Using Semantics to Enhance Query Reformulation in Dynamic Environments

Damires Souza[1,2], Thiago Arruda[1], Ana Carolina Salgado[1], Patricia Tedesco[1], Zoubida Kedad[3]
[1] Center for Informatics/UFPE, PO Box 7851, 50732-970, Recife, PE, Brazil
{dysf, tan, acs, pcart}@cin.ufpe.br
[2] Federal Institute of Education, Science and Technology of Paraiba/IFPB, Brazil
[3] Université de Versailles/UVSQ, 45 Avenue des Etats-Unis, 78035 Versailles, France
zoubida.kedad@prism.uvsq.fr

Abstract. One key issue for query answering in dynamic environments is the reformulation of a query posed at a peer into another one over a target peer. Traditional approaches usually accomplish such task by considering equivalence mappings. 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. 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 paper, we propose a semantic-based approach which brings together both query enrichment and query reformulation techniques. To this end, we make use of semantics acquired from a set of mappings (that extend the ones commonly found) and from the context. We present the algorithms underlying our approach, examples illustrating how they work, and some promising experimental results.

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

1 Introduction

Query answering among data sources in networked environments is a challenge which has been addressed in different dynamic settings, such as Peer Data Management Systems (PDMS) [1, 2]. These environments have a diversity of perspectives and are composed by autonomous data sources (*peers*) which are linked by means of mappings (here called *correspondences*). One special problem concerning these architectures is how to exploit the correspondences in order to answer queries and provide users with relevant results. Particularly, a crucial point is how to reformulate queries among the peers in such a way that the resulting set of answers expresses, as closely as possible, what the users defined as important at query submission time, considering the current status of the environment.

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, what 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 work, we present a query reformulation approach which uses semantics as a way to better deal with these mentioned aspects. Thus, in order to capture user preferences, query semantics and environmental parameters we use contextual information [3]. 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 equivalence correspondences and (ii) an *enriched* one, resulting from the set of the 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, according to their preferences.

We address query reformulation 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 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 [4, 5]. Instead, we focus on reformulating a query posed at a source peer in terms of a target peer. In this paper, we present the algorithms underlying our approach. To clarify matters, we provide some examples illustrating how they can be used. We discuss some experiments that have yielded promising results.

This paper is organized as follows: Section 2 discusses the semantic elements used in this work; Section 3 describes the *SemRef* algorithm and how it works; Section 4 presents some experimental results. Related work is discussed in Section 5. Finally, Section 6 draws our conclusions and points out some future work.

2 Applying Semantics to Query Reformulation in PDMS

We have instantiated our approach in a PDMS, although it can be instantiated in any dynamic distributed environment. PDMS consist of a set of peers, each one with an associated schema that represents the data to be shared with other peers. In such systems, schema matching techniques are used to establish schema correspondences which form the basis for query reformulation. Queries submitted at a peer are answered with data residing at that peer and with data that is reached through correspondences over the network of peers.

PDMS are highly dynamic systems. To help matters, we use semantics as a way to enhance tasks such as peer clustering, and, particularly, query reformulation. In our approach, the peers are clustered according to the same knowledge domain (e.g., *Education*, *Health*), and an ontology describing the domain is available to be used as background knowledge. Ontologies are also used as a uniform conceptual representation of peer schemas, and correspondences between these ontologies are set to provide an understanding of their data sources. We consider that correspondences are defined between pairs of semantic neighbor peers, i.e., peers that are semantically related as identified by a clustering process. Another semantic issue we use is context [3]. In query