_1:TechnicalStaff $\xrightarrow{\perp}$ O_2:AdministrativeStaff
Lecturer	O_1:Lecturer $\xrightarrow{\sqsubseteq}$ O_2:Faculty O_1:Lecturer $\xrightarrow{\approx}$ O_2:PostDoc O_1:Lecturer $\xrightarrow{\approx}$ O_2:Professor

In this light, Q is a disjunction of two queries Q_1 = ¬TechnicalStaff, and Q_2 = Lecturer. *SemRef* algorithm first deals with Q_1. At first S_1C_1, S_2C_1 and Neg_S_2C_1 are set to empty. Since there is no equivalence correspondence concerning the concept TechnicalStaff in query Q_1, the set S_1C_1 remains empty (S_1C_1 = { }). On the other hand, there are some unusual correspondences (closeness and generalization), thus the set S_2C_1 receives {Assistant, Faculty, Worker}. Besides, Neg_S_2C_1 is set to {AdministrativeStaff}, due to the negation over the concept in Q_1 and the disjointness correspondence between this concept and AdministrativeStaff in P_2. After analyzing the correspondences and including matching concepts in the related sets, the algorithm checks the sets which are not empty. In this case, S_1C_1 is empty, so there is no exact reformulation for Q_1. Nevertheless, there is an enriched reformulation for Q_1. $Q_{1\_enriched}$ is set to [¬Assistant ⊔ ¬Faculty ⊔ ¬Worker ⊔ AdministrativeStaff]. Such partial result is produced by the Build_Enriched_Reformulation function, which provides the negation over the concepts from set S_2C_1 and gets the concept provided by the set Neg_S_2C_1.

Following the same idea, Q_2 = Lecturer is reformulated. At first, S_1C_1 is set to empty, since, again, there is no equivalence correspondence concerning Lecturer; S_2C_1 receives {PostDoc, Professor, Faculty}, considering the closeness and generalization correspondences, and Neg_S_2C_1 is set to empty, since there is no negation over this current concept. As a result, Q_{2_exact} is empty, but $Q_{2\_enriched}$ is set to [PostDoc $\sqcup$ Professor $\sqcup$ Faculty].

Still in example 2, in order to build the final reformulations, we have to check the reformulation mode the user has chosen. Since it was RESTRICTED, usually, we would not build the enriched reformulation, but, in this case, Q_{exact} was empty, so the algorithm builds the enriched one. Therefore the final query reformulations are:

- $Q_{exact} = \varnothing$
- $Q_{enriched}$ = [[¬Assistant $\sqcup$ ¬Faculty $\sqcup$ ¬Worker $\sqcup$ AdministrativeStaff]] $\sqcup$ [[PostDoc $\sqcup$ Professor $\sqcup$ Faculty]].

The third example concerns Q = [AdministrativeStaff $\sqcap$ Professor], now submitted in Peer P_2. To perform query reformulation, the algorithm considers the set of correspondences presented in Table 5.4. Furthermore, we assume that the user has set to TRUE only *approximate* variable and EXPANDED value to reformulation mode.

Table 5.4. Some Semantic Correspondences between O_2 and O_1

O_2 Concept	Semantic Correspondences
AdministrativeStaff	O_2:AdministrativeStaff $\xrightarrow{\approx}$ O_1:Faculty O_2:AdministrativeStaff $\xrightarrow{\perp}$ O_1:TechnicalStaff O_2:AdministrativeStaff $\xrightarrow{\sqsupseteq}$ O_1:SystemsStaff O_2:AdministrativeStaff $\xrightarrow{\sqsupseteq}$ O_1:ClericalStaff O_2:AdministrativeStaff $\xrightarrow{\sqsubseteq}$ O_1: Worker O_2:AdministrativeStaff $\xrightarrow{\equiv}$ O_1:AdministrativeStaff
Professor	O_2:Professor $\xrightarrow{\approx}$ O_1:Lecturer O_2:Professor $\xrightarrow{\sqsupseteq}$ O_1:FullProfessor O_2:Professor $\xrightarrow{\sqsupseteq}$ O_1:AssistantProfessor O_2:Professor $\xrightarrow{\sqsubseteq}$ O_1:Faculty

This submitted query is indeed a conjunction of two concepts, what implies in one query Q_1 = [AdministrativeStaff $\sqcap$ Professor]. Considering the first concept C_1 = AdministrativeStaff, initially, S_1C_1, S_2C_1 and Neg_S_2C_1 are set to empty. Due to existing equivalence correspondences (Table 5.4), S_1C_1 is set to {AdministrativeStaff} and, due to closeness correspondences, S_2C_1 is set to {Faculty}. The set Neg_S_2C_1 remains empty, since there is no negation over the concept. Now considering the second concept C_2 = Professor, the algorithm states S_1C_2 = { }, due to the fact that there is no equivalence correspondence, S_2C_2 = {Lecturer}, because of the closeness correspondence and Neg_S2C2 = ∅, since there is no negation over the concept. Since one of the S_1C_n is empty, all the conjunction for exact reformulation fails. Thus, Q_{1_exact} = ∅. The algorithm builds $Q_{1\_enriched}$ = [[Faculty] ⊓ [Lecturer]], using the Build_Enriched_Reformulation function. Since there is only one query (Q_1) in Q, the final exact and enriched reformulations are the same as Q_{1_exact} and $Q_{1\_enriched}$. Therefore, the final reformulations are:

- Q_{exact} = ∅
- $Q_{enriched}$ = [[Faculty] ⊓ [Lecturer]]

5.6. Comparative Analysis

As we have discussed in Chapter 2, query reformulation techniques have been addressed in different computational environments. In this section, we provide a comparison between the approaches we have covered in Chapter 2 and ours. Table 5.5 summarizes the main features of the different query reformulation approaches as well as of *SemRef* ‘s.

The works of Necib [2007], Kostadinov [2007] and Stuckenschmidt et al. [2005] are similar to ours in what concerns the employment of some kind of query enrichment, although in the first work, the used knowledge is obtained from the domain ontology while in the second one, it is gathered from the user profiles. Instead, in our work, we acquire semantic knowledge mainly from the semantic correspondences among the peers’ ontologies and also from the context of the user, of the query and of the environment. The third work is similar to ours in trying to establish concept approximation and reformulating the query in such a way that these close concepts may be included. On the other hand, they only consider approximation by means of generalization and specialization (upper and lower bounds). Instead, we consider other kinds of correspondences which allow different levels of approximation. Also, they perform query relaxation, i.e., they simplify the query by putting away constraints that cannot be matched in the target peer. In our approach, constraints by means of roles definition are maintained and matched through semantic correspondences concerning them.

Like our approach, the works of Xiao and Cruz [2006], Calvanese and his group [2004] and Adjiman et al. [2007] consider mappings between peer ontologies. On the other hand, Piazza [Halevy et al. 2003] considers mappings among peer schemas. However, in most of the referred works, the mappings considered are restricted to equivalence and subsumption (SomeRDFS also considers disjunction). Therefore, we go one step further in our process as we also use other kind of semantic reformulation

Table 5.5: Comparative Analysis of Query Reformulation Approaches with Ours

Approach	Environment	Representation Model	Formalism	Query Language	Mapping/Correspondence Types	Semantics Usage	Reformulation Rules
[Necib 2007]	Single Databases	Relational	Term Rewriting Systems	SQL	Equivalence (between database schema and ontology)	Ontology as additional knowledge	Extension Rules Reduction Rules
[Kostadinov 2007]	Mediator-based System	Relational	Conjunctive Query	SQL	LAV mappings	User Profiles	Enrichment Rules Translation rules using LAV approach
Piazza [Halevy et al. 2005]	PDMS	Relational and XML	Conjunctive Query	XQuery or Conjunctive Query	Equivalence, Inclusion and Definitional Mappings	Metadata in a Catalog	Translation Rules, using GAV/LAV approaches
OPDMS [Xiao and Cruz 2006]	PDMS	RDF	FOL (First Order Logic)	Conjunctive RQL Query	Equivalence, Broader, Narrower, Union and Intersection	Mapping Ontology	Translation Rules
WTA [Calvanese et al. 2004]	PDMS	Knowledge-based (First Order Logic - FOL)	FOL (First Order Logic)	FOL Query	Subsumption between classes, Participation of classes in roles, Mandatory participation of classes in roles	___	Translation Rules
SomeRDFS [Adjman et al. 2007]	PDMS/Semantic Web	RDF	DL (Description Logics) and FOL (First Order Logic)	FOL Query	Equivalence, Inclusion, Disjunction	___	Translation Rules
Concept Approximation [Stuckenschmidt et al. 2005]	Weakly-Structured Environments	Terminological Knowledge base using DL	DL (Description Logics)	Boolean Query	Equivalence, Specialization (Lower Approximation), Generalization (Upper Approximation)	Terminological reasoning and query relaxation	Concept Approximation in terms of Lower and Upper Bounds
SemRef	**Dynamic Distributed Environments; OPDMS**	**OWL**	**DL (Description Logics)**	**$\mathcal{ALC}$/DL SPARQL**	**Equivalence, Specialization, Generalization, Closeness, Disjointness, Aggregation (PartOf) and Aggregation (WholeOf)**	**Domain Ontology, Semantics underlying Correspondences and Contextual Information**	**Exactness and Enrichment Rules**

rules (e.g. disjointness and closeness) which are obtained from the set of semantic correspondences between two peer ontologies. In fact, to the best of our knowledge, *closeness* is a kind of semantic correspondence that is not found in any related work. As presented in Section 5.4, when users enable approximation, close correspondences may make a difference and provide expanding concepts related in a given context. Another difference concerns the use we make of *disjointness* correspondences when there are negations to deal with. We are able to directly obtain the disjoint concept as a solution to the negation of the original concept.

Furthermore, none of these works deal with contextual information. Differently, our work produces reformulated queries taking into account the context of the user (e.g., preferences), of the query (e.g., mode, semantics) and of the environment (e.g., peers availability). Since these works do not deal with context nor with semantics around the query reformulation, most of them are not concerned with producing enriched reformulations. Exceptions are the work of Kostadinov and, somehow, Stuckenschmidt and his group. Our work, on the other hand, prioritizes the generation of exact reformulations, but, depending on the context and reformulation execution mode, it also generates an enriched version, which brings more relevance to the set of reformulated queries, and, consequently, a set of expanded answers to the users.

5.7. Concluding Remarks

One key issue for query answering in dynamic distributed environments is the reformulation of a query posed at a peer into another one over a target peer. A problem that still persists is the fact that concepts from a source peer do not always have exact corresponding concepts in a target one, what results in an empty set of reformulations and, possibly, no answer to users. Depending on the users' preferences, it may be better to produce an adapted/enriched query reformulation and, consequently, close answers than no answer at all. In this chapter, we presented *SemRef*'s approach as a solution to such problem. *SemRef* brings some advantages in relation to other approaches:

i. It brings together the concepts of query reformulation and query enrichment within a dynamic distributed ontology-based environment;
ii. It focuses on reformulating a query in terms of one or two kinds of reformulation, by means of exact (the best) and/or enriched reformulations;
iii. In order to accomplish query reformulation, it makes use of a set of semantic correspondences, namely, equivalence, specialization, generalization, closeness, part-of, whole-of and disjointness, providing different levels of concept approximation;

iv. It uses closeness as a way to provide expanding concepts related in a given context. Also, it uses disjointness to deal with negations, i.e., it directly obtains the disjoint concept as a solution to the negation over an original concept;

v. It performs query reformulation considering the context of the users, of the query and of the environment. More specifically, context of users are acquired from the set of enriching variables that users may define; context of queries are identified from their semantics and from the reformulation mode; context of the environment is gathered on-the-fly from the peer where the query has been submitted and its available neighbor peers; and

vi. Considering semantics usage, it can provide users with a set of expanded answers as a result of the execution of exact and/or enriched reformulations.

It is worth noting that we addressed our problem in a setting based on just two peers, although our approach can also be used in an extended scenario composed by a set of diverse peers. In fact, query reformulation strategies and query routing mechanisms [Montanelli and Castano 2008; Faye et al. 2007; Mandreoli et al. 2006] have a great influence on each other. In our approach, we consider that every peer P_i maintains a neighborhood $N(P_i)$ selected from the set of existing peers in the setting. Our global query management process allows to specify a user query at some peer P_i, and to compute it in a fully decentralized manner involving the set of relevant neighbor peers. A given query is submitted in Peer P_i and reformulated in P_i's neighbors, and in its neighbors that are also considered relevant, according to a routing policy which verifies the *path_length* defined variable. In this sense, a submitted query must be reformulated in such a way that it is possible to ensure effective query routing, preserving the query semantics at the best possible level of approximation. This is possible by means of the semantic correspondences among the peers, and also by means of preserving the choices of the user. Thus, at each query reformulation, the enriching variables and query reformulation mode values are also propagated and therefore taken into account.

CHAPTER 6

"Begin, be bold and venture to be wise".
Horace

Implementation Issues

We have instantiated our approach in a Peer Data Management System (PDMS), where ontologies are used as uniform conceptual representations of peer schemas. In this chapter, we briefly describe its architecture in order to explain how the implementation of the *SemRef* approach has been accomplished. Then, we provide details of the *SemRef*'s implementation. In particular, we show how the query reformulation module works and we discuss the solutions we gave for bridging the gap between $\mathcal{ALC}$/DL and SPARQL semantics, thus providing users with queries in both languages.

The chapter structure is as follows. In Section 6.1 we describe our running setting. In Section 6.2, we discuss the implementation issues regarding *SemRef*'s approach. In Section 6.3 we present some concluding remarks.

6.1. Running Setting

We have instantiated our query reformulation approach in a PDMS. The system we use is an Ontology-based Peer Data Management System (OPDMS), since it adopts an ontology-based approach to assist relevant issues in peer data management, e.g., query answering and peer connectivity. In the following, we describe its architecture and how mapping expressions and correspondences are defined and dealt with.

6.1.1. System Architecture

The system, named SPEED (Semantic PEEr-to-Peer Data Management System), employs a mixed network topology (DHT and super-peer) in order to exploit the strength of both topologies [Pires et al. 2008]. A DHT network [Sung et al. 2005] is used to assist peers with common interests to find each other and form semantic communities. Within a community, peers are arranged in a super-peer topology [Yang and Garcia-Molina 2003].

As shown in Figure 6.1, three distinct types of peers are considered in the system: *data peers*, *integration peers*, and *semantic peers*. A *data peer* represents a data source sharing structured or semi-structured data with other data peers in the system. In Figure 6.1, I_1D_1 and I_1D_2 are examples of data peers. Data peers are grouped within *semantic clusters* according to their semantic interest. A semantic interest includes the peer's interest theme and a local peer ontology. The interest theme is an abstract description of the peer's semantic domain, whereas the local peer ontology (LO) describes the peer's exported schema. To ensure correct query answering, such local ontology representation preserves the structure and the integrity constraints (e.g., relational foreign keys) expressed on the peer's schema.

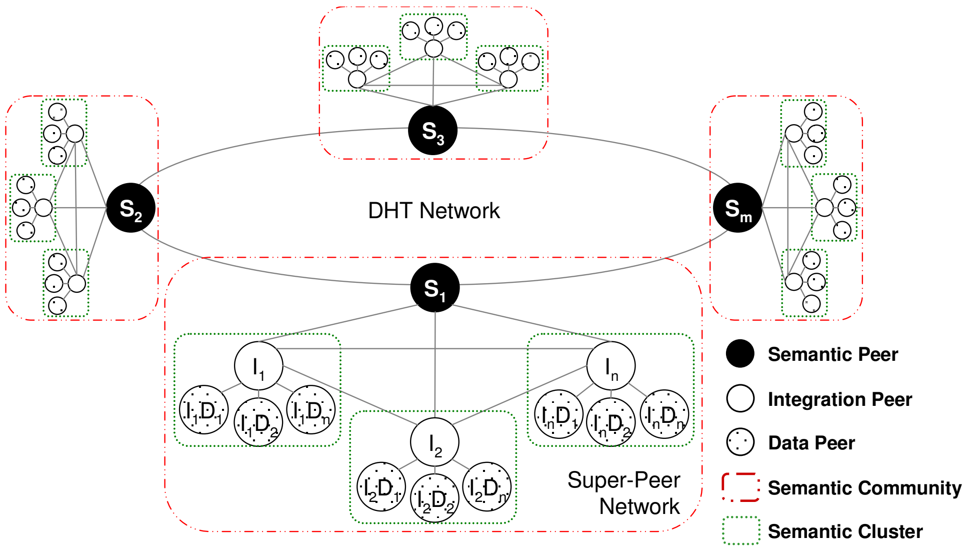


Figure 6.1 Overview of SPEED's architecture [Pires et al. 2008]

Each *semantic cluster* has a special type of peer with higher computational capacity, named *integration peer*. Actually, integration peers are data peers with higher availability, network bandwidth, processing power, and storage capacity. Such peers are responsible for tasks like managing data peers' metadata, query answering, and data integration. In Figure 6.1, I_1 is the integration peer of the semantic cluster composed by the data peers I_1D_1, I_1D_2, and I_1D_n.

An integration peer maintains a cluster ontology (CLO), which is obtained through the merging of the local ontologies representing data peers' and integration peer's exported schemas. Integration peers communicate with a *semantic peer*, which is responsible for storing and offering a community ontology (CMO) containing elements of a particular knowledge domain. Semantic peers are responsible for managing integration peers' metadata. In Figure 6.1, S_1 is an example of a semantic peer. A set of clusters sharing semantically similar interests composes a *semantic community*.

6.1.2. Mapping Expressions and Correspondences

Since ontologies are used as uniform conceptual representation of peer schemas in SPEED, we need to establish correspondences among these ontologies in order to allow query reformulation. To this end, we have instantiated our correspondences definition (described in Chapter 4) in SPEED's architecture. In order to describe how correspondences and mapping expressions are dealt with in such setting, we formalize SPEED as a distributed data source system, denoted by S = {{P}, {C}, {M}} composed by a set of peers ({P}), semantic correspondences among them ({C}) and a set of mapping expressions ({M}) inside a cluster, as depicted in Figure 6.2.

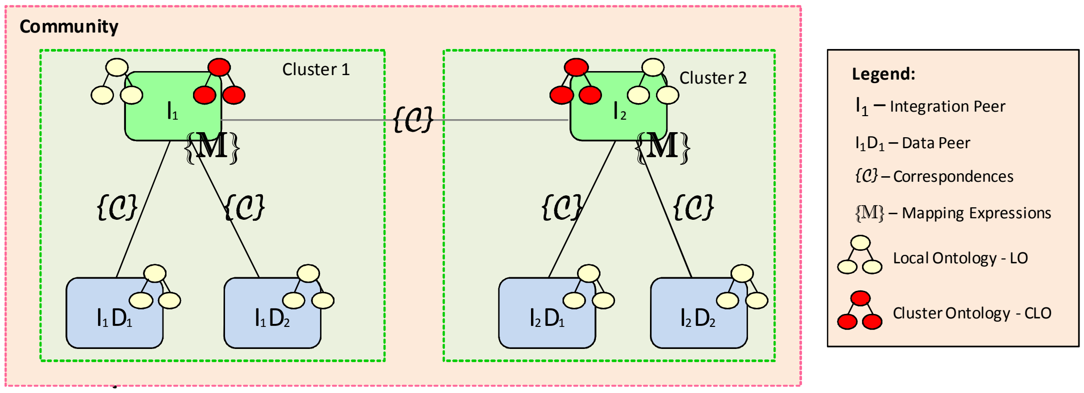


Figure 6.2 Mapping Expressions and Correspondences in SPEED

When a requesting peer asks to enter the system, the discovery of its semantic community is taken through the use of knowledge domain keywords, using the DHT network. After that, the cluster to which it will be assigned is found out through ontology matching. More precisely, a semantic matchmaker module performs a matching between the local ontology - LO (requesting peer) and some cluster ontologies - CLOs (integration peers) of a community, producing a similarity degree between them. The requesting peer takes part of a cluster if the similarity function produces a value higher than a pre-defined cluster threshold. As a result of such matching, we have an alignment, i.e., a set of correspondences among the concepts and properties of both ontologies (LO and CLO). In Figure 6.2, this set of *correspondences* is denoted by {C} and each correspondence is directionally defined from the CLO concept or slot (property) to the LO concept or slot.

The most suitable cluster for a requesting peer is the one whose similarity function produces the highest semantic similarity value between the LO and the corresponding CLO. Inside a cluster, *mapping expressions* ({M}) are defined between a CLO concept and views over the data peers, as shown in Figure 6.2. More specifically, mapping expressions are built using the semantic information gathered from the correspondences between the CLO and each data peer LO, following a GAV-like strategy. Thus, when querying data inside a cluster, mapping expressions are used as input of the (unfolding) algorithm to reformulate user queries into sub-queries executed at data peers. Returned results from data peers are then combined by the integration peer. A mapping expressions M can be defined as one of the following forms (adapted from [Lóscio 2003]):

$$\text{CLOConcept} \equiv \text{QueryDP}_1 \text{ Op}_j \text{ QueryDP}_2 \ldots \text{Op}_j \ldots \text{QueryDP}_i$$

$$\text{CLOConcept} \sqsupseteq \text{QueryDP}_1 \text{ Op}_j \text{ QueryDP}_2 \ldots \text{Op}_j \ldots \text{QueryDP}_i$$

where:

- CLOConcept is a concept in the Cluster Ontology
- QueryDP_i in M holds a *query* (view) over a data peer P_i and is responsible for computing its contents; and
- Op_j represents an *operator* (e.g., union, intersection, difference) that may be applied between two queries over data peers (QueryDP_i).

In this sense, we consider the vision inside a cluster as a generalization of a data integration system. In other words, a cluster acts like a mediator-based integration system, where we have a mediated schema represented as a cluster ontology and a set of data sources (data peers) that are mapped to this single CLO. We refer the reader to the work of Lóscio [2003] for more details about mediator-based integration systems.

Furthermore, integration peers are semantically related to a set of other ones. We call this group of semantically related integration peers *semantic neighbors*. In order to determine this set of semantic neighbors, a semantic similarity measure function is applied among pairs of integration peers and those with higher values (than a given threshold) are set as neighbors of a given integration peer. Thus, for instance, considering an integration peer I_1 and a set of other integration peers I_2, I_3 and I_4, each one with its own CLO (CLO_1, CLO_2, CLO_3 and CLO_4), we assume that a similarity function takes as input two CLOs (from I_1 and another integration peer) and generates a value in the range of 0-1. If such value is higher than 0.4, for example, the corresponding integration peer is a semantic neighbor of I_1. In Table 6.1, we provide an example of this strategy and of these results. According to this example, I_1 is a semantic neighbor of I_3 and I_4, since the output of their similarity evaluation is higher than 0.4.

Table 6.1 Semantic Neighboring of a Peer I_1

Input 1	Input 2	Similarity Measure	Semantic Neighbor?
CLO_1	CLO_2	0.3	No
CLO_1	CLO_3	0.7	Yes
CLO_1	CLO_4	0.8	Yes

We also consider a set of correspondences between pairs of semantic neighbors (integration peers). A correspondence between two CLOs occurs directionally from one cluster to the other and vice-versa. This type of correspondence is obtained when two CLOs are matched in order to verify the existing similarity degree between them.

In summary, in SPEED, we have two types of *correspondences*:

i. correspondences between local ontology (LO) and cluster ontology (CLO)' elements (concepts and properties), inside a cluster; and
ii. correspondences between semantic neighbor integration peers, i.e., between CLOs' elements (concepts and properties), inside a community.

Both kinds of correspondences are generated by the matching processes (between CLOs and LOs and between two different CLOs). As an illustration of mapping expressions and correspondences definition, assume we have a semantic community composed by two clusters – Cluster_1 and Cluster_2, as shown in Figure 6.2. Each cluster has one integration peer (I_1 and I_2) and two respective data peers. The schemas are concerned with an "Education" domain and include concepts such as Professor and Instructors.

In Cluster_1, considering that Professor is a concept belonging to CLO_1, we have the following set of correspondences between CLO_1 and the two data peer local ontologies:

- I_1.Professor $\xrightarrow{\equiv}$ I_1D_1.Prof
- I_1.Professor.Name $\xrightarrow{\equiv}$ I_1D_1.Prof.Name
- I_1.Professor.Email $\xrightarrow{\equiv}$ I_1D_1.Prof.Address.Email
- I_1.Professor $\xrightarrow{\sqsupseteq}$ I_1D_2.Instructor
- I_1.Professor.Name $\xrightarrow{\equiv}$ I_1D_2.Instructor.Name
- I_1.Professor.Email $\xrightarrow{\equiv}$ I_1D_2.Instructor.Email

A mapping expression between CLO_1 and the LOs may be stated as follows:

- I_1.Professor ≡ $\text{QProf}_1 \sqcup \text{QInst}_1$

Where QProf_1 is a query (view) over data peer I_1D_1 regarding the concept Prof, and QInst_1 is a query over data peer I_1D_2, regarding the concept Instructor.

In the same way, as a result of matching processes between CLO_1 and CLO_2, we have the following set of correspondences between the cluster ontologies:

- I_1.Professor $\xrightarrow{\equiv}$ I_2.Professeur
- I_1.Professor.Name $\xrightarrow{\equiv}$ I_2.Professeur.Nom
- I_1.Professor.Email $\xrightarrow{\equiv}$ I_2.Professeur.Email

Nevertheless, we do not have mapping expressions among cluster ontologies, since we do not use view-based query rewriting in such level. Instead, we perform query reformulation between two neighbor integration peers, i.e., we basically reformulate a query posed at a source peer in terms of a target peer. Currently, returned results obtained from the execution of queries over integration peers are integrated by means of the *union* operator.

In fact, in our query reformulation approach, we abstract the mediation level inside the clusters, and work with the *integration peers level*, considering its unstructured pure P2P topology. The underlying reason is that the implementation concerning the cluster level, i.e., the mediation level is already done in the *Integra* data integration system [Lóscio 2003]. Such implementation is being reused in SPEED to provide query reformulation inside the clusters.

Therefore, in our query reformulation working scenario, each peer I_i (hereafter called P_i) is an *integration peer* belonging to a set of peers $\{P\}$. Each peer P_i is considered a server when providing data, a router when forwarding queries, and a client when receiving data from other peers. Every peer P_i maintains a semantic neighborhood $N(P_i)$ selected from the set of integration peers.

6.2. *SemRef* Implementation

We have developed the *SemRef* approach within the query submission and execution module for SPEED. As previously explained, we focus on reformulating and executing queries among neighbor integration peers which are linked through semantic correspondences. Between each pair of neighbor integration peers, there is a similarity measure (computed previously) [Pires 2009] which may be used for routing strategies.

The data management module of each integration peer P_i is responsible for managing queries and answers. Upon receiving a query, the data management module performs the following tasks:

I. Query Handling. Analyzes the query and extracts its semantics by means of its goal, required entities and operators, and important parameters;
II. Query Translation. Matches the query to its own schema to try to execute it;
III. Peer Selection. Selects its own relevant neighbors according to the semantic correspondences and similarity measure between itself and its neighbors;
IV. Query Reformulation. Reformulates the query put to the current peer using the schemas (ontologies) of the selected relevant neighbors; and
V. Query Routing: autonomously forwards the query to the previously defined relevant neighbors according to the *path_length* variable value (it states the number of subsequent routings that can occur once the first routing with reformulation has been initiated).

Since our focus is on reformulating queries, we do not provide details regarding query routing. Besides, in order to provide query reformulation using semantic correspondences, at first, we had to develop a semantic matcher which has been responsible for identifying them. In the same way, we had to code our context ontology, so we could store contextual elements. As a result, the *SemRef* approach has been built considering these two important artifacts. We show our implementation focus in Figure 6.3, through the query module architecture. A graphical-based (GUI) interface is provided through which users submit their queries and view obtained answers. We give more details about the interface in Section 6.2.2.

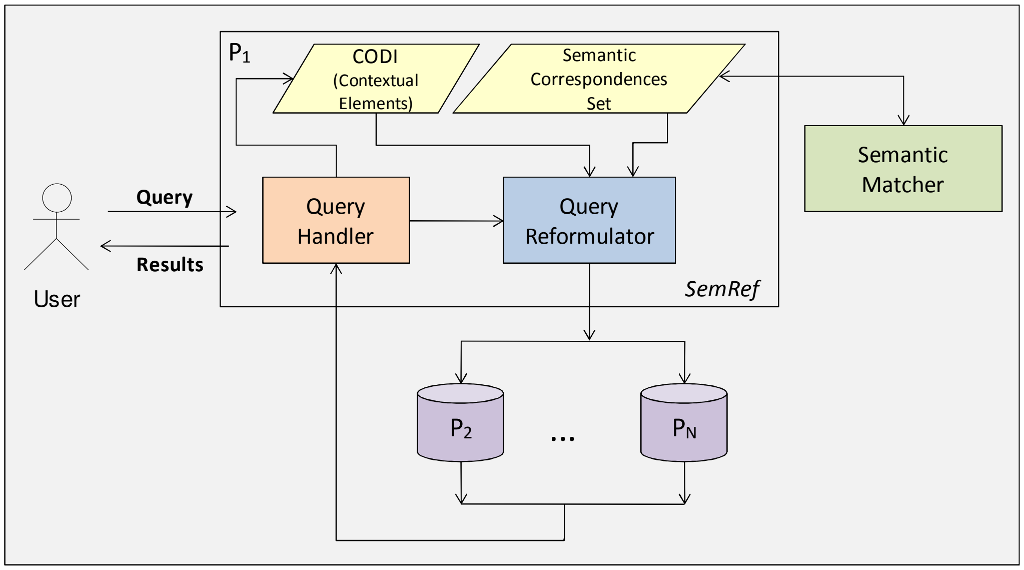


Figure 6.3 Query Module Architecture

In the following, we briefly introduce the main components: CODI, Semantic Correspondences Set, Query Handler, Query Reformulator and Semantic Matcher.

- CODI: *context ontology* where we store contextual elements such as user preferences (enriching variables and query reformulation mode), query entities and operators, path-length and submission peer. CODI has been coded in OWL.
- Semantic Matcher: receives as input two matching ontologies – O_1 and O_2, as well as a domain ontology to be used as background knowledge. Then, it applies the set of semantic rules explained in Section 4.2 in order to derive the type of semantic correspondence between O_1 and O_2 elements.
- Semantic Correspondences Set: concerns the alignment resulting from the semantic matcher process. This set of semantic correspondences has been designed to be stored either in a database or in an OWL file.
- Query Handler: is responsible for analyzing query semantics, identifying its required entities, operators and goal. This component is also responsible for receiving query answers from remote peers, integrating the results and presenting the user with the final one.

- Query Reformulator: is actually the key component of our *SemRef* approach, since it reflects the algorithm presented in Section 5.4. In this sense, it verifies the surrounding contextual elements and existing semantic correspondences between source and target peers and reformulates an original query producing one or two reformulated queries (Q_{exact} and/or $Q_{enriched}$).

Although our implementation has been designed to consider any set of neighbor integration peers, for the sake of simplicity, as already done in Chapter 5, we have only considered two peers that wanted to communicate, sending and receiving queries. Thus, our implementation has been performed over two integration peers – a source and a target. Next, we present the specification and implementation issues of both semantic matcher and *SemRef* components, including related details regarding the others.

6.2.1. Semantic Matcher

The semantic matcher is part of a general semantic matching process which uses a domain ontology (DO) to complement linguistic and structural matching techniques. It uses a DO as background knowledge and applies the described set of semantic rules (Section 4.1) to derive semantic correspondences for two matching ontologies. These ontologies may be of different levels of granularity (in terms of size, partition of concepts and/or conceptual organization). Both matching ontologies and domain ontology are coded in the same language, i.e., OWL[16]. OWL has been chosen due to the fact that it is nowadays a standard uniform notation for representing and storing ontologies. Furthermore, peer ontologies are also terminologically normalized in a pre-matching step where their element names are adjusted to become compatible with the element names found in the DO.

In order to provide a better understanding of the semantic matcher's goals, we have specified a use case diagram. Such diagram presents its main functional requirements and is depicted in Figure 6.4. As non-functional requirements, we have considered the following: (i) the matcher should be platform independent; (ii) it should run on SPEED system supporting the matching between two CLOs and/or between a CLO and a LO; (iii) the matcher's interface should reflect the matching process, thus allowing the administrator to check its overall execution.

The semantic matcher has been implemented [Pereira 2008] in Java. In order to provide ontology manipulation and reasoning, we have used Jena[17] and OWL API[18]. Jena is a Java framework which provides a programmatic environment for RDF, OWL

[16] http://www.w3.org/TR/owl-features/
[17] Jena, http://jena.sourceforge.net/
[18] OWL API, http://owlapi.sourceforge.net/

and SPARQL and includes a rule-based inference engine. OWL API is a Java interface and implementation for OWL.

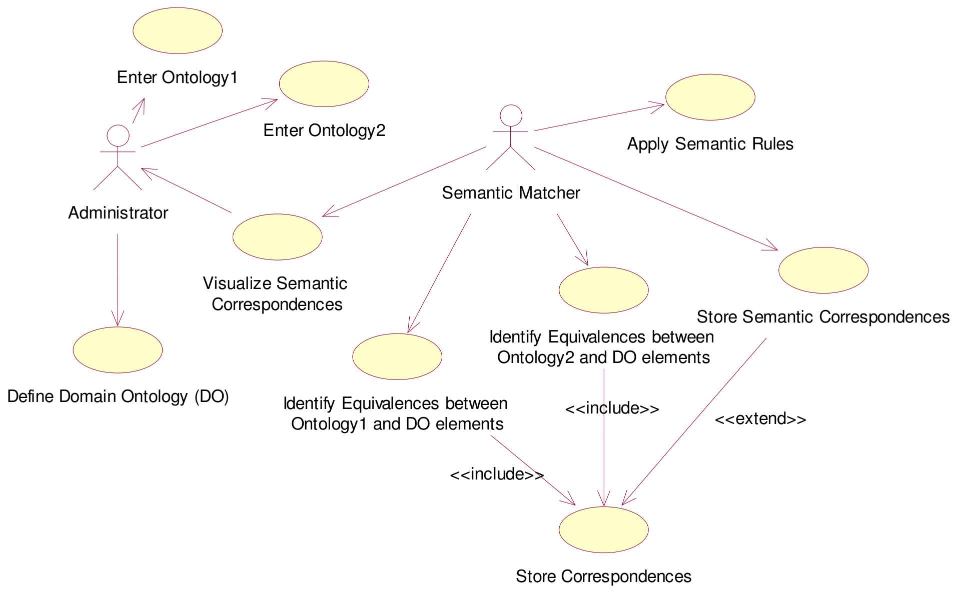
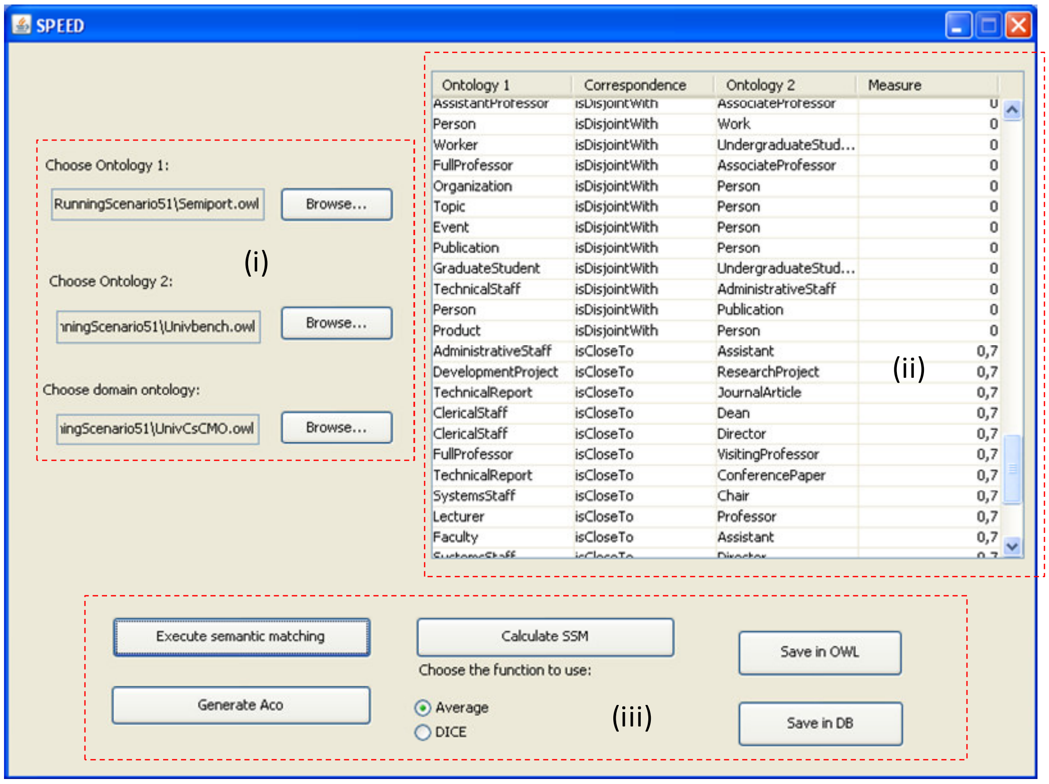

Enter Ontology1
Enter Ontology2
Administrator
Define Domain Ontology (DO)
Visualize Semantic Correspondences
Semantic Matcher
Apply Semantic Rules
Store Semantic Correspondences
Identify Equivalences between Ontology2 and DO elements
Identify Equivalences between Ontology1 and DO elements
<<include>>
<<include>>
<<extend>>
Store Correspondences

Figure 6.4 Use Case Diagram for Semantic Matcher

This version is able to identify all semantic correspondences defined in section 4.1.1 for concepts, allowing the generation of 1:1 correspondences. However, regarding properties, we have restricted the correspondence identification to equivalence, specialization and generalization. In order to accomplish that, we use the hierarchy of properties provided by the DO. Figure 6.5 shows a screenshot of the tool's main window that is split into three parts: (i) an area for choosing matching ontologies; (ii) an area for depicting the resulting semantic correspondences and their respective weights; and (iii) an area for executing the main options, concerned with identifying the semantic correspondences, generating the ACO alignment (resulting set of correspondences identified by the linguistic-structural and semantic matcher) and calculating the global similarity measure. These two latter functions are described in Pires [2009].

At first, the administrator chooses the matching ontologies and points out the domain ontology to be used (in the future, this step will be accomplished in an automatic way by the PDMS). The semantic matcher derives the semantic correspondences and displays them in the screen. The administrator verifies the resulting set and then stores it in an OWL file. Figure 6.6 presents an excerpt from an OWL file with some semantic correspondences between elements of the ontologies chosen in Figure 6.5 (the ontologies are also described in Section 5.5). In order to identify such correspondences, we have set the *thresholdroot* as 10% of the DO's height, and the *thresholdcommonancestor* as 10% of the height of this concept in relation to the sub-tree where it is found in the DO.



Choose Ontology 1: RunningScenario51\Semiport.owl

Choose Ontology 2: nningScenario51\Univbench.owl

Choose domain ontology: iingScenario51\UnivCsCMO.owl

Choose the function to use: [✓ Average] DICE

Ontology 1	Correspondence	Ontology 2	Measure
AssistantProfessor	isDisjointWith	AssociateProfessor	0
Person	isDisjointWith	Work	0
Worker	isDisjointWith	UndergraduateStud...	0
FullProfessor	isDisjointWith	AssociateProfessor	0
Organization	isDisjointWith	Person	0
Topic	isDisjointWith	Person	0
Event	isDisjointWith	Person	0
Publication	isDisjointWith	Person	0
GraduateStudent	isDisjointWith	UndergraduateStud...	0
TechnicalStaff	isDisjointWith	AdministrativeStaff	0
Person	isDisjointWith	Publication	0
Product	isDisjointWith	Person	0
AdministrativeStaff	isCloseTo	Assistant	0,7
DevelopmentProject	isCloseTo	ResearchProject	0,7
TechnicalReport	isCloseTo	JournalArticle	0,7
ClericalStaff	isCloseTo	Dean	0,7
ClericalStaff	isCloseTo	Director	0,7
FullProfessor	isCloseTo	VisitingProfessor	0,7
TechnicalReport	isCloseTo	ConferencePaper	0,7
SystemsStaff	isCloseTo	Chair	0,7
Lecturer	isCloseTo	Professor	0,7
Faculty	isCloseTo	Assistant	0,7
SystemsStaff	isCloseTo	Director	0,7

Figure 6.5 The Semantic Matching Tool Interface

```
<rdf:RDF
   <rdf:Description rdf:about="http://swrc.ontoware.org/ontology/portal#UndergraduateStudent">
  <j.0:isDisjointWith>http://www.lehigh.edu/~zhp2/univ-bench.owl#Worker</j.0:isDisjointWith>
  <j.0:isDisjointWith>http://www.lehigh.edu/~zhp2/univ-
bench.owl#GraduateStudent</j.0:isDisjointWith>
  <j.0:isPartOf>http://www.lehigh.edu/~zhp2/univ-bench.owl#Course</j.0:isPartOf>
  <j.0:isPartOf>http://www.lehigh.edu/~zhp2/univ-bench.owl#ResearchProject</j.0:isPartOf>
  <j.0:isSuperConceptOf>http://www.lehigh.edu/~zhp2/univ-
bench.owl#Monitor</j.0:isSuperConceptOf>
  <j.0:isSubConceptOf>http://www.lehigh.edu/~zhp2/univ-bench.owl#Student</j.0:isSubConceptOf>
    </rdf:Description>
 <rdf:Description rdf:about="http://swrc.ontoware.org/ontology/portal#TechnicalReport">
  <j.0:isCloseTo>http://www.lehigh.edu/~zhp2/univ-bench.owl#ConferencePaper</j.0:isCloseTo>
  <j.0:isCloseTo>http://www.lehigh.edu/~zhp2/univ-bench.owl#JournalArticle</j.0:isCloseTo>
  <j.0:isSubConceptOf>http://www.lehigh.edu/~zhp2/univ-bench.owl#Article</j.0:isSubConceptOf>
  <j.0:isEquivalentTo>http://www.lehigh.edu/~zhp2/univ-
bench.owl#TechnicalReport</j.0:isEquivalentTo>
 </rdf:Description>
```

Figure 6.6 Some Correspondences between Matching Ontologies

6.2.2. *SemRef* Module

In order to provide a detailed description about the implementation of our query reformulation approach, we first present a use case diagram (Figure 6.7) which shows the functional requirements that have been considered. Such requirements are based on the theoretical basis described previously in this thesis. Besides such functional requirements, we have also considered as non-functional ones the following: (i) the system should be platform independent; (ii) it should run on SPEED integration peers' level; (iii) the query interface should be easy and friendly; and (iv) queries should be formulated by means of $\mathcal{ALC}$/DL, SPARQL syntaxes and/or by using concepts provided by the peer ontology.

There are four actors in the diagram. The first is the *User*, i.e., users which wish to query the system. To this end, they have to set the enriching (*generalize*, *specialize*, *compose*, *approximate)* and *path_length* variables. In addition, they establish how reformulation algorithm will deal with such enriching variables definition by providing the *query reformulation mode*. User preferences will be stored as contextual elements in order to be later verified by the query reformulator.

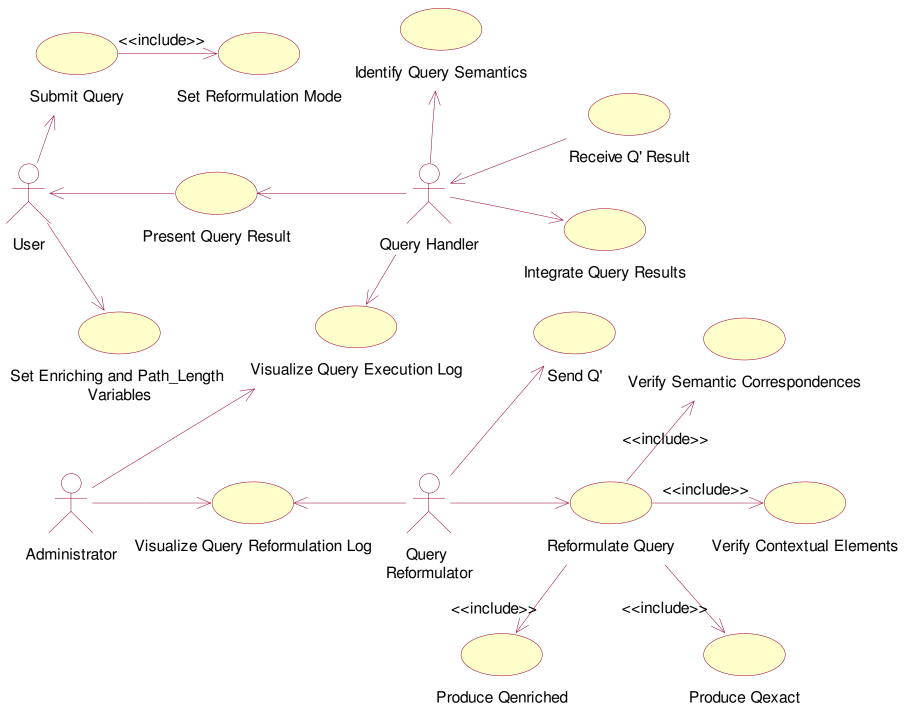


Figure 6.7 Use Case Diagram for *SemRef*

The second actor is the *Query Handler* which is responsible for analyzing the query semantics, identifying its required entities, operators and goal. This module is also responsible for receiving query answers from remote peers, integrating the results and presenting the user with the final one. The third actor – *Query Reformulator* - is the main module of our *SemRef* approach. It verifies the surrounding contextual elements and existing semantic correspondences between source and target peers and reformulates the query producing one or two reformulated queries (Q_{exact} and $Q_{enriched}$). For performance reasons, although it produces one or two reformulations of a given query, it puts both reformulations together in one execution query (Q') and sends it to the target peer.

The fourth actor is indeed the *administrator*. In order to provide details regarding both query reformulation and query execution processes, the system allows verifying a related log. Therefore, the administrator can check whether the reformulation has been done correctly as well as whether query answers have been produced accordingly.

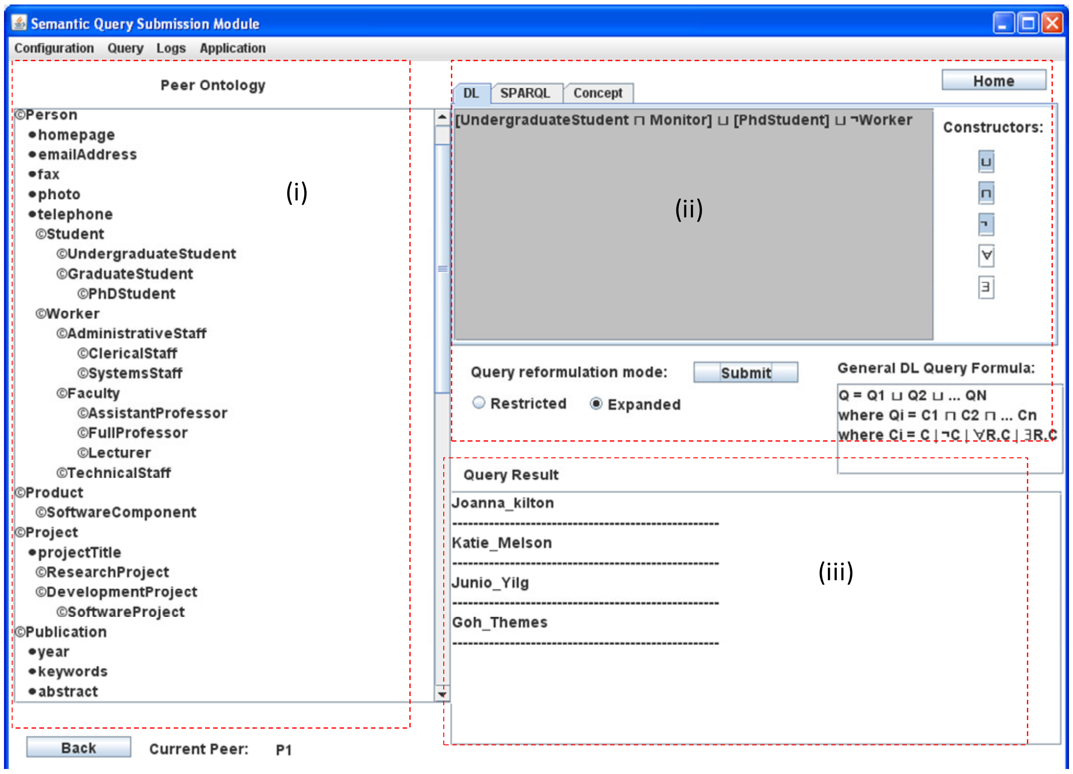
The *SemRef* approach has been developed within a query submission module for our PDMS. Such module has been implemented in Java and intends to provide users with a friendly query submission interface [Neves 2008]. RMI (*Remote Method Invocation)*[19] has been used for peer communication. In addition, we have adopted both Jena and Protégé's API[20] in order to manipulate the underlying ontologies and execute queries over them, through SPARQL language. Figure 6.8 shows a screenshot of the module's main window that is split into three parts: (i) the peer ontology area; (ii) the query formulation area and (iii) the query results area. Queries can be formulated using the concepts provided by the peer ontology, using SPARQL[21] or using $\mathcal{ALC}$-DL. In this current version, we have implemented both DL and SPARQL options.

In this light, after logging in the system, users can set the enriching variables and *path_length*, as described in the activity diagram shown in Figure 5.3. These choices can be changed or updated whenever users require during query session. In addition, to facilitate the process of query formulation, and to provide users with a starting point for query specification, the query interface shows an Ontology Browser component, with the ontology of the current submission peer (see Figure 6.8). In such browser, concepts (labeled by "©") and properties (labeled by "•") of the peer ontology are depicted.

Queries may be formulated using $\mathcal{ALC}$/DL and SPARQL. The reasons underlying these primary choices are: (i) it is important to validate our query reformulation approach using $\mathcal{ALC}$/DL, since it has been formally coded as such; (ii) we execute queries over ontologies that represent data sources, thus in order to facilitate

[19] http://java.sun.com/j2se/1.4.2/docs/guide/rmi/
[20] http://protege.stanford.edu/
[21] http://www.w3.org/TR/rdf-sparql-query/

our tests (simulating the data sources access), we decided to use an ontology query language. Due to the fact that SPARQL is the W3C proposed standard, it has been chosen as our query language.



```
©Person
  •homepage
  •emailAddress
  •fax
  •photo
  •telephone
  ©Student
    ©UndergraduateStudent
    ©GraduateStudent
      ©PhDStudent
  ©Worker
    ©AdministrativeStaff
      ©ClericalStaff
      ©SystemsStaff
    ©Faculty
      ©AssistantProfessor
      ©FullProfessor
      ©Lecturer
    ©TechnicalStaff
©Product
  ©SoftwareComponent
©Project
  •projectTitle
  ©ResearchProject
  ©DevelopmentProject
    ©SoftwareProject
©Publication
  •year
  •keywords
  •abstract
```

[UndergraduateStudent ⊓ Monitor] ⊔ [PhdStudent] ⊔ ¬Worker

Query reformulation mode: ○ Restricted [✓ ◉ Expanded]

General DL Query Formula:

```
Q = Q1 ⊔ Q2 ⊔ ... QN
where Qi = C1 ⊓ C2 ⊓ ... Cn
where Ci = C | ¬C | ∀R.C | ∃R.C
```

```
Joanna_kilton
-----------------------------------------------------
Katie_Melson
-----------------------------------------------------
Junio_Yilg
-----------------------------------------------------
Goh_Themes
-----------------------------------------------------
```

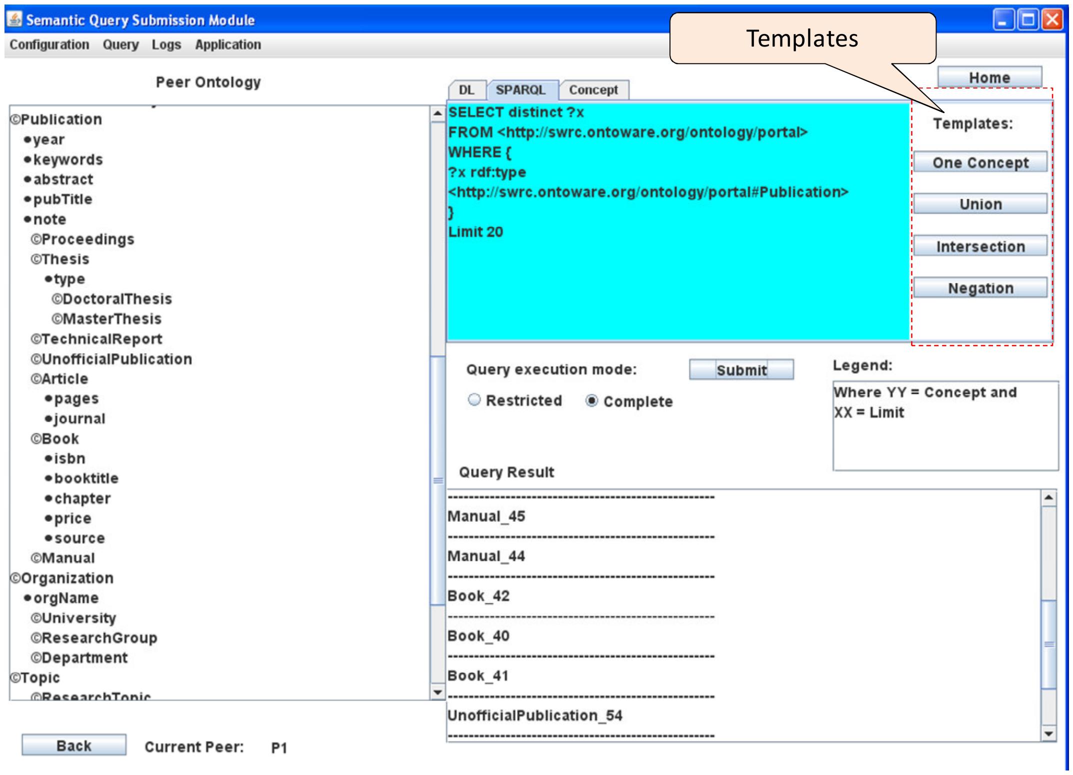
Current Peer: P1

Figure 6.8 Query Interface with DL Query Formulation Option

While typing queries in such formal languages provides a great level of expressivity and control for the user, we know it is also a less user-friendly access interface. The solution to this problem was to create an additional abstraction level that might provide a user friendly way of generating formal queries. Thereby, in both options, we tried to organize the query formulation area, constructors/templates, query reformulation mode and captions in a standard way, thus providing users with an uniform query formulation interface. The former option is shown in Figure 6.8. The latter is depicted in Figure 6.9. As further work, this query interface will be extended with more friendly mechanisms such as providing query formulation by using concepts and properties found out in the peer ontology.

In the DL option, the interface provides basic $\mathcal{ALC}$-DL constructors (disjunction, conjunction, negation, universal and existential quantification), so users are able to formulate the queries more easily. These constructors are graphically depicted by a special node near the query formulation area. Since our *SemRef* approach has been built with $\mathcal{ALC}$/DL constructors, we aimed at providing users with the possibility of also querying using SPARQL in the same way they would do in DL.

Nevertheless, SPARQL is a language with a broad range of constructors and query formats, which causes some problems to be faced: (i) it makes difficult to users to formulate queries without knowing well its syntax and (ii) it requires special effort to bridge DL semantics with SPARQL own semantics. As a way to face such difficulties, we have defined some templates which may be used by users to write their SPARQL queries. To this end, we have investigated some techniques that could be used in order to provide the translation between $\mathcal{ALC}$/DL queries into SPARQL queries. The templates are displayed in a special area, near the corresponding query formulation area.



©Publication
•year
•keywords
•abstract
•pubTitle
•note
©Proceedings
©Thesis
•type
©DoctoralThesis
©MasterThesis
©TechnicalReport
©UnofficialPublication
©Article
•pages
•journal
©Book
•isbn
•booktitle
•chapter
•price
•source
©Manual
©Organization
•orgName
©University
©ResearchGroup
©Department
©Topic
©ResearchTopic

```
SELECT distinct ?x
FROM <http://swrc.ontoware.org/ontology/portal>
WHERE {
?x rdf:type
<http://swrc.ontoware.org/ontology/portal#Publication>
}
Limit 20
```

Query execution mode: ( ) Restricted [✓ Complete]

Legend:
Where YY = Concept and
XX = Limit

Query Result

```
----------------------------------------------------
Manual_45
----------------------------------------------------
Manual_44
----------------------------------------------------
Book_42
----------------------------------------------------
Book_40
----------------------------------------------------
Book_41
----------------------------------------------------
UnofficialPublication_54
----------------------------------------------------
```

Current Peer: P1

Figure 6.9 Query Interface with SPARQL Query Formulation Option

We are able to verify the complete query reformulation and query execution processes through logs. Screenshots of both logs concerning the queries submitted in Figures 6.8 (in $\mathcal{ALC}$/DL) and in Figure 6.9 (in SPARQL) are depicted in Appendix C. Next, we describe our approach for bridging the semantics of $\mathcal{ALC}$/DL with the semantics of SPARQL, so we can provide users with the same query formulation semantics by means of both languages.

6.2.3. Semantics Preserving $\mathcal{ALC}$/DL-to-SPARQL Query Translation

Before we describe our work on translating $\mathcal{ALC}$/DL queries into SPARQL queries, we give a short introduction to the SPARQL query language and the operators of a

SPARQL query that are considered for this translation process. To this aim, we follow the definitions provided by Quilitz and Leser [2008] and Cao [2007]. For a more detailed introduction to SPARQL, we refer the reader to Perez et al. [2006].

A SPARQL query Q is defined as a tuple Q = ⟨E, DS, R⟩, where E is an algebra expression that is evaluated with respect to a RDF graph[22] in a dataset DS. The results of the matching process are processed according to the statements of the result form R (e.g., SELECT). The algebra expression E is built from different graph patterns and can also include solution modifiers, such as PROJECTION, DISTINCT, LIMIT, or ORDER BY. A typical structure of a SPARQL query statement has four parts:

- Prefix: indicates the default prefix;
- From: specifies the RDF dataset to query;
- Select: lists the variables that should be present in the output;
- Where: restriction conditions; and
- Modifiers: if present, these modifiers will change the number of results (limit and offset) and/or their order (order by).

As an example, consider the following query:

```
SPARQL query 1
PREFIX w3Contact: < http://www.w3.org/People/EM/contact#>
SELECT ?name, ?mail
WHERE
  {
   w3Contact:me w3Contact:fullname ?name.
   w3Contact:me w3Contact:mail ?mail.
   FILTER regex (?name , "^Tomaz" )
  }
ORDER BY ?name ?mail
LIMIT 5
```

In the above example, '*?name*' and '*?email*' are variables and '*w3Contact*' identifies the data set against the query will be executed. The keyword FILTER, ORDER BY and LIMIT have the following purposes: FILTER is a restriction on solutions over the whole group in which the filter appears, and FILTER regex is an operation to test strings, based on regular expressions; ORDER BY specifies a sorted result list and LIMIT specifies a limitation on the number of results. Thus, this example query *retrieves the names and email addresses of persons whose names start with "Tomaz"*. The results are ordered by the name, followed by mail. The number of results is limited to five.

[22] http://www.w3.org/RDF/

In this sense, our rationale for translating $\mathcal{ALC}$DL queries into SPARQL ones is to specially consider such presented SPARQL syntax scope as the one we are going to deal with. Explaining better, in order to provide the constructors we find in $\mathcal{ALC}$DL, namely, conjunction, disjunction, negation and quantification[23], we have restricted the usage of SPARQL to elements of its syntax which may provide such constructors semantics. In the following, we present our solutions to each one of the $\mathcal{ALC}$DL constructors we deal with.

a. $\mathcal{ALC}$DL query with one concept

A query with one concept is stated as Q = C, i.e., we want to retrieve the instances of concept C in the ontology. To this end, we have defined the following template:

```
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
Prefix prf: <PeerOntology.owl#>
SELECT distinct ?x
FROM <PeerOntology.owl>
WHERE {
        ?x rdf:type prf:Concept
    }
Limit LL
```

b. $\mathcal{ALC}$DL query with a disjunction of concepts (union)

A query with a disjunction of concepts is stated as $Q = C_1 \sqcup C_2$, i.e., we want to retrieve the instances of the union between C_1 and C_2 in the ontology. To this end, we have defined the following template:

```
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
Prefix prf: <PeerOntology.owl#>
SELECT distinct ?x
FROM <PeerOntology.owl>
WHERE {
        {?x rdf:type prf:Concept1}
        UNION
        {?x rdf:type prf:Concept2}
    }
Limit LL
```

c. $\mathcal{ALC}$DL query with a conjunction of concepts (intersection)

[23] Quantification is currently under development.

A query with a conjunction of concepts is stated as $Q = C_1 \sqcap C_2$, i.e., we want to retrieve the instances which are in the intersection of C_1 and C_2 in the ontology. To this end, we have defined the following template:

```
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
Prefix prf: <PeerOntology.owl#>
SELECT ?x
FROM < PeerOntology.owl>
WHERE {
        ?x rdf:type prf:Concept1 .
        ?y rdf:type prf:Concept2
        FILTER (?x = ?y)
   }
Limit LL
```

d. $\mathcal{ALC}$/DL query with negation over a concept

A query with a negation over a concept is defined as $Q = \neg C$, i.e., we want to retrieve the instances of all the concepts belonging to a given interpretation, with the exclusion of the instances of the concept C. In our work, the current interpretation is provided by the domain ontology, and, more specifically, by the super-concept of the negated concept and its siblings. In SPARQL, negations are interpreted by utilizing the operators \!" and \bound". Thus, we have defined the following template:

```
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>
Prefix prf: <PeerOntology.owl#>
SELECT distinct ?x
FROM < PeerOntology.owl>
WHERE { {?x rdf:type ?y .
          :Concept1 rdfs:subClassOf ?y}
          UNION
         {?x rdf:type ?z .
          :Concept1 rdfs:subClassOf ?y .
          ?z rdfs:subClassOf ?y.
          FILTER (?z != :Concept1) }
   }
Limit LL
```

Following such templates, users are only required to substitute the name of the using concepts (*prf:Concept*$_1$ and/or *prf:Concept*$_2$) and the limit *LL*. Also, it is possible to compose their definitions and formulate conjunctions with negations, disjunctions with negations, disjunctions with conjunctions and any kind of DL constructor composition. We show some examples of the templates-based translation technique in Table 6.2.

Table 6.2 $\mathcal{ALC}$DL-to-SPARQL Translation Examples

$\mathcal{ALC}$DL	SPARQL

Student	SELECT distinct ?x
	FROM <Semiport.owl>
	WHERE { ?x rdf:type <Semiport.owl#Student> }
	Limit 10

TechnicalReport ⊔ Book	SELECT distinct ?x
	FROM <Semiport.owl>
	WHERE {
	{ ?x rdf:type <Semiport.owl#TechnicalReport>} UNION
	{ ?x rdf:type <Semiport.owl#Book> }
	}
	Limit 20

FullProfessor ⊓ Researcher	SELECT distinct ?x
	FROM <Semiport.owl>
	WHERE { { ?x rdf:type <Semipor.owl#FullProfessor>
	.
	?x rdf:type <Semiport.owl#Researcher> } }
	Limit 20

¬GraduateStudent	SELECT distinct ?x
	FROM <Semiport.owl>
	WHERE { {
	?x rdf:type ?y .
	<Semiport.owl#GraduateStudent> rdfs:subClassOf ?y}
	UNION
	{ ?x rdf:type ?z . <Semiport.owl#GraduateStudent>
	rdfs:subClassOf ?y . ?z rdfs:subClassOf ?y .
	FILTER (?z != <Semiport.owl#GraduateStudent>) }}
	Limit 20

6.3. Concluding Remarks

In this chapter, we have presented the SPEED system as our running setting. This system has been considered an OPDMS, since it uses ontologies as a way of enhancing its services. One example is query answering and, more specifically, query reformulation. In our approach, we use ontologies in order to store contextual information, to represent peer schemas and as background knowledge. Ontology matching (done by a semantic matcher) is used to provide peer clustering as well. As a result of such matching, the system identifies the set of semantic correspondences between neighbor peers.

We have presented the main features of the semantic matcher, discussing issues related to the identification of correspondences between concepts and properties. Our current implementation is able to identify all seven semantic correspondences for concepts, although only equivalence, generalization and specialization were feasible for properties. The other kinds of correspondences for properties are under analysis.

The *SemRef* implementation has put the theoretical foundations we have provided in this thesis in practice. Through our implementation solution, we provided users with queries in $\mathcal{ALC}$/DL and SPARQL. To this end, we have bridged the gap between $\mathcal{ALC}$/DL semantics in terms of SPARQL, by creating some templates that match each $\mathcal{ALC}$/DL constructor. In order to facilitate query formulation, we have designed the interface in such a way that users use patterns both to $\mathcal{ALC}$/DL and SPARQL options. We have also defined logs which show how query reformulation was performed as well as query answers have been produced. As a result, administrators can verify the correctness and adequacy of both tasks.

Next chapter describes experiments we have performed to investigate the feasibility of the proposed ideas. In addition, it provides the results we have obtained.

CHAPTER 7

"All life is an experiment.
The more experiments you make the better"
Ralph Waldo Emerson

Experiments and Results

In this chapter, we provide an experimentation of the proposed *SemRef* approach. In order to accomplish such task, we have instantiated the main steps of a methodology belonging to the Experimental Software Engineering [Travassos et al. 2002]. We have defined the experimentation purposes, planned its steps and performed controlled experiments in order to characterize and evaluate our approach. As a consequence, measured results were obtained.

This chapter is organized as follows. In Section 7.1, we provide an overview of the adopted methodology. In Section 7.2, we define the experimentation purposes; in Section 7.3, we present the experimentation design; in Section 7.4, we show the execution of the experiments and, in Section 7.5, we discuss obtained results. Finally, in Section 7.6, we conclude the chapter with some remarks.

7.1. Overview of the Adopted Methodology

Experimentation plays a very important role in evolving scientific knowledge [Basili 2007]. It can produce pieces of evidence to confirm or refute items which are subject of research. However, although experimental studies in software engineering have been carried out for several decades, designers or developers are still at a loss when deciding which issues to consider in accomplishing such task.

We decided to adopt an experimentation methodology which presents a series of questions that should be addressed, the types of studies and actions that best address those questions and guidelines that should be taken into account in order to achieve specific measurements. The adopted methodology has been proposed by Travassos et al.

[2002]. It covers different experimental process areas such as setting, design, operation and data collection, results analysis, reporting and interpretation.

The methodology is composed by a set of experimentation phases, namely:

i. Definition: the experimentation is stated in terms of problems and purposes;
ii. Planning: hypotheses are formulated, instrumentation is described, the experimentation setting is established, and variables to be measured are determined;
iii. Execution: this phase puts in practice what has been established in planning; and
iv. Result Analysis and Interpretation: obtained data from experiments are organized, analyzed and packaged in order to be properly presented.

In the next sections, we go step-by-step following the guidelines proposed by the methodology. Our main goals when conducting this experimentation are twofold: (i) we want to verify whether the use of semantics really enhances a query reformulation process; and (ii) we want to guarantee that our *SemRef* algorithm produces sound and complete query reformulations.

7.2. Experimentation Purposes

The overall purpose of this experimentation is to evaluate whether semantics employment, by means of contextual information and the use of semantic correspondences, enhances a query reformulation process in a dynamic and distributed environment. To this end, we use two main evaluation criteria resulting from the determined properties of *SemRef*:

- *Soundness*: Given an original submitted query Q, each reformulated query Q' in the resulting reformulation set RS is a correct reformulation.
- *Completeness*: Given an original submitted query Q, the *SemRef* algorithm is able to find all the existing solutions (reformulations) for Q in the resulting set of reformulations RS.

This means that, by using semantics in query reformulation time, we aim to provide users with not only exact answers but also approximate, additional (i.e., expanded) answers, according to their preferences when formulating queries. To achieve such goal, our algorithm produces two kinds of query reformulations: exact and enriched. We want to show that these query reformulations are correct and all possible reformulations are provided, considering the context of the query, of the environment and of users.

According to such overall purpose, there are two measurement purposes to be accomplished in the experimentation:

- To characterize what happens when semantics is applied in a query reformulation process; and
- To evaluate if the use of semantics, through the set of correspondences and contextual information, really enhances the overall query reformulation process, by guaranteeing soundness and completeness.

By characterize, we mean describing what happens when considering and not considering semantics in a query reformulation process. In other words, we want to distinguish a query reformulation process that makes use of semantics from other one which does not. By evaluate, we mean assessing added value of using semantics in the referred process, i.e., what we gain by such usage. Furthermore, we want to guarantee that the *SemRef* algorithm is able to generate sound and complete reformulations.

Considering that, we define the components of our experimentation, through four parameters, as follows:

- Object of study: our semantic-based query reformulation approach, named *SemRef;*
- Purpose: characterize and evaluate;
- Focus: the use of semantics in the query reformulation process; and
- Point of view[24]: the researcher, in characterizing and evaluating the query reformulation process instantiation.

The questions we want to answer through the experimentation are the following:

- Question_1: What is the difference in producing query reformulations considering semantics and not considering semantics? To what extent does the use of semantics change the resulting set of query reformulations?

  Measure: Resulting Set of Query Reformulations, with/without semantics

- Question_2: Given an original query Q, is there a possibility to produce an empty set of query reformulations of Q? In which situations could the use of semantics help to avoid an empty set of query reformulations?

  Measure: Empty Resulting Set of Query Reformulations, with/without semantics.

- Question_3: Is it possible to produce correct query reformulations, either exact or enriched, with the aid of semantics?

[24] The person who benefits from the experimentation.

Measure$_1$: Exact query reformulations with/without semantics

Measure$_2$: Enriched query reformulations with/without semantics

Next section, we provide our experimentation design.

7.3. Experimentation Planning

In this section, we focus on design issues concerning the experimentation. We formulate hypotheses to be proved or refuted, provide how the experimentation will be executed and present the setting to be considered during experiments.

7.3.1. Hypotheses Definition

One of the goals of an experimentation process is to observe and measure in order to test, prove or refute formulated hypotheses. Hypotheses and variables (to be defined in Section 7.3.4) influence the choice of the experimental design. We have defined two hypotheses: a null hypothesis (H0) and an alternative hypothesis (H1). The null hypothesis represents a theory that has not been proved but can be used as a basis for our argument. The hypothesis contrary to the null hypothesis is the alternative hypothesis which is a statement of what we want to prove in order to achieve our purposes. In our experimentation, they are stated as follows:

i. Null Hypothesis (H0): A query reformulation process which is carried out without considering semantics is similar to a query reformulation process which takes into account obtained semantics. Both produce the same set of query reformulations.
ii. Alternative Hypothesis (H1): A query reformulation process which is carried out without considering semantics may produce empty query reformulations and thereby empty query results for a given query. Considering semantics, exact and/or enriched query reformulations may be produced, thus providing a larger set of query reformulation possibilities. Such query reformulations are correct according to the acquired contextual information and to the set of semantic correspondences.

The null hypothesis describes the general statement concerning query reformulation processes, i.e., there is no difference when applying or not applying semantics in these processes. This hypothesis is treated as valid unless the actual behavior of the current experiments contradicts this assumption. Thus, the null hypothesis relates to the statement being tested, whereas the alternative hypothesis relates to the statement to be accepted if/when the null hypothesis is rejected. In fact, the null hypothesis is the reverse of what we actually believe; it is put forward to allow the experimental data to contradict it.

7.3.2. Instrumentation Description

In our experiments, we are dealing with two peers – P_1 and P_2, which may be labeled source and target depending on where the original query is submitted. A set of queries expressed using concepts from the source peer's ontology will be assigned to it. Users can set their preferences through enriching variables (*approximate*, *generalize*, *specialize* and/or *compose*). These preferences relate to a set of submitted queries and help to guide the execution of the query reformulation algorithm. Whenever users want, they can redefine these variables. In addition, users can define the query reformulation mode through *expanded* or *restricted* options (the default is the latter).

In this sense, a variety of queries from peer P_1 to peer P_2 and from P_2 to P_1 are executed. For each query, the concepts are identified, and, if, for each concept, a semantic correspondence is found, then each concept is rewritten or expanded according to the corresponding concepts of the target peer ontology.

For each query, we characterize and evaluate its reformulation, considering semantics and not considering semantics, both in restricted and expanded modes.

7.3.3. Setting Overview

We use two scenarios in order to evaluate queries. Both are composed by the peers P_1 and P_2, with their respective ontologies O_1 and O_2. The first scenario is concerned with the "Education" knowledge domain. The other one is related to the "Travel" knowledge domain. In the former, peers have complementary data about academic people and their work (e.g., research) from different institutions. In the latter, peers share information about tourism, such as accommodation and destination. Ontologies from the Education scenario are depicted in Appendix B. Ontologies from the Travel one are shown in Appendix D. In both cases, it is very likely that a query may obtain a more complete answers resulting set according to the diverse data sources.

We have conducted our evaluation using queries expressed in $\mathcal{ALC}$/DL, although using SPARQL would have produced the same set of query reformulations[25]. In this sense, the selected queries have been chosen to represent a variety of $\mathcal{ALC}$/DL query formulation possibilities, namely: queries with one concept, queries with negation over a concept, queries with conjunctions, queries with disjunctions, queries with disjunctions of conjunctions. All selected queries follow the general query formula, provided in Definition 6. Furthermore, for each one of the presented queries, we check the combination of reformulation possibilities according to the set of enriching variables specification and reformulation mode, as previously shown in Table 5.1. In the following, we present a relevant fragment of the set of queries we have used for each

[25] An excerpt of the used queries expressed in SPARQL is also shown in Appendix E.

scenario, considering the variety of possible constructions we have defined. The complete set of used queries is shown in Appendix E.

a. Queries with One Concept

Q_1: Student

Q_{47}: Safari

b. Queries with Negation over a Concept

Q_9: ¬PhDStudent

Q_{52}: ¬FamilyDestination

c. Queries with Conjunctions

Q_{22}: Professor ⊓ PostDoc

Q_{40}: Activity ⊓ Sightseeing

d. Queries with Disjuntions

Q_6: Proceedings ⊔ Thesis ⊔ ¬TechnicalReport

Q_{49}: Campgroung ⊔ ¬Hotel

e. Queries with Disjuntions of Conjunctions

Q_{11}: [AdministrativeStaff ⊓ ClericalStaff] ⊔ [Faculty ⊓ Lecturer]

Q_{33}: [Student ⊓ Monitor] ⊔ [Worker ⊓ Chair] ⊔ [Assistant ⊓ ¬TeachingAssistant] ⊔ [Faculty ⊓ AssociateProfessor]

Q_{44}: [Destination ⊓ Capital] ⊔ [Destination ⊓ ¬Farmland] ⊔ ¬NationalPark

Q_{53}: [Destination ⊓ RetireeDestination] ⊔ [UrbanArea ⊓ City]

7.3.4. Variables

Variables provide the means to organize our observations and obtained experimental data. The idea is trying to define variables avoiding redundancy. According to our experimentation purposes and evaluation criteria, we have defined some variables to be measured. They are stated as follows:

i. V1: #Empty Reformulations or #EmptyRef
ii. V2: #Exact Reformulations or #ExactRef
iii. V3: #Enriched Reformulations or #EnrichedRef
iv. V4: Degree of Soundness or DS, defined as:

$$\mathrm{DS}(\mathrm{Correct}, \mathrm{PossibleRef}) = \frac{\#\,\mathrm{Correct}}{\#\,\mathrm{PossibleRef}}$$

Where, given an original query Q, #Correct is the number of correct reformulations (Exact or Enriched) of Q and #PossibleRef is the total of all possible reformulations of Q.

v. V5: Degree of Completeness or DC, stated as:

$$DC(Correct, ProducedRef) = \frac{\#Correct}{\#ProducedRef}$$

Where, given an original query Q, #Correct is the number of correct reformulations (Exact or Enriched) of Q and #ProducedRef is the total of all produced reformulations of Q.

It is important to note that both DS and DC are adapted from standard metrics (*precision* and *recall*, respectively) commonly used in information retrieval systems [Baeza-Yates and Ribeiro-Neto 1999]. These metrics are usually used to measure the results of a query execution and to compare them to ideal results expected by the user. In our case, we aim to evaluate the query reformulation phase. Consequently, we are concerned with the quality (correctness) of query reformulation instead of with the query execution results (which currently are outside of our scope).

7.4. Operation

In order to execute the experiments, we have defined three types of evaluation tasks:

i. *Query Reformulation without semantics*: this is the basic kind of query reformulation process. It means that the set of queries will be submitted, and *SemRef* will try to reformulate each one, without considering any kind of semantics, i.e., enriching variables will be disabled and query reformulation mode will operate on its default – *restricted*. In other words, only equivalence correspondences will be verified when reformulating the original query in terms of the target one. As a result, only *exact* reformulations will be present in the resulting reformulation set.

ii. *Query Reformulation with semantics, in restricted mode*: in this case, users have set at least one enriching variable, allowing the algorithm to verify the possibility to produce enriched reformulations in case of empty exact reformulations have been generated. To this end, semantic correspondences, besides equivalence, will be verified in order to produce enriched reformulations (in place of empty exact ones). As a consequence, *either* exact *or* enriched reformulations will be present in the resulting set of query reformulations.

iii. *Query Reformulation with semantics, in expanded mode*: expanded mode means that the algorithm will always try to produce both exact *and/or* enriched reformulations. To this end, at least one of the enriching variables must have been set. In this option, the resulting set of reformulations will be the largest one, since both exact and/or enriched reformulations will be produced, according to the enriching variables setting.

In order to provide a greater variety of possibilities regarding the reformulation mode and kinds of enrichment that could be employed, during the experiments, we have defined enriching variables considering different combinations of choices.

In the following, we depict part of the experiments execution. The overall set of experiments which has been conducted is shown in Appendix E. The first set of experiments concerns the reformulation of queries without any kind of semantics. An excerpt of its results is presented in Table 7.1.

Table 7.1 Query Reformulation without Semantics – Mode: Restricted

Query	Q_{exact}
Q_1	[[Student]]
Q_4	∅
Q_5	FullProfessor
Q_9	∅
Q_{12}	∅
Q_{27}	∅
Q_{43}	[[Destination] ⊓ [Capital]]
Q_{46}	∅

As we can see, some of the produced query reformulations were empty (∅). This means that, regarding these queries, no equivalence correspondence was found to accomplish the reformulation. Next, we perform the experiments by considering the definition of enriching variables, i.e., with semantics. In the same way, we maintain restricted as our reformulation mode.

Table 7.2 Query Reformulation with Semantics – Mode: Restricted with Enriching Variables

Query	Spec	Gen	Approx	Comp	Q_{exact}	$Q_{enriched}$
Q_1	X		X		[[Student]]	∅
Q_2	X		X		[Student ⊓ UndergraduateStudent]	∅
Q_4	X		X		∅	[[VisitingProfessor]]
Q_{19}	X	X	X		[[¬Book]] ⊔ [[¬Article]]	∅
Q_{20}	X	X	X		∅	∅
Q_{25}		X	X		∅	[[¬PhDStudent ⊔ ¬GraduateStudent]]
Q_{54}	X	X	X	X	[[City]] ⊔ [[¬Beach]]	∅

As we can see, some of the queries whose produced reformulations resulted empty in the first set of experiments (without semantics) are now able to be enriched. As a result, most of submitted queries could be properly reformulated. Next, we execute the experiments by considering the definition of enriching variables, i.e., with semantics, in expanded mode.

Table 7.3 Query Reformulation with Semantics – Mode: Expanded and Enriching Variables

Query	Spec	Gen	Approx	Comp	Q_{exact}	$Q_{enriched}$
Q_3	X		X		[[Worker]]	[[Assistant ⊔ Faculty ⊔ AdministrativeStaff]]
Q_4	X		X		∅	[[VisitingProfessor]]
Q_{13}	X			X	∅	[[Course ⊔ ResearchProject] ⊓ [PostDoc ⊔ Professor]]
Q_{15}	X	X	X	X	Publication	[[Work ⊔ ResearchProject]] ⊔ [[Specification ⊔ Software ⊔ Article ⊔ Manual ⊔ Book ⊔ UnofficialPublication]]
Q_{37}	X	X			[[Sightseeing]]	[[Activity]] ⊔ [[Activity ⊔ Safari]]
Q_{38}	X	X			[[¬RetireeDestination]]	[[¬Destination]]
Q_{39}			X	X	[[BedAndBreakfast]]	∅
Q_{46}		X	X		∅	∅
Q_{48}	X	X			[[Sightseeing]]	[[Activity ⊔ Museums]]

In this case, most of submitted queries were properly reformulated in terms of exact and enriched reformulations. Still, it is possible that no reformulation can happen. This is due to the fact that sometimes, the enriching variable that has been set does not match any semantic correspondence. For instance, the user can have set *approximate* variable, but there is no *closeness* correspondence for the concepts in the submitted query. An example of such occurrence is provided in the reformulation of query Q_{46}.

We organized these experimental data in terms of the variables we have defined to measure. Table 7.4 shows the number of *exact*, *enriched* and *empty* reformulations over the total of possible reformulations for each evaluation process.

For these experiments, we had a total of 55 submitted queries. Thereby, in *restricted* mode, the total of possible produced query reformulations was 55, since in the option *without semantics* it is only possible to produce exact or empty reformulations, i.e., we cannot produce enriched ones. This results in 55 possible reformulations in all. In the option *restricted with semantics*, the algorithm also produces only exact or empty reformulations. The difference is that, in this latter option, when an empty reformulation is produced, it can be replaced by an enriched one, if enriching variables have been set. As a result, the number of possible reformulations for the 55 submitted queries is also 55 (computing exact or empty or enriched reformulations).

Table 7.4 Number of Produced Exact, Enriched and Empty Reformulations

Evaluation Process	#ExactRef (%)	#EnrichedRef (%)	#EmptyRef (%)	Total of Produced Query Reformulations (%)
Query Reformulation without Semantics	53	---	47	53
Query Reformulation with Semantics – *Restricted Mode*	53	38	9	91
Query Reformulation with Semantics – *Expanded Mode*	26	43	31	69

When considering *expanded mode*, the scope changes a little bit since the algorithm now tries to produce both exact and/or enriched reformulations. Thus, we can have a total of reformulations varying from 55 (only one of exact or enriched is produced) to 110 (both exact and enriched are produced). Thereby, we have considered as the total of possible reformulations for this set the number of 110, although we know we are considering the worst case.

On the other hand, if we analyze the viability of producing reformulations (exact or enriched, or both together as one), we change the last line of Table 7.4 by Table 7.5, as follows.

Table 7.5 Produced Reformulations, considering Exact and Enriched as One

Evaluation Process	#EmptyRef (%)	Total of Produced Query Reformulations (%)
Query Reformulation with Semantics – *Expanded Mode*	9	91

This result is based on the fact that, given a query Q, if *SemRef* is able to produce at least one of the expected reformulations, it succeeds, although it can produce both of them as well. Explaining better, considering the submitted query Q_3 = Worker, an exact reformulation Qexact = [[Worker]] and an enriched reformulation Qenriched = [[Assistant ⊔ Faculty ⊔ AdministrativeStaff]] were produced. This means that the algorithm did not produce an empty reformulation, instead it produced both kinds of possible reformulations. The main point in this case is that when computing empty and produced reformulations for a given query, we can state that we had a succeeded reformulation, if it was not empty. Even in the example of Q_4 = AssistantProfessor, where $Q_{exact} = \varnothing$ and $Q_{enriched}$ = [[VisitingProfessor]], the algorithm also succeeded, since at end there was a reformulation of Q_4. Therefore, in Table 7.5, we have an overview of how many reformulations resulted empty or not, and, at the same time, of how many reformulations succeeded.

We have measured the degree of soundness and the degree of completeness for the set of submitted queries as shown in Table 7.6. The results presented in this table consolidate the numbers we have found in the two previous tables of results. Regarding soundness, we verify that its degree increases when we apply semantics, i.e., the *SemRef* algorithm is able to produce a higher number of correct (i.e., exact and/or enriched) query reformulations. Regarding completeness, we also verify this truth. Moreover, we verify that, considering semantics, the *SemRef* algorithm is able to provide the complete set of query reformulations, i.e., it is able to find all the existing solutions (reformulations) for a given query Q, of course, taking into account the contextual information and existing semantic correspondences.

Table 7.6 Degree of Soundness and Completeness

Evaluation Process	Degree of Soundness (%)	Degree of Completeness (%)
Query Reformulation without Semantics	53	58
Query Reformulation with Semantics – *Restricted Mode*	91	100
Query Reformulation with Semantics – *Expanded Mode*	69	100

Next section, we detail the analysis we have done concerning these results.

7.5. Results Analysis

The number of produced exact, enriched and empty query reformulations over the total of possible reformulations is reported in Figure 7.1. In the first process, only exact reformulations were produced and there was a high number of empty reformulations. When reformulations were empty, this meant that no reformulation at all was produced and, moreover, no answers from the target peer were returned. In these cases, returned answers were only originated from the source peer. In the second run, the same number of exact reformulations was generated, but when exact reformulations were empty, an enriched one was provided, considering values defined in the enriching variables. Thereby, exact or enriched reformulations were produced.

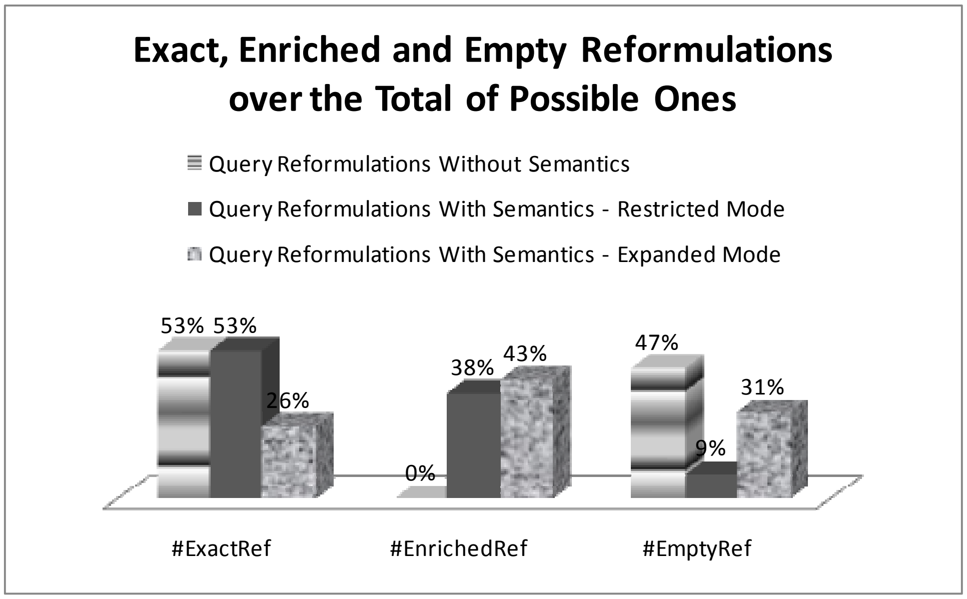


Figure 7.1 Exact, Enriched and Empty Reformulations over Possible Ones

In the third option (queries with semantics, in expanded mode), we could have as resulting query reformulations both exact and/or enriched, which entailed a larger set of reformulations possibilities. As a result, we had both exact and enriched reformulations calculated in separate (see Figure 7.1). The number of empty reformulations was taken individually from exact or enriched reformulations set. We argue that such number can not be considered meaningful, since, in expanded mode, when one of exact or enriched reformulations is empty, the other one can be produced instead, e.g., if Q_{exact} is empty, $Q_{enriched}$ may be produced or vice-versa. To clarify this situation, we present another graph, depicted in Figure 7.2, which shows the total number of produced query reformulations, when considering Q_{exact} and $Q_{enriched}$ as one reformulation solution. In other words, in this case, we only denote as an empty reformulation the one whose both Q_{exact} and $Q_{enriched}$ were empty.

More precisely, in Figure 7.3 we report the degrees of soundness and completeness of our *SemRef* algorithm. As we can see, these results materialize what we have stated in Section 5.4, when we proved *SemRef* main properties in terms of soundness and completeness. Therefore, we show that when we do not consider semantics usage, we have a high number of empty reformulations, which results in a lower number of correct query reformulations. When we consider semantics by the use of semantic correspondences and contextual information, we are able to obtain enriched reformulations substituting empty exact ones (in restricted mode) or adding another reformulation (in expanded mode) as a way of query enrichment.

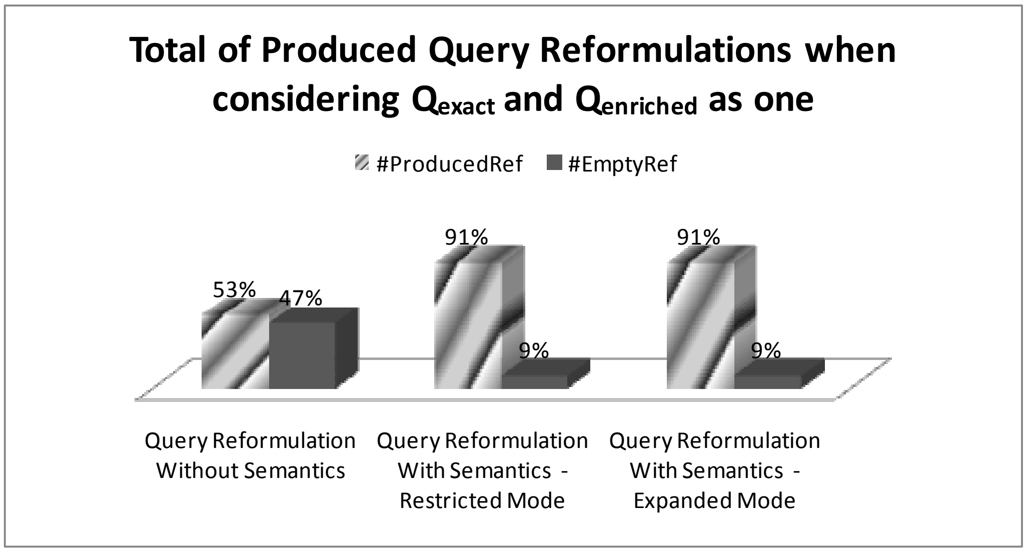


Figure 7.2 Total of Query Reformulations, when considering Q_{exact} and $Q_{enriched}$ as one

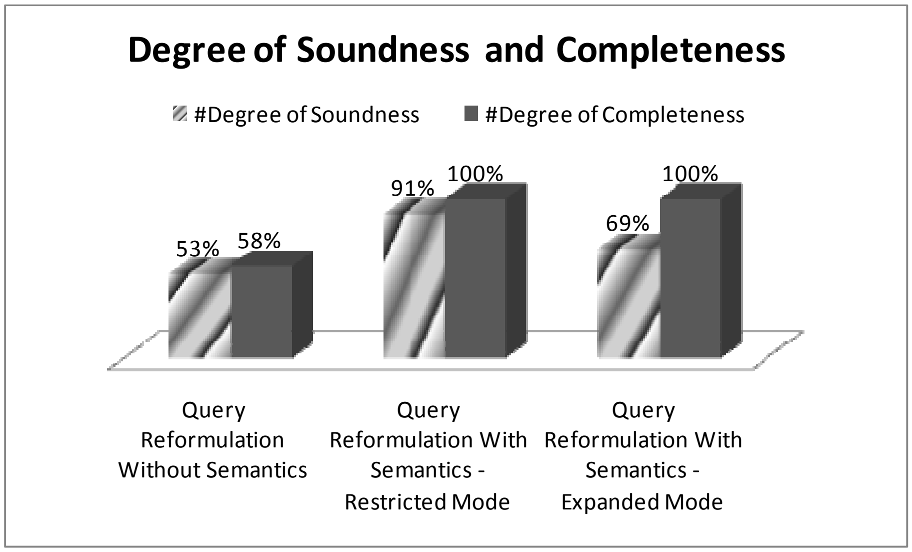


Figure 7.3 Degree of Soundness and Completeness

Explaining better, we observed that concepts that only exist in one of the peer ontologies usually do not have an equivalent concept in the target one, thus entailing an empty exact reformulation. In these cases, enriching the reformulation has been essential, otherwise, no reformulation query would be obtained. Moreover, even enabling only one of the enriching variables has shown a promising query reformulation result. When our approach takes into account the preference of the user and exploits the correspondences built from them, we are able to obtain new queries including additional concepts and, consequently, additional expanded answers. Besides, when users set at least one of the enriching variables, they are also defining that the negation over concepts must be dealt with, not only with the usual correspondences, but, particularly, with disjointness.

Although the use of semantics is highly context-dependent, considering our particular experimental setting (the two scenarios with their respective peer ontologies and the set of submitted queries), we are able to refute null hypothesis (H0), since we have shown that a query reformulation process which is carried out without considering semantics is able to produce only a subset of the query reformulation set provided by considering semantics. This is mainly due to the fact that these traditional approaches only consider equivalence correspondences in order to perform query reformulation, what implies in a high number of impossible reformulations. On the other hand, in our approach, we go further when we take into account other semantic correspondences and context as well, providing a larger set of possibilities of query reformulations.

Furthermore, we can conclude that our alternative hypothesis (H1) is true, i.e., a query reformulation process which is carried out without considering semantics may produce empty query reformulations and thereby empty query results for a given query. Considering semantics, exact and/or enriched query reformulations may be produced, thus providing a larger set of query reformulation possibilities. Such produced query reformulations are correct. Moreover, they are considered sound and complete, according to the acquired contextual information and to the set of semantic correspondences between the current peers.

7.6. Concluding Remarks

The most important conclusion of the experiments is that they have demonstrated a proof-of-concept of the *SemRef* approach. However, some considerations need to be made. We have run experiments considering and not considering semantics, and we have obtained interesting results regarding that usage.

In those experiments, we have used all the flexibility of the *SemRef* approach to consider different settings of enriching variables and query reformulation mode, existing semantic correspondences and alternatives to produce an enriched reformulation in case of an empty one. Our main goal was to characterize such instantiation, learn with it and evaluate whether it really provides an enhancement to query reformulation phase. As we have proved, the null hypothesis was refuted and the alternative one can be accepted, considering the obtained experimental results.

In this sense, the experimental results have supported the hypothesis that considering semantics through the set of correspondences and acquired context enhances the query reformulation process, by providing exact and/or enriched reformulations. In this version, we have considered the context of the user/query, through the set of defined preferences, verifying the possibilities that may arrive when expanded or restricted option is enabled. For performance reasons, although we produce one or two reformulations of a given query, we put both reformulations together in one execution query. Thus, a peer executes one query and returns its answers to the submission peer which integrates all the results and presents the final one.

Another important remark is that, during experiments, we have identified some conflicts with respect to the produced reformulations. These conflicts arise due to the fact that when we substitute concepts, at query reformulation time, one concept may be substituted by a set of other concepts by using different semantic correspondences. If there is a negation, sometimes (whether there are disjointness correspondences), the negated concept may be replaced by its disjoint (not negated) concepts. As a result, these produced sets of replaced concepts may be in conflict (e.g., a reformulated query may result in [¬UrbanArea ⊔ UrbanArea]) or they may be redundant (e.g., a reformulated query results in [[RuralArea ⊔ UrbanArea] ⊔ RuralArea]). We are aware of such problems and intend to deal with them in future work using some optimization strategy.

Next chapter presents conclusions regarding the overall work and points out some further work that can be accomplished.

CHAPTER 8

"More than words
is all you have to do to make it real".
Gary Cherone e Nuno Bettencourt

Conclusions

Query answering among peers in networked environments is a challenge which has been addressed in different settings, including Data Integration and Peer Data Management Systems (PDMS). In such environments, a query posed at a peer is routed to other peers so that an answer can be found. An important step in this process is reformulating a query issued at a peer into a new query expressed in terms of a target peer, considering the correspondences between them. Most traditional approaches aim at reformulating a given query into another one by using equivalence correspondences. However, concepts from a source peer do not always have exact corresponding concepts in a target one, what results in an empty set of reformulations and, possibly, no answer to users. In this case, if users define that it is relevant for them to receive semantically related answers, it may be better to produce an adapted query reformulation and, consequently, close answers than no answer at all.

Due to the fact that these computational environments are highly dynamic, the use of semantics (including context) surrounding processes such as query reformulation may be rather important to produce results in conformance with users' needs and environment's capabilities. Besides, a considerable effort has been employed in recent years to provide reformulation techniques which enrich user queries before their execution. We argue that by bringing together these two worlds, i.e., semantics and query enrichment techniques, we can enhance the query reformulation process. Moreover, using semantics as a way of enriching the query reformulation process may enhance the overall query answering process.

This work was motivated by these issues and had the objective to present a query reformulation approach – *SemRef*, which brings together the concepts of *query reformulation* and *query enrichment* in dynamic distributed environments. In our approach, exact and enriched query reformulations are produced as a means to provide users with a set of expanded answers. To this end, it makes use of semantics which is mainly acquired from a set of semantic correspondences that extend the ones commonly found. Examples of such unusual correspondences are closeness and disjointness. Furthermore, *SemRef* takes into account the context of the user, of the query and of the environment as a way to enhance the overall process and to deal with information that can only be acquired on the fly.

This chapter is organized as follows: Section 8.1 discusses the contributions achieved with this research; Section 8.2 indicates some directions in which this research could be extended; and Section 8.3 concludes the thesis with some remarks.

8.1. Thesis Contributions

The contributions of this work are both theoretical and practical. We split them into four specific ones:

I. The specification and implementation of an approach to identify semantic correspondences

We used domain ontologies as background knowledge in order to identify semantic correspondences between matching ontologies. The motivation underlying that was the observation that concepts from two matching ontologies are rarely precisely equivalent, but rather have some semantic overlap. Thereby, finding such degree of semantic overlap became more useful for our task of query reformulation. In this sense, we tried to overcome the limitations of traditional approaches and we went one step further since, besides the common correspondence of equivalence, we also identified other semantic ones, namely, specialization, closeness, generalization, part-of, whole-of and disjointness, providing various and semantically-rich degrees of similarity between ontology elements. To the best of our knowledge, closeness is a type of semantic correspondence that is not found in any related work.

II. The specification and implementation of a context ontology

We designed and developed CODI - **C**ontextual **O**ntology for **D**ata **I**ntegration which is an ontology for representing context according to some Data Integration (DI) and PDMS issues [Souza et al. 2008]. This work used the concept of context as a way to enhance the query reformulation process. More specifically, *SemRef* used three types of context: of the users, represented by the set of preferences that they define; of the query, acquired from the identification of its semantics and its query reformulation mode; and of the environment, regarding the set of relevant peers to where queries would be reformulated.

III. The specification and implementation of the *SemRef* approach

The *SemRef* approach is the main contribution of this work [Souza et al. 2009].

We formalized the *SemRef* approach using $\mathcal{ALC}$/DL. Also, we proved that the *SemRef* algorithm is sound, complete and terminates. We compared our approach with other existing ones and verified that it has various advantages in relation to the others:

- *SemRef* brings together the concepts of query reformulation and query enrichment within a dynamic distributed ontology-based environment;
- It focuses on reformulating a query in terms of one or two kinds of reformulation, by means of exact (the best) and/or enriched reformulations;
- It makes use of a set of semantic correspondences, namely, equivalence, specialization, generalization, closeness, part-of, whole-of and disjointness, providing different levels of concept approximation;
- It uses closeness as a way to provide expanding concepts related in a given context. Also, it uses disjointness to deal with negations, i.e., it directly obtains the disjoint concept as a solution to the negation over an original concept;
- It performs query reformulation considering the context of the users, of the query and of the environment; and
- It is able to provide users with a set of expanded answers as a result of the execution of exact and/or enriched reformulations;

IV. The specification and implementation of the Translation between $\mathcal{ALC}$/DL and SPARQL queries

We used $\mathcal{ALC}$/DL and SPARQL in *SemRef* query module. The reasons underlying these choices were: (i) it was important to validate our query reformulation approach using $\mathcal{ALC}$/DL, since it had been formally coded as such; (ii) we executed queries over ontologies using SPARQL.

Our rationale for translating $\mathcal{ALC}$/DL queries into SPARQL ones was to provide users with the possibility of querying using SPARQL in the same way they would do in DL. Thus, considering a subset of SPARQL syntax, we defined some templates which may be used by users to write their SPARQL queries, using the constructors we find in $\mathcal{ALC}$/DL, namely, conjunction, disjunction and negation.

8.2. Future Work

Every work has its limitations and deserves extensions and/or improvements. Therefore, we present some verified limitations and/or issues which may be extended, pointing out many possible directions for future research on semantic-based query reformulation. They are briefly described as follows:

- Optimizing Query Reformulation

  During experiments, we identified the possibility of conflicts arising from the substitution of concepts by their semantically related ones. This happened due to the fact that when we substitute concepts at query reformulation time, one concept may be substituted by a set of other concepts by using different semantic correspondences. If there was a negation, the negated concept might be replaced by its disjoint (not negated) concepts. As a result, these produced sets of replaced concepts might be in conflict or even redundant. We intend to deal with these problems in future work using some optimization strategy, such as the one provided (as a kind or query enrichment) in Section 5.2.

- Reasoning over Context

  Currently, CODI stores contextual information provided by users' preferences. We will extend such usage by considering also the context of the environment and of the semantic correspondences. We will develop rules to allow reasoning over the context already instantiated and work with the ones acquired on the fly. This reasoning might improve the query reformulation and routing process.

- Implementing Property Correspondences.

  Currently, we have restricted the implementation of correspondences between ontologies' properties to equivalence, specialization and generalization. There is still work to be done with respect to closeness, disjointness and aggregation. We intend to provide a solution to this problem by using the domain and range of the concepts to which the properties are linked.

- Extending the *SemRef* Approach to a Set of Diverse Peers.

  This entails working with query routing policies. To this end, semantics should also be used as a way to enhance the selection of relevant semantic neighbors and their ranking. Query routing should also ensure preserving the query semantics at the best possible level of approximation.

- Providing users with a more friendly query interface

  The query module is intended to provide users with a high-level interface, in such a way that both novel and experienced users can formulate their queries.

Thus, query formulation by using the concepts visually provided in the ontology may be a partial solution to this problem. Other kinds of query formulation interfaces will be explored.

- Performing Experiments with Real Users

As soon as the interface becomes more friendly and high-level, it will be possible to perform experiments with real users. By accomplishing this task, we can evaluate if the set of expanded answers really matches the users' preferences.

8.3. Concluding Remarks

This work investigated the use of semantics in query reformulation processes in dynamic distributed environments. The *SemRef* approach, with its formalized definitions, implemented algorithms and performed experiments, was presented as a solution to this key issue. We used semantics acquired from a set of extended semantic correspondences and from the context of the user and of the query. We showed that by using such semantics, the approach developed in this thesis is able to produce exact and enriched reformulations and, consequently, a set of expanded answers to users. We proved that the algorithm underlying our approach is sound and complete, and also tested these properties in our conducted experiments.

REFERENCES

Adjiman P., Goasdoué F., Rousset M.-C. (2007): SomeRDFS in the Semantic Web. In Journal on Data Semantics, LNCS, 2007, vol. 8, p. 158-181.

Aleksovski Z., Klein M., Katen W., Harmelen F. (2006): Matching Unstructured Vocabularies using a Background Ontology. In: S. Staab and V. Svatek, editors, Proc. of EKAW, LNAI. Springer-Verlag, 2006.

Arenas, M., Kantere, V., Kementsietsidis, A., Kiringa, I., Miller, R. J., e Mylopoulos, J. (2003): The Hyperion Project: From Data Integration to Data Coordination. ACM SIGMOD Record 32, 3, 2003.

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

Baader F., Horrocks I., and Sattler U. (2007): Description Logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, Handbook of Knowledge Representation. Elsevier, 2007.

Baeza-Yates R., Ribeiro-Neto B. (1999): Modern Information Retrieval. ACM Press/Addison-Wesley, 1999.

Bai J., Nie J.,Bouchard H. and Cao G. (2007): Using Query Contexts in Information Retrieval. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07), July 2007, Amsterdam.

Basili V. (2007): The Role of Controlled Experiments in Software Engineering Research. In Empirical Software Engineering Issues, LNCS 4336, V. Basili et al., (Eds.), Springer-Verlag, pp. 33-37, 2007.

Bazire, M., Brézillon P. (2005): Understanding Context Before Using It. 5th International and disciplinary Conference, CONTEXT 2005, Paris, France, July 5-8, 2005.

Belian, R. B. (2008): A Context-based Name Resolution Approach for Semantic Schema Integration, PhD thesis, Center for Informatics, UFPE, 2008.

Bellotti, V., Edwards, K. (2001): Intelligibility and Accountability: Human Considerations in Context-Aware Systems. In: Human Computer Interaction, v. 16, n. 2, 3 & 4, pp. 193-212.

Benerecetti M., Bouquet P., and Ghidini C. (2001): On the dimensions of context dependence: partiality, approximation, and perspective. In Proc. 3rd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT), volume

2116 of Lecture notes in computer science, pages 59–72, Dundee (UK), 2001.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001): The Semantic Web. Scientific American 284(5), pag: 34-43.

Bilke A. (2007): Duplicate-based Schema Matching. PhD Thesis. Berlin University.

Bolchini C., Curino C., Orsi G., Quintarelli E., Rossato R., Schreiber F., and Tanca L. (2008). And what can context do for data? Communications of the ACM.

Bolchini C., Curino C.A., Quintarelli E., Tanca L., Schreiber F. (2007):. A data-oriented survey of context models. SIGMOD Record, 2007.

Borgida A. and Serafini L. (2003): Distributed description logics: Assimilating information from peer sources. Journal of Data Semantics, 1:153–184. LNCS 2800, Springer Verlag, 2003.

Borst W. (1997): Construction of Engineering Ontologies for Knowledge Sharing and Reuse: Ph.D. Dissertation, University of Twente.

Brézillon P. (2003): Context Dynamic and Explanation in Contextual Graphs. Proceedings of the 4th International and Interdisciplinary Conference, CONTEXT 2003, USA, pp: 94-106.

Cai, G., Wang, H., MacEachren, A. (2003): Communicating Vague Spatial Concepts in Human-GIS Interactions: A Collaborative Dialogue Approach. In: Conference on Spatial Information Theory, LNCS2825, pp. 304-319, Kartause Ittingen, Switzerland.

Calvanese D., Giacomo G., Lembo D., Lenzerini M. and Rosati R. (2004): What to ask to a peer: Ontology-based query reformulation. In Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004), 2004.

Cao Y. (2007): Processing SparQL Queries in an Object Oriented Mediator. Uppsala Master's Theses in Computing Science, ISSN 1100-1836 (2007).

Chalmers M. (2004): A historical view of context. Computer Supported Cooperative Work, 13(3):223–247, 2004.

Chamiel, G. and Pagnucco, M. (2008). Utilizing Ontological Structure for Reasoning with Preferences. In Proc. Knowledge Representation Ontology Workshop (KROW 2008), Sydney, Australia. CRPIT, 90. Meyer, T. and Orgun, M. A., Eds. ACS. 1-9.

Chaudhuri S. and Dayal U. (1997): An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26(1):65{74, 1997.

Dey A. (2001): Understanding and Using Context. Personal and Ubiquitous Computing Journal, Volume 5, pp. 4-7, 2001.

Duschka O. M. and Genesereth M.R. (1997): Answering recursive queries using views. In Proceedings of ACM Symposium on Principles of Database Systems, pages 109–116, 1997.

Egenhofer M. (1991): Reasoning about Binary Topological Relations. In Oliver Günther, Hans-Jörg Schek (Eds.): Advances in Spatial Databases, Second International Symposium, SSD'91, Zürich, Switzerland Proceedings. Lecture Notes in Computer Science 525 Springer, ISBN 3-540-54414-3, 1991.

Eisenstein, J., Puerta A. (2000): Adaptation in Automated User-Interface Design. In Proceedings of the International Conference on Intelligent User Interfaces, LA, USA, 2000.

Euzenat J. and Shvaiko P. (2007): Ontology matching. Springer, Heidelberg (DE), 2007.

Faye, D., Nachouki, G., and Valduriez, P. (2007): Semantic Query Routing in SenPeer, a P2P Data Management System. In Proc. of the 18th Int. Conf. on Database and Expert Systems Applications, Regensburg, Germany. pp. 365-374.

Fernández-López, M., Gómez-Pérez, A. (2002): The integration of OntoClean in WebODE. In Proceedings of the EON2002 Workshop at 13th EKAW, Siguenza, Spain.

Franklin M., Halevy A. and Maier D. (2005): From databases to dataspaces: a new abstraction for information management. SIGMOD Record, 34(4):27–33, 2005.

Ghidini C., Serafini L. (2006): Reconciling Concepts and Relations in Heterogeneous Ontologies. ESWC 2006: 50-64.

Giunchiglia F. and Zaihrayeu I. (2002): Making peer databases interact - a vision for an architecture supporting data coordination. In Proceedings of the 6th International Workshop on Cooperative Information Agents (CIA), pages 18–35, Madrid (ES), 2002.

Giunchiglia, F., Shvaiko, P., Yatskevich, M. (2004): S-match: an algorithm and an implementation of semantic matching. In: European Semantic Web Symposium (ESWC). pp. 61-75, 2004.

Glimm B. (2007): Querying Description Logic Knowledge Bases. PhD thesis, The University of Manchester, 2007.

Goh, C. (1997): Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems. Ph.D. Thesis, MIT Sloan School of Management (1997).

Grootjen F. A., van der Weide T. P. (2006): Conceptual query expansion, Data & Knowledge Engineering, v.56 n.2, p.174-193, February 2006.

Gruber, T. R. (1993): Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Knowledge Systems Laboratory, Stanford University, 1993.

Guarino N., Welty C. (2002): Evaluating ontological decisions with ontoclean.

Communications of ACM, Volume 45, Number 2, pages 61.65, 2002.

Guarino N., Welty, C. (2004): An overview of OntoClean. In Handbook on Ontologies in Information Systems (pp. 151- 172). Berlin: Springer.

Haase P., Wang Y. (2007): A decentralized infrastructure for query answering over distributed ontologies. SAC 2007: 1351-1356.

Halevy A. (2001): Answering queries using views: A survey. VLDB Journal, 10(4):270{294, 2001.

Halevy A., and Pottinger R. (2001): MiniCon: A scalable algorithm for answering queries using views, Very Large Data Bases Journal, Vol. 10, Num. 2-3, p. 182-198, 2001.

Halevy A., Ives Z., Suciu D., and Tatarinov I. (2005). Schema mediation for large-scale semantic data sharing. VLDB J., 14(1):68--83, 2005.

Halevy A., Rajaraman A. and Ordille J. (2006): Data integration: the teenage years. In. Proceedings of the 32nd international conference on Very large data bases - Volume 32, Pages: 9 – 16, 2006.

Herschel, S. and Heese, R. (2005): Humboldt Discoverer: A Semantic P2P index for PDMS. In Proc. of the International Workshop Data Integration and the Semantic Web, Porto, Portugal (2005).

Homola M. (2007): Towards Distributed Ontologies with Description Logics. Proceedings of the Knowledge Web PhD Symposium - KWEPSY 2007.

Horrocks I. (2005): Applications of description logics: State of the art and research challenges. Proc. of the 13th Int. Conf. on Conceptual Structures (ICCS'05) April 2005.

Horrocks I. and Tessaris S. (2000): A conjunctive query language for description logic aboxes. In AAAI/IAAI, pages 399–404, 2000.

Ives Z., Khandelwal N., Kapur A., Cakir M. (2005): ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data. Conference on Innovative Database systems Research (CIDR), Asilomar, CA, 2005.

Kashyap, V. and Sheth, A. (1996): Semantic and schematic similarities between database objects: a context-based approach. The VLDB Journal, v. 5. Springer-Verlag, pp. 276-304, 1996.

Kostadinov D. (2007): Data Personalization: an approach for profile management and query reformulation. PHD Thesis. Universite de Versailles Saint-Quentin-en-Yvelines, 2007.

Koutrika G., Ioannidis Y. (2005): Personalized Queries under a Generalized Preference Model. In Proc. of 21st Intl. Conf. On Data Engineering (ICDE), 841-852, 5-8 April 2005, Tokyo, Japan.

Kramer, R., Modsching, M., Schulze, J., Hagen, K. (2005): Context-Aware Adaptation in a Mobile Tour Guide. In: Proc. of the 5th International and Interdisciplinary Conference, CONTEXT 2005, LNCS3554, Paris, France.

Lenzerine M., Milano D., Poggi A. (2009): Ontology Representation & Reasoning. Technical Report, available at http://www.dsi.uniroma1.it/~estrinfo/1%20Ontology%20representation.pdf

Lenzerini M. (2002): Data integration: a theoretical perspective. In Proceedings of ACM Symposium on Principles of Database Systems, pages 233–246, New York, NY, USA, 2002. ACM Press.

Levy A. (1999): Combining Artificial Intelligence and Databases for Data Integration. Artificial Intelligence Today, 1999, pp. 249-268.

Litwin W., Mark L., and Nick Roussopoulos N. (1990): Interoperability of multiple autonomous databases. ACM Computing Surveys, 22(3):267{293, September 1990.

Lóscio, B. (2003): Managing the Evolution of XML-based Mediation Queries". PHD Thesis, Federal University of Pernambuco, Brazil.

Lv Q., Cao P., Cohen E., Li K., and Shenker S. (2002): Search and Replication in Unstructured Peer-to-Peer Networks. In Proc. of the 16th ACM International Conference on Supercomputing (ICS'02), New York, USA, 2002.

Madhavan J. and Halevy A. (2003): Composing mappings among data sources. In Proceedings of the International Conference on Very Large Databases (VLDB), pages 572{583, 2003.

Mandreoli F., Martoglia R., Penzoy W., and Sassatelli S. (2006): Semantic Query Routing Experiences in a PDMS. Proceedings of the 3rd Italian Semantic Web Workshop. SWAP (Semantic Web Applications and Perspectives), Pisa, Italy, 18-20 December, 2006.

Manning C. D., Raghavan P. and Schütze H. (2008): Introduction to Information Retrieval, Cambridge University Press. 2008.

McBrien P.J. and Poulovassilis A. (2006): P2P query reformulation over Both-as-View data transformation rules. In Proceedings of DBISP2P 2006.

Mills J., Goossenaerts J.B.M. (2005): Using contexts in managing product knowledge. In: E. Arai, J. Goossenaerts, F. Kimura, K. Shirase (eds) Knowledge and Skill Chains in Engineering and Manufacturing: Information Infrastructure in the Era of Global Communications, Springer, pp. 57-65, 2005.

Montanelli S., Castano S. (2008): Semantically routing queries in peer-based systems: the H-Link approach. Knowledge Eng. Review 23(1): 51-72 (2008).

Naumann F. (2001): From databases to information systems - information quality makes the difference. In Proceedings of the 6th International Conference on Information Quality (IQ), pages 244–260, Cambridge (MA US), 2001.

Necib B. (2007): Ontology-based semantic query processing in database systems. Berlin, Humboldt Universität, PhD. Thesis, 2007.

Necib C. B. and Freytag J. (2004) "Using Ontologies for Database Query Reformulation". Proceedings of the 18th Conference on Advances in Databases and Information Systems (ADBIS'04), Budapest, Hungary, 2004.

Necib C. B. and Freytag J. (2005) "Query Processing Using Ontologies". Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAISE'05), Porto, Portugal, 2005.

Neves, T. (2008): Desenvolvimento do Módulo de Reformulação de Consultas no Sistema SPEED. Federal University of Pernambuco (UFPE/CIn). Undergraduate Conclusion Monograph. Recife, PE, Brazil.

Ng W. S., Ooi B., Tan K., and Zhou A. (2003) "PeerDB: a P2P-based System for Distributed Data Sharing". (2003) In Proc. of 19th International Conference on Data Eng. (ICDE).

Noy N. and McGuinness D. (2001): Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory, Technical Report KSL-01-05 and Stanford Medical Informatics, Technical Report SMI-2001-0880, March 2001.

Pereira T. (2008): Mapeamento Semântico de Ontologias no SPEED. Federal University of Pernambuco (UFPE/CIn). Undergraduate Conclusion Monograph. Recife, PE, Brazil.

Perez, J., Arenas, M., Gutierrez, C. (2006): Semantics and Complexity of SPARQL. In: 4th International SemanticWeb Conference (ISWC), Athens, GA, USA. November 2006.

Pires C. E. (2007): Um Sistema P2P de Gerenciamento de Dados com Conectividade Baseada em Semântica. Federal University of Pernambuco (UFPE/CIn). Thesis Proposal and Qualification Exam. Recife, PE, Brazil.

Pires C. E. S., Souza D., Lóscio, B. F., and Salgado A. C. (2008): An Ontology-based Approach for Data Management in a P2P System. SPEED Project Technical Report, No. 2. Center for Informatics, Federal University of Pernambuco, 2008.

Pires C.E.S. (2009): Ontology-Based Clustering in a Peer Data Management System. PhD thesis, Center for Informatics, UFPE, 2009 (Work in progress).

Power, R. (2003): Topic Maps for Context Management. In International Symposium on Information and Communication Technologies (ISICT 2003), pp. 199-204 (2003).

Quilitz B. and Leser U. (2008): Querying Distributed RDF Data Sources with SPARQL. In Procceedings of the European Semantic Web Conference (ESWC2008). Springer Verlag, 2008.

Ram, S. and Park, J. (2004): Semantic Conflict Resolution Ontology (SCROL): An ontology for detecting and resolving Data- and Schema-Level Semantic Conflicts. Knowledge and Data Engineering, IEEE Transactions on (2004).

Reynaud C., Safar B. (2007): Exploiting WordNet as Background Knowledge. In: International ISWC'07 Ontology Matching (OM-07) Workshop, Busan, Corea, 2007.

Sabou M., D'Aquin M., Motta E. (2006): Using the Semantic Web as Background Knowledge for Ontology Mapping, In: ISWC'06 Ontology Matching WS, 2006.

Salles M. A. V., Dittrich J.-P., Karakashian S. K., Girard O. R., and Blunschi L.: iTrails: Pay-as-you-go information integration in dataspaces. In Proc. Of VLDB, 2007.

Schilit, B., Adams, N., Want, R. (1994): Context-Aware Computing Applications. In: Proc. Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA.

Scriver A. (2006): Semantic Distance in WordNet: A Simplified and Improved Measure of Semantic Relatedness. Master Dissertation. University of Waterloo, Canada, 2006.

Serafini L., Zanobini S., Sceffer S., Bouquet P. (2006): Matching Hierarchical Classifications with Attributes. In: ESWC'06. pp. 4-18, 2006.

Sheth A. P., Larson J. A. (1990): Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22(3):183{236, September 1990.

Souza D. (2007): Reformulação de Consulta Baseada em Semântica para PDMS. Federal University of Pernambuco (UFPE/CIn). Thesis Proposal and Qualification Exam. Recife, PE, Brazil.

Souza D., Arruda T., Salgado A. C., Tedesco P. and Kedad, Z. (2009): Using Semantics to Enhance Query Reformulation in Dynamic Environments. To appear in the Proocedings of the 13th East European Conference on Advances in Databases and Information Systems – ADBIS 2009. September, 7-10, 2009.

Souza, D., Belian R., Salgado A. C., Tedesco P. (2008). Towards a Context Ontology to Enhance Data Integration Processes. In Proceedings of the 4th Workshop on Ontologies-based Techniques for DataBases in Information Systems and Knowledge Systems (ODBIS). VLDB '08, August 24-30, 2008, Auckland, New Zealand.

Souza, D., Salgado, A. C., Tedesco, P. (2006): Towards a Context Ontology for Geospatial Data Integration. In: Second International Workshop on Semantic-based Geographical Information Systems (SeBGIS'06), Montpellier, France, 2006.

Souza, D., Salgado A. C. (2007): Semantic-Based Query Reformulation for PDMS. In Proceedings of the VI Workshop of Thesis and Dissertations on Databases. 22nd Brazilian Symposium on Databases, João Pessoa, PB, Brazil, 2007.

Stefanidis K., Pitoura E., Vassiliadis P. (2005): On Supporting Context-Aware Preferences in Relational Database Systems. In Proc. of the first International Workshop on Managing Context Information in Mobile and Pervasive Environments (MCMP'2005), in conjunction with MDM 2005, Cyprus (2005).

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001): Chord: a Scalable Peer-to-Peer Lookup Service for Internet Applications". ACM SIGCOMM'01, San Diego, USA. pp. 149-160, 2001.

Strang, T., Linnhoff-Popien, C. (2004): A Context Modeling Survey. In: Workshop on Advanced Context Modeling, Reasoning and Management, In: 6th International Conference on Ubiquitous Computing, Nottingham/England, 2004.

Stuckenschmidt H., Giunchiglia F., and van Harmelen F. (2005): Query processing in ontology-based peer-to-peer systems. In V. Tamma, S. Craneeld, T. Finin, and S. Willmott, editors, Ontologies for Agents: Theory and Experiences. Birkhuser. (2005).

Stuckenschmidt, H. (2002): Ontology-Based Information Sharing in Weakly-Structured Environments. PhD thesis, Faculty of Sciences, Vrije Universiteit Amsterdam, 2002.

Studer R., Benjamins V., Fensel D. (1998): Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering 25(1-2):161–197.

Styltsvig H. (2006): Ontology-based Information Retrieval. PhD Thesis. Roskilde University,May, 2006.

Sung, L. G. A., Ahmed, N., Blanco, R., Li, H, Soliman, M. A., and Hadaller, D. (2005): A Survey of Data Management in Peer-to-Peer Systems. School of Computer Science, University of Waterloo, 2005.

Tatarinov, I., Halevy, A. (2004): Efficient query reformulation in peer-data management systems. In Proceedings of the ACM International Conference on Management of Data (SIGMOD), pages 539{550, 2004.

Tonin I., Bittencourt G. (2000): Forma normal disjuntiva em lógica de primeira ordem. Anais do I Congresso de Lógica Aplicada à Tecnologia (LAPTEC'2000), Faculdade SENAC de Ciências Exatas e Tecnologia (ISBN 85-85795-29-8), pp. 417-429, São Paulo, SP, 11 a 15 de setembro de 2000.

Travassos G., Gurov D., Amaral E. (2002): Introdução à Engenharia de Software Experimental. 2002. Technical Report ES590/02-April. COPPE/UFRJ.

Ullman J. (1997): Information integration using logical views. In Proceedings of the International Conference on Database Theory (ICDT), pages 19{40, 1997.

Vieira V. (2008): CEManTIKA: A Domain-Independent Framework for Designing Context-Sensitive Systems. PhD. Thesis, Centro de Informática - UFPE, Brasil, 2008.

Vieira V., Souza D., Salgado, A. C., Tedesco, P. (2006): Uso e Representação de Contexto em Sistemas Computacionais. In: Cesar A. C. Teixeira; Clever Ricardo G. de Farias; Jair C. Leite; Raquel O. Prates. (Org.). Tópicos em Sistemas Interativos e Colaborativos. São Carlos: UFSCAR, 2006, v. , p. 127-166.

Völker J., Vrandecic D., Sure Y., Hotho A. (2007): Learning Disjointness. In Enrico Franconi, Michael Kifer, Wolfgang May, Proceedings of the 4th European Semantic Web Conference (ESWC'07), volume 4519 of Lecture Notes in Computer Science, pp. 175-189. Springer, June 2007.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S. (2001): Ontology-based Integration of Information: a Survey of Existing Approaches. In Stuckenschmidt, H., ed., IJCAI-01 Workshop: Ontologies and Information Sharing. pp. 108-117.

Wang X., Zhang D., Gu T., Pung H. (2004): Ontology Based Context Modeling and Reasoning using OWL. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, p.18.

Wiederhold Gio (1992): Mediators in the architecture of future information systems. IEEE Computer, 25(3):38{49, 1992.

Xiao H. (2006): Query processing for heterogeneous data integration using ontologies. PhD Thesis in Computer Science. University of Illinois at Chicago, 2006.

Xiao H., Cruz I. (2006). "Ontology-based Query Rewriting in Peer-to-Peer Networks". In Proc. of the 2nd International Conference on Knowledge Engineering and Decision Support, 2006.

Yang, B. and Garcia-Molina, H. (2003): Designing a Super-Peer Network. In Proc. of International Conference on Data Engineering (ICDE'03), Bangalore, India (2003).

Yatskevich M., Giunchiglia F., McNeill F., and Shvaiko P. (2006): OpenKnowledge Deliverable 3.3: A methodology for ontology matching quality evaluation. http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D3.3.pdf, 2006.

Yu S., Al-Jadir L., Spaccapietra S. (2005): Matching User's Semantics with Data Semantics in Location-Based Services. Proc. 1st Workshop on Semantics in Mobile Environments (SME'05), Ayia Napa (Cyprus), 2005.

Zaihrayeu I. (2006): Towards Peer-to-Peer Information Management Systems. PhD thesis, University of Trento, March 2006

Zhao J. (2006): Schema Mediation and Query Processing in Peer Data Management Systems. Master Thesis, The University Of British Columbia, October (2006).

APPENDIX A

SemRef Complementary Functions

In this appendix, we present the complementary pseudo-codes of the algorithms underlying the *SemRef* approach.

```
Build_Exact_Reformulation ($Q_k$, $S_1C_1$, ..., $S_1C_p$)
$Q'_k \leftarrow \varnothing$
For each $S_1C_i$
        $e_i \leftarrow \varnothing$
        For each C in $S_1C_i$
                If $C_i$ is negated in $Q_k$
                    Then $e_i \leftarrow e_i \sqcup \neg C$
                    Else $e_i \leftarrow e_i \sqcup C$
                End If;
        End For;
        $Q'_k \leftarrow Q'_k \sqcap e_i$
End For;
return ($Q'_k$)
End_Build_Exact_Reformulation
```

Figure A.1 Build_Enriched Reformulation Function

```
Build_Enriched_Reformulation ($Q_k$, $S_2C_1$, ..., $S_2C_p$, Neg_$S_2C_1$, ..., Neg_$S_2C_p$)
$Q'_k \leftarrow \varnothing$
For each i
        $e_i \leftarrow \varnothing$
        For each C in $S_2C_i$
                If $C_i$ is negated in $Q_k$
                  Then $e_i \leftarrow e_i \sqcup \neg C$
                  Else $e_i \leftarrow e_i \sqcup C$
                End If;
        End For;
        For each C in Neg_$S_2C_i$
                $e_i \leftarrow e_i \sqcup C$
        End For;

        $Q'_k \leftarrow Q'_k \sqcap e_i$
End For:
returns ($Q'_k$)
End_Build_Enriched_Reformulation
```

Figure A.2 Build_Enriched Reformulation Function

```
**Build_Final_Exact_Reformulation (Q, $Q_1$_exact, …, $Q_m$_exact)**

Q'_exact ← ∅

For each $Q_i$_exact
        Q'_exact ← Q'_exact ⊔ $Q_i$_exact
End For

Return(Q'_exact)

**End Build_Final_Exact_Reformulation;**
```

Figure A.3 The Build_Final_Exact_Reformulation Function

```
**Build_Final_Enriched_Reformulation (Q, $Q_1$_enriched, …, $Q_m$_enriched)**

Q'_enriched ← ∅

For each $Q_i$_enriched
        Q'_enriched ← Q'_enriched ⊔ $Q_i$_enriched
End For

Return(Q'_enriched)

**End Build_Final_Enriched_Reformulation;**
```

Figure A.4 The Build_Final_Enriched_Reformulation Function

APPENDIX B

Education Ontologies

In this appendix, we present the complete ontologies of *Semiport.owl*, *UnivBench.owl* and the domain ontology *UnivCSCMO.owl*.

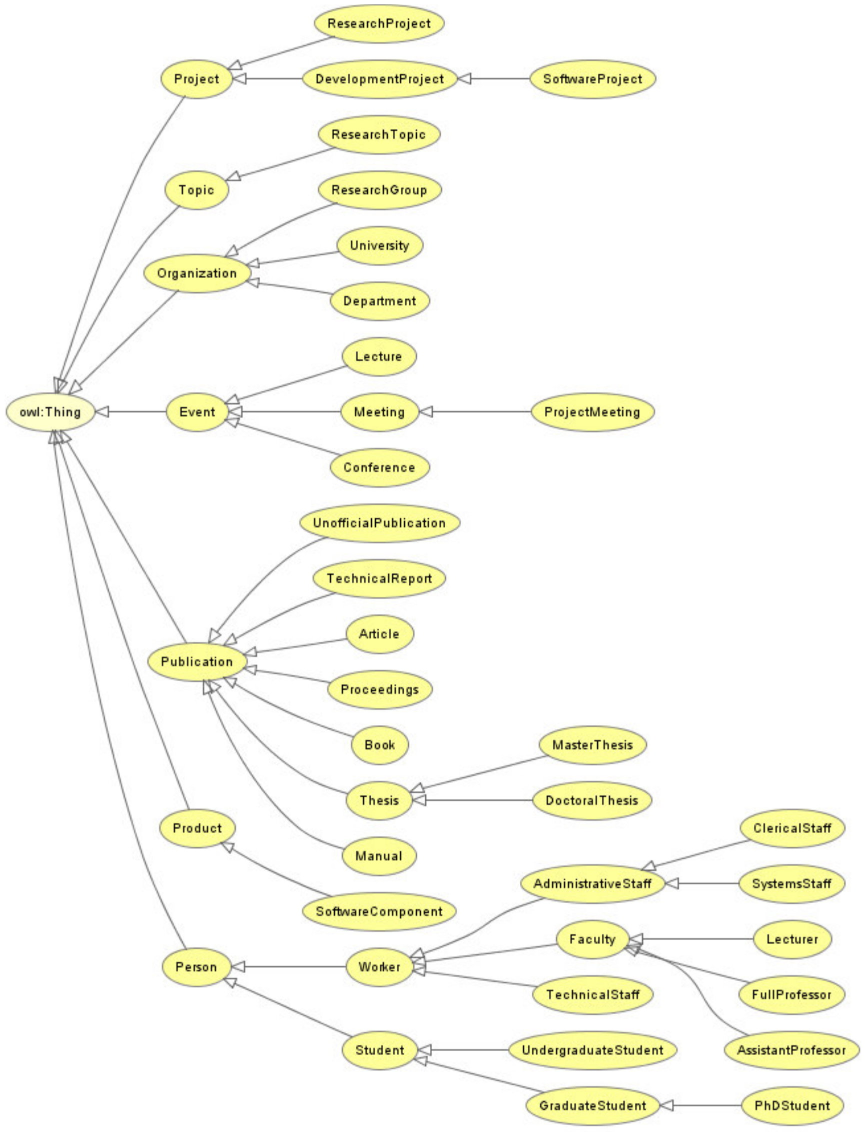


Figure B.1 SemiPort.OWL Ontology

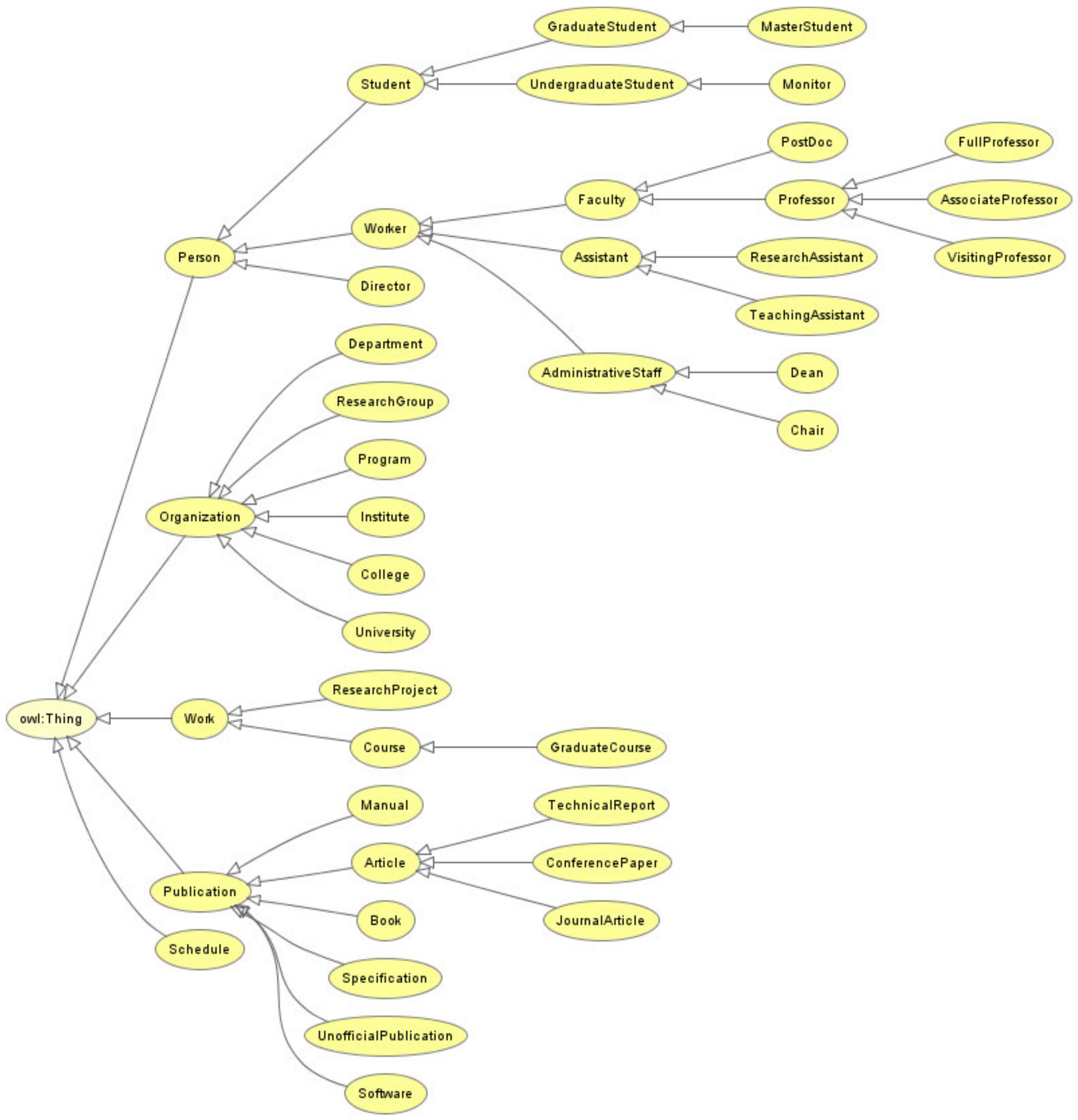


Figure B.2 UnivBench.OWL Ontology

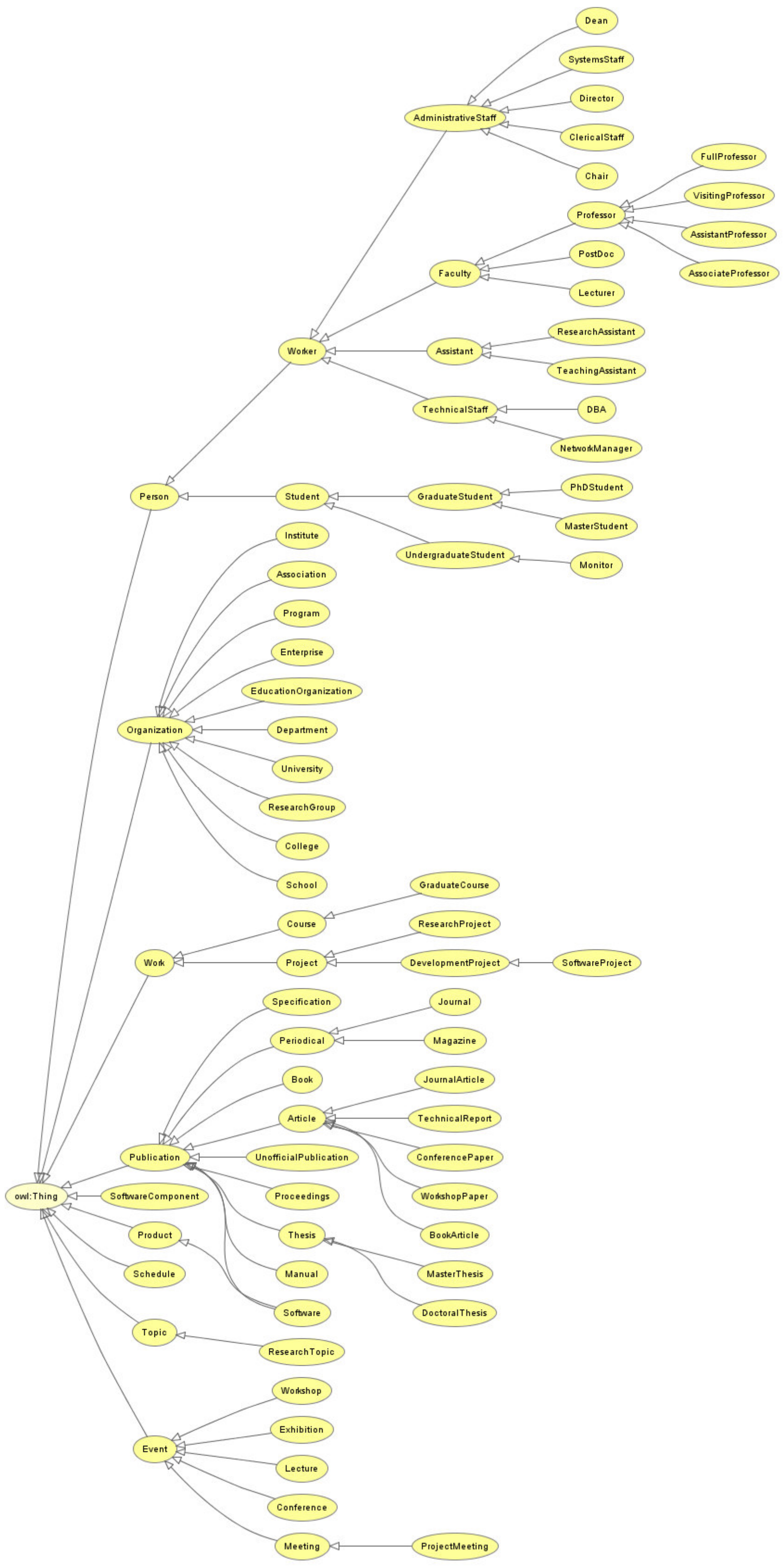
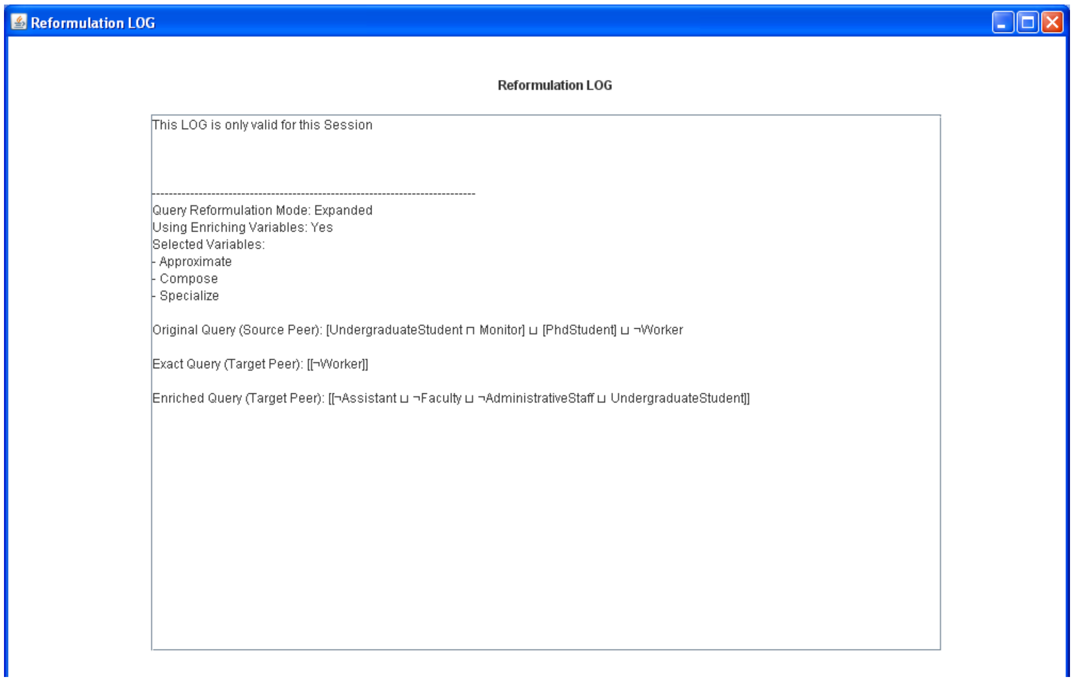


Figure B.3 UnivCsCMO.OWL Ontology

APPENDIX C

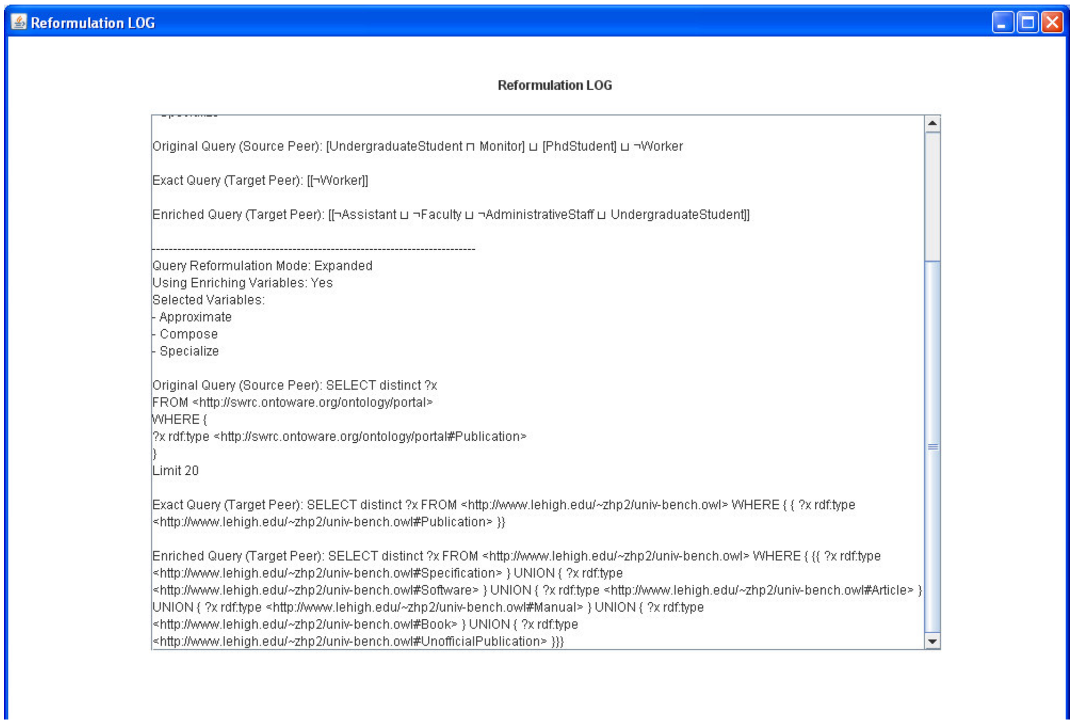
The *SemRef* Query Module

In this appendix, we present additional screenshots of the query module interface.



```
This LOG is only valid for this Session



--------------------------------------------------------------------------------
Query Reformulation Mode: Expanded
Using Enriching Variables: Yes
Selected Variables:
- Approximate
- Compose
- Specialize

Original Query (Source Peer): [UndergraduateStudent ⊓ Monitor] ⊔ [PhdStudent] ⊔ ¬Worker

Exact Query (Target Peer): [[¬Worker]]

Enriched Query (Target Peer): [[¬Assistant ⊔ ¬Faculty ⊔ ¬AdministrativeStaff ⊔ UndergraduateStudent]]
```

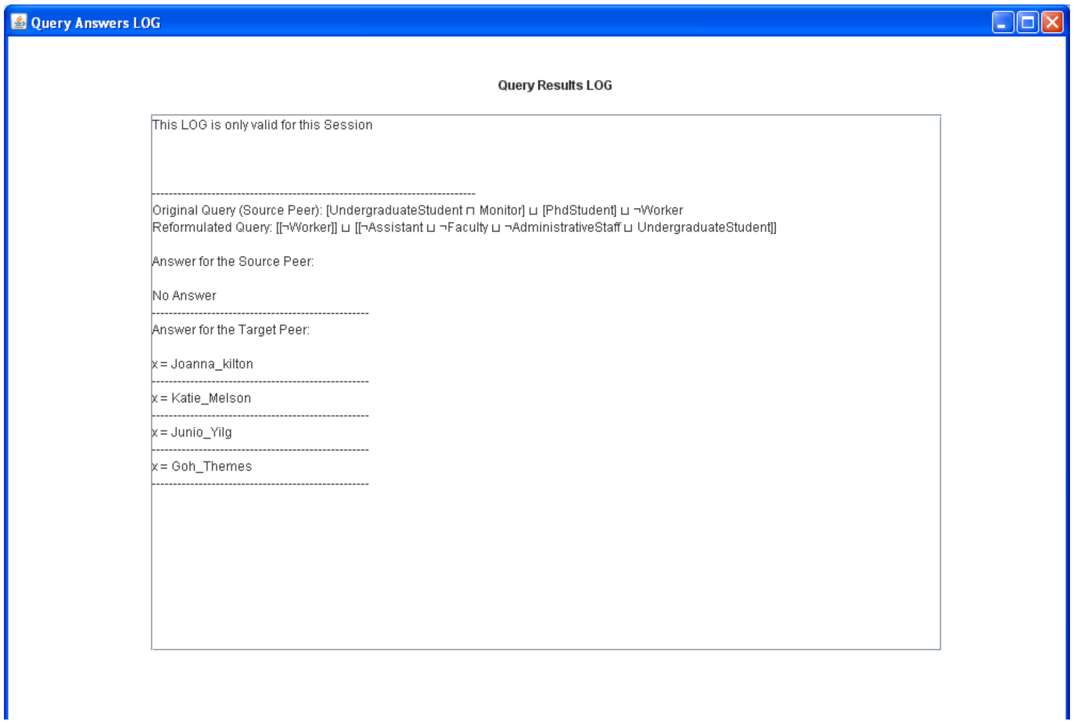
Figure C.1 Reformulation Log for an $\mathcal{ALC}$DL Query



Reformulation LOG

```
Original Query (Source Peer): [UndergraduateStudent ⊓ Monitor] ⊔ [PhdStudent] ⊔ ¬Worker

Exact Query (Target Peer): [[¬Worker]]

Enriched Query (Target Peer): [[¬Assistant ⊔ ¬Faculty ⊔ ¬AdministrativeStaff ⊔ UndergraduateStudent]]

-----------------------------------------------------------------------------
Query Reformulation Mode: Expanded
Using Enriching Variables: Yes
Selected Variables:
- Approximate
- Compose
- Specialize

Original Query (Source Peer): SELECT distinct ?x
FROM <http://swrc.ontoware.org/ontology/portal>
WHERE {
?x rdf:type <http://swrc.ontoware.org/ontology/portal#Publication>
}
Limit 20

Exact Query (Target Peer): SELECT distinct ?x FROM <http://www.lehigh.edu/~zhp2/univ-bench.owl> WHERE { { ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#Publication> }}

Enriched Query (Target Peer): SELECT distinct ?x FROM <http://www.lehigh.edu/~zhp2/univ-bench.owl> WHERE { {{ ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#Specification> } UNION { ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#Software> } UNION { ?x rdf:type <http://www.lehigh.edu/~zhp2/univ-bench.owl#Article> }
UNION { ?x rdf:type <http://www.lehigh.edu/~zhp2/univ-bench.owl#Manual> } UNION { ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#Book> } UNION { ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#UnofficialPublication> }}}
```

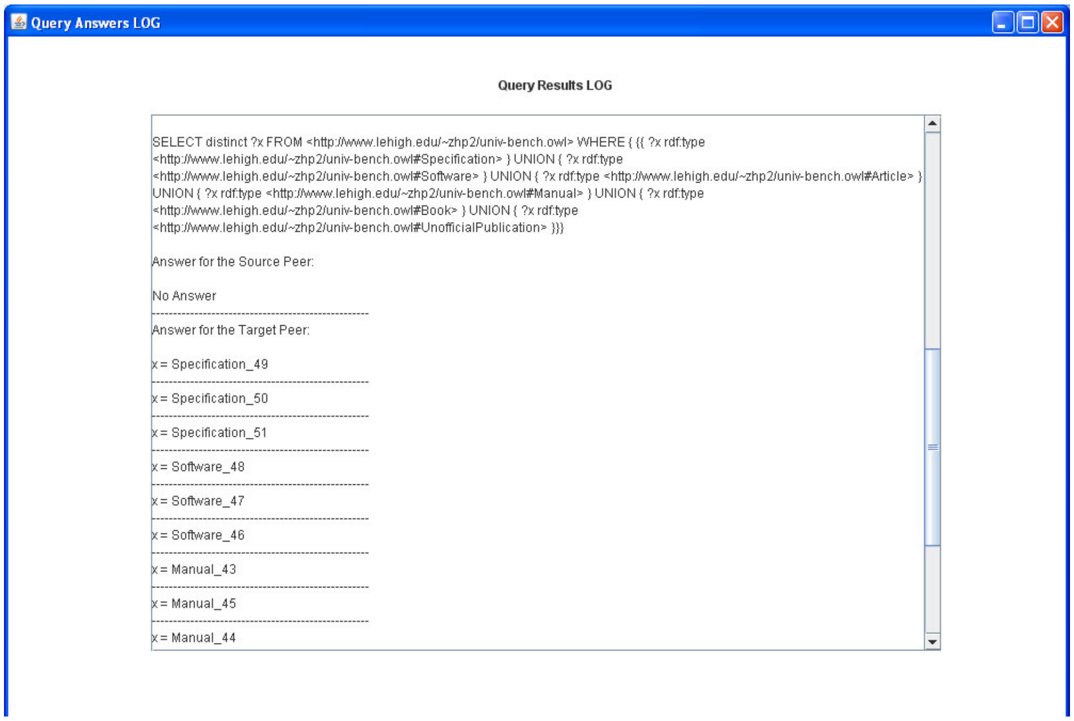
Figure C.2 Reformulation Log for a SPARQL Query



Query Results LOG

```
This LOG is only valid for this Session



-----------------------------------------------------------------------------
Original Query (Source Peer): [UndergraduateStudent ⊓ Monitor] ⊔ [PhdStudent] ⊔ ¬Worker
Reformulated Query: [[¬Worker]] ⊔ [[¬Assistant ⊔ ¬Faculty ⊔ ¬AdministrativeStaff ⊔ UndergraduateStudent]]

Answer for the Source Peer:

No Answer
-------------------------------------------------
Answer for the Target Peer:

x = Joanna_kilton
-------------------------------------------------
x = Katie_Melson
-------------------------------------------------
x = Junio_Yilg
-------------------------------------------------
x = Goh_Themes
-------------------------------------------------
```

Figure C.3 Answers' Log for an $\mathcal{ALC}$ DL Query



```
SELECT distinct ?x FROM <http://www.lehigh.edu/~zhp2/univ-bench.owl> WHERE { {{ ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#Specification> } UNION { ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#Software> } UNION { ?x rdf:type <http://www.lehigh.edu/~zhp2/univ-bench.owl#Article> }
UNION { ?x rdf:type <http://www.lehigh.edu/~zhp2/univ-bench.owl#Manual> } UNION { ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#Book> } UNION { ?x rdf:type
<http://www.lehigh.edu/~zhp2/univ-bench.owl#UnofficialPublication> }}}

Answer for the Source Peer:

No Answer
--------------------------------------------------
Answer for the Target Peer:

x = Specification_49
--------------------------------------------------
x = Specification_50
--------------------------------------------------
x = Specification_51
--------------------------------------------------
x = Software_48
--------------------------------------------------
x = Software_47
--------------------------------------------------
x = Software_46
--------------------------------------------------
x = Manual_43
--------------------------------------------------
x = Manual_45
--------------------------------------------------
x = Manual_44
```

Figure C.4 Answers' Log for a SPARQL Query

APPENDIX D

Travel Ontologies

In this appendix, we present the complete taxonomies concerning the knowledge domain of Tourism.

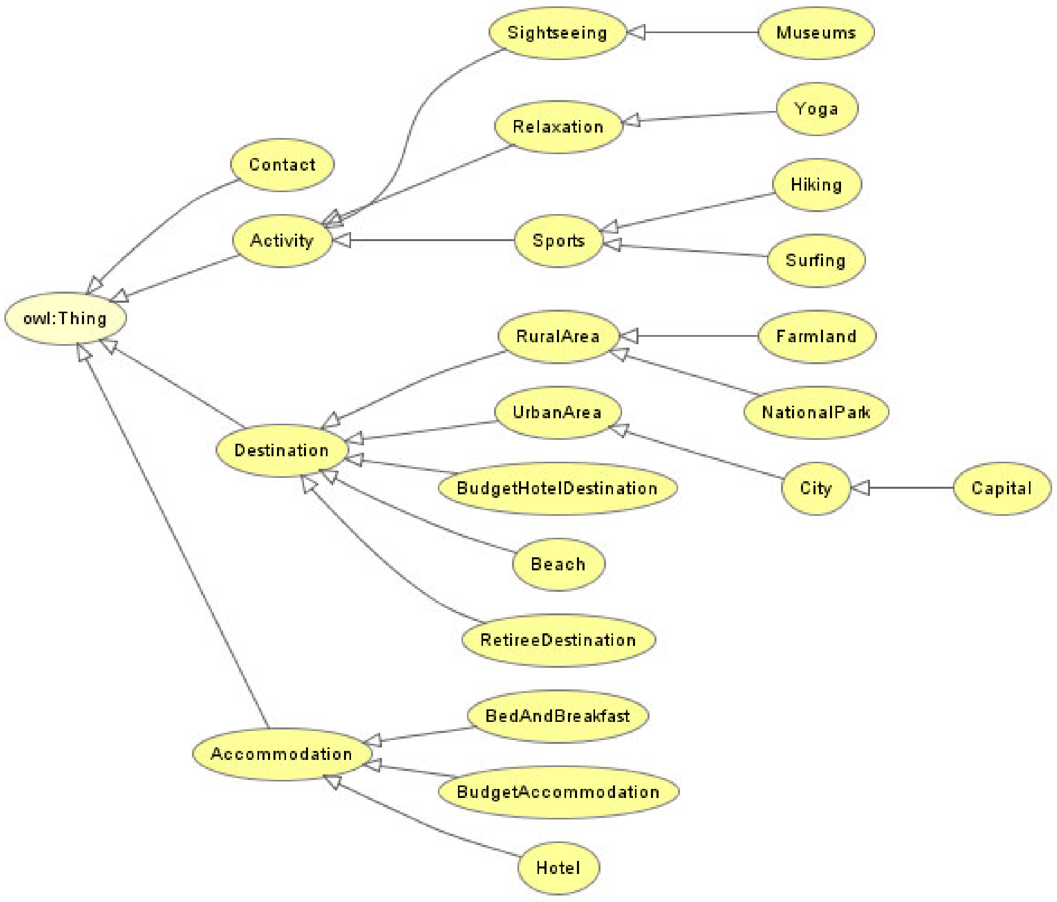


Figure D.1 TravelCLO1.OWL Ontology

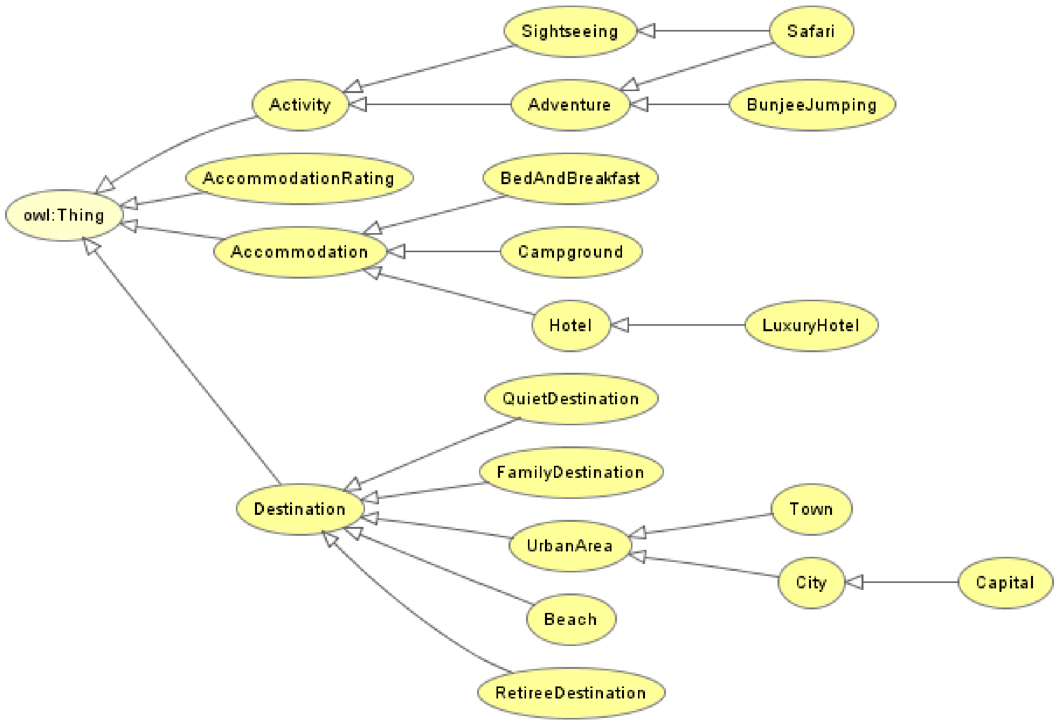


Figure D.2 TravelCLO2.OWL Ontology

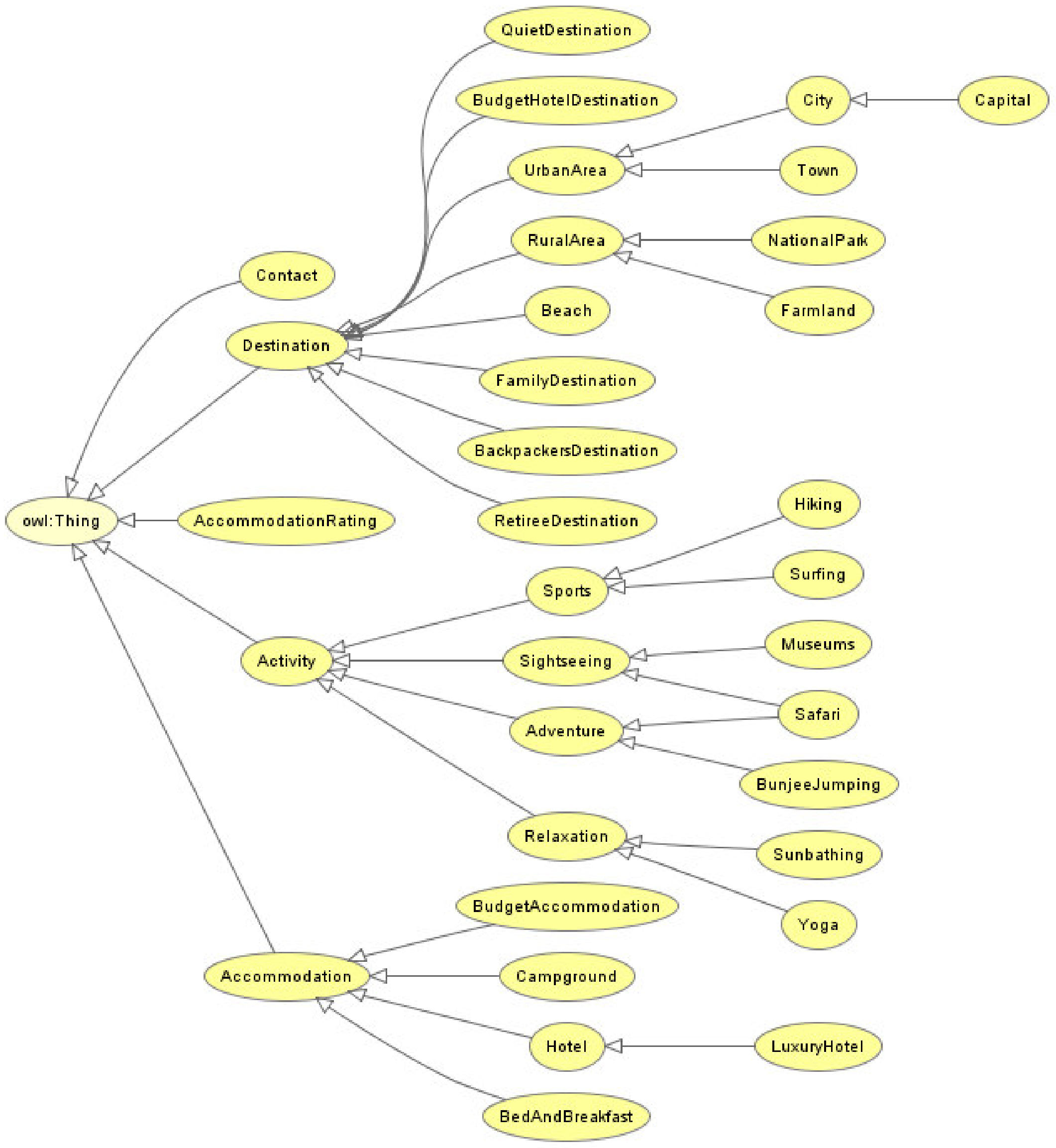


Figure D.3 TravelCMO.OWL Ontology

APPENDIX E

SemRef Experimentation

In this appendix, we present additional details regarding queries and conducted experiments of the *SemRef* experimentation.

E.1 Complete Set of Submitted Queries

a. Education

The set of queries are divided into two groups:

- Queries From P_1 to P_2:

  Q1: Student

  Q2: Student ⊓ UndergraduateStudent

  Q3: Worker

  Q4: AssistantProfessor

  Q5: FullProfessor ⊔ Lecturer

  Q6: Proceedings ⊔ Thesis ⊔ ¬TechnicalReport

  Q7: Lecture ⊔ Meeting ⊔ Conference

  Q8: ¬Manual

  Q9: ¬PhDStudent

  Q10: UndergraduateStudent ⊔ PhDStudent

  Q11: [AdministrativeStaff ⊓ ClericalStaff] ⊔ [Faculty ⊓ Lecturer]

  Q12: [SystemsStaff ⊓ Worker] ⊔ [Student ⊓ ¬PhDStudent]

  Q13: [AssistantProfessor ⊓ Faculty] ⊔ ¬Lecturer

  Q14: ¬SoftwareComponent

  Q15: Project ⊔ Publication

  Q16: Faculty ⊔ ¬AssistantProfessor

Q17: [Student ⊓ PhDStudent] ⊔ [Worker ⊓ SystemsStaff] ⊔ [Lecturer ⊓ Faculty]

Q18: Project ⊔ ¬SofwareProject ⊔ Thesis

Q19: ¬Book ⊔ ¬Article

Q20: [Event ⊓ Meeting] ⊔ ¬Lecture

- Queries from P_2 to P_1

Q21: Professor

Q22: Professor ⊓ PostDoc

Q23: Department

Q24: Article ⊔ Book

Q25: ¬MasterStudent

Q26: ResearchProject ⊔ Course

Q27: Worker ⊓ Dean

Q28: College ⊔ Institute ⊔ ¬Program

Q29: ¬Monitor

Q30: [Student ⊓ Monitor] ⊔ [GraduateStudent ⊓ MasterStudent]

Q31: Dean ⊔ Director

Q32: UndergraduateStudent ⊔ ¬MasterStudent

Q33: [Student ⊓ Monitor] ⊔ [Worker ⊓ Chair] ⊔ [Assistant ⊓ ¬TeachingAssistant] ⊔ [Faculty ⊓ AssociateProfessor]

Q34: ¬FullProfessor ⊔ ¬VisitingProfessor

Q35: Director ⊔ Chair ⊔ PostDoc

b. Travel

Similarly, the set of queries are divided into two groups:

- Queries From P_1 to P_2:

Q36: Contact

Q37: Sports ⊔ Sightseeing

Q38: ¬RetireeDestination

Q39: BedAndBreakfast

Q40: Activity ⊓ Sightseeing

Q41: [Destination ⊓ Beach] ⊔ ¬RuralArea

Q42: Hotel ⊔ BudgetAccomodation

Q43: [Destination ⊓ Capital] ⊔ [Destination ⊓ BudgetHotelDestination]

Q44: [Destination ⊓ Capital] ⊔ [Destination ⊓ ¬Farmland] ⊔ ¬NationalPark

Q45: ¬BudgetHotelDestination ⊔ ¬Farmland ⊔ ¬NationalPark

- Queries from P_2 to P_1

Q46: AccomodationRating

Q47: Safari

Q48: BunjeeJumping ⊔ Sightseeing

Q49: Campgroung ⊔ ¬Hotel

Q50: [Hotel ⊓ LuxuryHotel] ⊔ ¬BedAndBreakfast

Q51: UrbanArea ⊓ Town

Q52: ¬FamilyDestination

Q53: [Destination ⊓ RetireeDestination] ⊔ [UrbanArea ⊓ City]

Q54: Town ⊔ City ⊔ ¬Beach

Q55: [City ⊓ Capital] ⊔ [Destination ⊓ Town] ⊔ ¬QuietDestination

E.2 Complete Set of Experiments

a. Queries without semantics – Mode: *Restricted*

Query	Q_{exact}
Q1	[[Student]]
Q2	[Student ⊓ UndergraduateStudent]
Q3	[[Worker]]
Q4	∅
Q5	FullProfessor
Q6	¬TechnicalReport
Q7	∅
Q8	¬Manual
Q9	∅
Q10	UndergraduateStudent
Q11	∅
Q12	∅
Q13	∅
Q14	∅
Q15	Publication
Q16	Faculty
Q17	∅
Q18	∅
Q19	[[¬Book]] ⊔ [[¬Article]]
Q20	∅
Q21	∅
Q22	∅
Q23	[[Department]]
Q24	[[Article]] ⊔ [[Book]]
Q25	∅
Q26	[[ResearchProject]]
Q27	∅
Q28	∅
Q29	∅
Q30	∅
Q31	∅
Q32	[[UndergraduateStudent]]
Q33	∅
Q34	[[¬FullProfessor]]
Q35	∅
Q36	∅
Q37	[[Sightseeing]]
Q38	[[¬RetireeDestination]]
Q39	[[BedAndBreakfast]]
Q40	[[Activity] ⊓ [Sightseeing]]
Q41	[[Destination] ⊓ [Beach]]
Q42	[[Hotel]]
Q43	[[Destination] ⊓ [Capital]]
Q44	[[Destination] ⊓ [Capital]]
Q45	∅
Q46	∅
Q47	∅
Q48	[[Sightseeing]]
Q49	[[¬Hotel]]
Q50	[[¬BedAndBreakfast]]
Q51	∅
Q52	∅
Q53	[[Destination] ⊓ [RetireeDestination]] ⊔ [[UrbanArea] ⊓ [City]]
Q54	[[City]] ⊔ [[¬Beach]]
Q55	[[City] ⊓ [Capital]]

b. Queries with semantics – Mode: Restricted with Enriching Variables

Query	Spec	Gen	Approx	Comp	Q_{exact}	$Q_{enriched}$
Q1	X		X		[[Student]]	∅
Q2	X		X		[Student ⊓ UndergraduateStudent]	∅
Q3	X		X		[[Worker]]	∅
Q4	X		X		∅	[[VisitingProfessor]]
Q5	X		X		FullProfessor	∅
Q6	X		X		¬TechnicalReport	∅
Q7	X		X		∅	∅
Q8	X		X		¬Manual	∅
Q9		X		X	∅	¬GraduateStudent
Q10		X		X	UndergraduateStudent	
Q11		X		X	∅	[[Worker] ⊓ [AdministrativeStaff]] ⊔ [[Worker] ⊓ [Faculty]]
Q12	X		X	X	∅	[[Dean ⊔ Director ⊔ Chair] ⊓ [Assistant ⊔ Faculty ⊔ AdministrativeStaff]] ⊔ [[UndergraduateStudent ⊔ GraduateStudent] ⊓ [¬MasterStudent]]
Q13	X			X	∅	[[Course ⊔ ResearchProject] ⊓ [PostDoc ⊔ Professor]]
Q14	X	X	X	X	∅	¬Software
Q15	X	X	X	X	Publication	∅
Q16	X	X	X	X	Faculty	∅
Q17	X	X	X		∅	[[Person ⊔ UndergraduateStudent ⊔ GraduateStudent] ⊓ [MasterStudent ⊔ GraduateStudent]] ⊔ [[Person ⊔ Assistant ⊔ Faculty ⊔ AdministrativeStaff] ⊓ [Dean ⊔ Director ⊔ Chair ⊔ AdministrativeStaff]] ⊔ [[PostDoc ⊔ Professor ⊔ Faculty] ⊓ [AdministrativeStaff ⊔ Assistant ⊔

						Worker ⊔ PostDoc ⊔ Professor]]
Q18	X	X	X		∅	[[Work ⊔ ResearchProject]] ⊔ [[Publication]]
Q19	X	X	X		[[¬Book]] ⊔ [[¬Article]]	∅
Q20	X	X	X		∅	∅
Q21	X		X		∅	[[Lecturer ⊔ FullProfessor ⊔ AssistantProfessor]]
Q22	X		X		∅	[[Lecturer ⊔ FullProfessor ⊔ AssistantProfessor] ⊓ [Lecturer]]
Q23		X	X		[[Department]]	∅
Q24		X	X		[[Article]] ⊔ [[Book]]	∅
Q25		X	X		∅	[[¬PhDStudent ⊔ ¬GraduateStudent]]
Q26	X			X	[[ResearchProject]]	∅
Q27	X		X	X	∅	[[TechnicalStaff ⊔ Faculty ⊔ AdministrativeStaff] ⊓ [ClericalStaff ⊔ SystemsStaff]]
Q28	X		X	X	∅	∅
Q29	X	X		X	∅	[[¬UndergraduateStudent]]
Q30	X	X	X	X	∅	[[Person ⊔ GraduateStudent ⊔ UndergraduateStudent] ⊓ [UndergraduateStudent]] ⊔ [[Student ⊔ PhDStudent ⊔ ResearchProject] ⊓ [PhDStudent ⊔ GraduateStudent]]
Q31	X	X	X	X	∅	[[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]]
Q32	X	X	X	X	[[UndergraduateStudent]]	∅
Q33		X	X	X	∅	[[Person] ⊓ [UndergraduateStudent]] ⊔ [[Person] ⊓ [ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[TechnicalStaff ⊔ AdministrativeStaff ⊔ Worker] ⊓ [ResearchProject]]
Q34		X	X	X	[[¬FullProfessor]]	∅
Q35		X	X	X	∅	[[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[Lecturer ⊔ Faculty]]
Q36	X	X			∅	∅

Q37	X	X			[[Sightseeing]]	∅
Q38	X	X			[[¬RetireeDestination]]	∅
Q39			X	X	[[BedAndBreakfast]]	∅
Q40			X	X	[[Activity] ⊓ [Sightseeing]]	∅
Q41			X	X	[[Destination] ⊓ [Beach]]	∅
Q42	X		X	X	[[Hotel]]	∅
Q43	X		X	X	[[Destination] ⊓ [Capital]]	∅
Q44	X	X	X	X	[[Destination] ⊓ [Capital]]	∅
Q45	X	X	X	X	∅	[[¬UrbanArea ⊔ ¬Beach ⊔ ¬RetireeDestination ⊔ ¬FamilyDestination ⊔ ¬QuietDestination ⊔ ¬Destination]]
Q46		X	X		∅	∅
Q47		X	X		∅	[[Museums ⊔ Sightseeing]]
Q48	X	X			[[Sightseeing]]	∅
Q49	X	X			[[¬Hotel]]	∅
Q50		X		X	[[¬BedAndBreakfast]]	∅
Q51		X		X	∅	[[Destination] ⊓ [UrbanArea]]
Q52	X		X		∅	[[¬BudgetHotelDestination ⊔ ¬RuralArea ⊔ ¬UrbanArea ⊔ ¬Beach ⊔ ¬RetireeDestination]]
Q53	X		X		[[Destination] ⊓ [RetireeDestination]] ⊔ [[UrbanArea] ⊓ [City]]	∅
Q54	X	X	X	X	[[City]] ⊔ [[¬Beach]]	∅
Q55	X	X	X	X	[[City] ⊓ [Capital]]	∅

c. Queries with semantics – Mode: Expanded and Enriching Variables

Query	Spec	Gen	Approx	Comp	Q_{exact}	$Q_{enriched}$
Q1	X		X		[[Student]]	[[UndergraduateStudent ⊔ GraduateStudent]]
Q2	X		X		[Student ⊓ UndergraduateStudent]	[[UndergraduateStudent ⊔ GraduateStudent] ⊓ [Monitor]]
Q3	X		X		[[Worker]]	[[Assistant ⊔ Faculty ⊔ AdministrativeStaff]]
Q4	X		X		∅	[[VisitingProfessor]]
Q5	X		X		FullProfessor	[[VisitingProfessor]] ⊔ [[PostDoc ⊔ Professor]]
Q6	X		X		¬TechnicalReport	[[¬ConferencePaper ⊔ ¬JournalArticle]]
Q7	X		X		∅	∅
Q8	X		X		¬Manual	∅
Q9		X		X	∅	¬GraduateStudent
Q10		X		X	UndergraduateStudent	[[Student ⊔ Course ⊔ ResearchProject]] ⊔ [[GraduateStudent]]
Q11		X		X	∅	[[Worker] ⊓ [AdministrativeStaff]] ⊔ [[Worker] ⊓ [Faculty]]
Q12	X			X	∅	[[Dean ⊔ Director ⊔ Chair] ⊓ [Assistant ⊔ Faculty ⊔ AdministrativeStaff]] ⊔ [[UndergraduateStudent ⊔ GraduateStudent] ⊓ [¬MasterStudent]]
Q13	X			X	∅	[[Course ⊔ ResearchProject] ⊓ [PostDoc ⊔ Professor]]
Q14	X	X	X	X	∅	¬Software
Q15	X	X	X	X	Publication	[[Work ⊔ ResearchProject]] ⊔ [[Specification ⊔ Software ⊔ Article ⊔ Manual ⊔ Book ⊔ UnofficialPublication]]
Q16	X	X	X	X	Faculty	[[AdministrativeStaff ⊔ Assistant ⊔ Worker ⊔ PostDoc ⊔ Professor]] ⊔ [[¬VisitingProfessor ⊔ ¬Professor ⊔ ¬Course ⊔ ¬ResearchProject ⊔ AssociateProfessor ⊔ FullProfessor]]
Q17	X	X	X		∅	[[Person ⊔ UndergraduateStudent ⊔ GraduateStudent] ⊓ [MasterStudent ⊔ GraduateStudent]] ⊔ [[Person ⊔ Assistant ⊔ Faculty ⊔ AdministrativeStaff] ⊓ [Dean ⊔

						Director ⊔ Chair ⊔ AdministrativeStaff]] ⊔ [[PostDoc ⊔ Professor ⊔ Faculty] ⊓ [AdministrativeStaff ⊔ Assistant ⊔ Worker ⊔ PostDoc ⊔ Professor]]
Q18	X	X	X		∅	[[Work ⊔ ResearchProject]] ⊔ [[Publication]]
Q19	X	X	X		[[¬Book]] ⊔ [[¬Article]]	[[¬Publication]] ⊔ [[¬Publication ⊔ ¬ConferencePaper ⊔ ¬TechnicalReport ⊔ ¬JournalArticle]]
Q20	X	X	X		∅	∅
Q21	X		X		∅	[[Lecturer ⊔ FullProfessor ⊔ AssistantProfessor]]
Q22	X		X		∅	[[Lecturer ⊔ FullProfessor ⊔ AssistantProfessor] ⊓ [Lecturer]]
Q23		X	X		[[Department]]	[Organization]]
Q24		X	X		[[Article]] ⊔ [[Book]]	[[Publication]] ⊔ [[Publication]]
Q25		X	X		∅	[[¬PhDStudent ⊔ ¬GraduateStudent]]
Q26	X			X	[[ResearchProject]]	[UndergraduateStudent ⊔ FullProfessor ⊔ AssistantProfessor ⊔ GraduateStudent]] ⊔ [[UndergraduateStudent ⊔ GraduateStudent ⊔ AssistantProfessor ⊔ FullProfessor]]
Q27	X		X	X	∅	[[TechnicalStaff ⊔ Faculty ⊔ AdministrativeStaff] ⊓ [ClericalStaff ⊔ SystemsStaff]
Q28	X		X	X	∅	∅
Q29	X	X		X	∅	[[¬UndergraduateStudent]]
Q30	X	X	X	X	∅	[[Person ⊔ GraduateStudent ⊔ UndergraduateStudent] ⊓ [UndergraduateStudent]] ⊔ [[Student ⊔ PhDStudent ⊔ ResearchProject] ⊓ [PhDStudent ⊔ GraduateStudent]]
Q31	X	X	X	X	∅	[[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]]
Q32	X	X	X	X	[[UndergraduateStudent]]	[[Student ⊔ ResearchProject]] ⊔ [[¬PhDStudent ⊔ ¬GraduateStudent]]
Q33		X	X	X	∅	[[Person] ⊓ [UndergraduateStudent]] ⊔ [[Person] ⊓ [ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[TechnicalStaff ⊔ AdministrativeStaff ⊔ Worker] ⊓ [ResearchProject]]
Q34		X	X	X	[[¬FullProfessor]]	[[¬ResearchProject ⊔ AssistantProfessor]] ⊔ [[¬FullProfessor ⊔ ¬AssistantProfessor]]
Q35		X	X	X	∅	[[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[ClericalStaff ⊔ SystemsStaff ⊔ AdministrativeStaff]] ⊔ [[Lecturer ⊔ Faculty]]
Q36	X	X			∅	∅
Q37	X	X			[[Sightseeing]]	[[Activity]] ⊔ [[Activity ⊔ Safari]]

Q38	X	X			[[¬RetireeDestination]]	[[¬Destination]]
Q39			X	X	[[BedAndBreakfast]]	∅
Q40			X	X	[[Activity] ⊓ [Sightseeing]]	∅
Q41			X	X	[[Destination] ⊓ [Beach]]	[[¬Beach ⊔ ¬RetireeDestination ⊔ ¬FamilyDestination ⊔ ¬QuietDestination ⊔ UrbanArea]]
Q42	X		X	X	[[Hotel]]	[[LuxuryHotel]]
Q43	X		X	X	[[Destination] ⊓ [Capital]]	[[UrbanArea ⊔ Beach ⊔ RetireeDestination ⊔ FamilyDestination ⊔ QuietDestination] ⊓ [UrbanArea ⊔ Beach ⊔ RetireeDestination ⊔ FamilyDestination ⊔ QuietDestination]]
Q44	X	X	X	X	[[Destination] ⊓ [Capital]]	[[UrbanArea ⊔ Beach ⊔ RetireeDestination ⊔ FamilyDestination ⊔ QuietDestination] ⊓ [City]]
Q45	X	X	X	X	∅	[[¬UrbanArea ⊔ ¬Beach ⊔ ¬RetireeDestination ⊔ ¬FamilyDestination ⊔ ¬QuietDestination ⊔ ¬Destination]]
Q46		X	X		∅	∅
Q47		X	X		∅	[[Museums ⊔ Sightseeing]]
Q48	X	X			[[Sightseeing]]	[[Activity ⊔ Museums]]
Q49	X	X			[[¬Hotel]]	[[¬Accommodation ⊔ BedAndBreakfast]]
Q50		X		X	[[¬BedAndBreakfast]]	[[Accommodation] ⊓ [Hotel]] ⊔ [[¬Accommodation ⊔ Hotel]]
Q51		X		X	∅	[[Destination] ⊓ [UrbanArea]]
Q52	X		X		∅	[[¬BudgetHotelDestination ⊔ ¬RuralArea ⊔ ¬UrbanArea ⊔ ¬Beach ⊔ ¬RetireeDestination]]
Q53	X		X		[[Destination] ⊓ [RetireeDestination]] ⊔ [[UrbanArea] ⊓ [City]]	[[BudgetHotelDestination ⊔ RuralArea ⊔ UrbanArea ⊔ Beach ⊔ RetireeDestination] ⊓ [BudgetHotelDestination ⊔ RuralArea ⊔ UrbanArea ⊔ Beach]] ⊔ [[BudgetHotelDestination ⊔ Beach ⊔ RetireeDestination ⊔ City] ⊓ [Capital]]
Q54	X	X	X	X	[[City]] ⊔ [[¬Beach]]	[[City ⊔ UrbanArea]] ⊔ [[UrbanArea ⊔ Capital]] ⊔ [[¬BudgetHotelDestination ⊔ ¬RuralArea ⊔ ¬UrbanArea ⊔ ¬RetireeDestination ⊔ ¬Destination]]
Q55	X	X	X	X	[[City] ⊓ [Capital]]	[[UrbanArea ⊔ Capital] ⊓ [City]] ⊔ [[BudgetHotelDestination ⊔ RuralArea ⊔ UrbanArea ⊔ Beach ⊔ RetireeDestination] ⊓ [City ⊔ UrbanArea]] ⊔ [[¬BudgetHotelDestination ⊔ ¬RuralArea ⊔ ¬UrbanArea ⊔ ¬Beach ⊔ ¬RetireeDestination ⊔ ¬Destination]]

E.3 A Small Fragment of the Experiments Using Queries Expressed in SPARQL

Reformulation Queries without semantics – Mode: *Restricted*

Query	Q_{exact}
Q28	SELECT distinct ?x FROM <http://swrc.ontoware.org/ontology/portal> WHERE { { ?x rdf:type <http://swrc.ontoware.org/ontology/portal#ResearchProject> }}

Reformulation Queries with semantics – Mode: *Restricted*

Query	S	G	A	C	Q_{exact}	$Q_{enriched}$
Q30	X	X	X	X	∅	SELECT distinct ?x FROM <http://swrc.ontoware.org/ontology/portal> WHERE { { ?x rdf:type <http://swrc.ontoware.org/ontology/portal#UndergraduateStudent> }}
Q32	X	X	X	X	SELECT distinct ?x FROM <http://swrc.ontoware.org/ontology/portal> WHERE { { ?x rdf:type <http://swrc.ontoware.org/ontology/portal#UndergraduateStudent> }}	∅

Reformulation Queries with semantics – Mode: *Complete*

Query	S	G	A	C	Q_{exact}	$Q_{enriched}$
Q21	X		X		∅	SELECT distinct ?x FROM <http://swrc.ontoware.org/ontology/portal> WHERE { {{ ?x rdf:type <http://swrc.ontoware.org/ontology/portal#Lecturer> } UNION { ?x rdf:type <http://swrc.ontoware.org/ontology/portal#FullProfessor> } UNION { ?x rdf:type <http://swrc.ontoware.org/ontology/portal#AssistantProfessor> }}}
Q32	X	X	X	X	SELECT distinct ?x FROM <http://swrc.ontoware.org/ontology/portal> WHERE { { ?x rdf:type <http://swrc.ontoware.org/ontology/portal#UndergraduateStudent> }}	SELECT distinct ?x FROM <http://swrc.ontoware.org/ontology/portal> WHERE { {{ ?x rdf:type <http://swrc.ontoware.org/ontology/portal#Student> } UNION { ?x rdf:type <http://swrc.ontoware.org/ontology/portal#ResearchProject> }}}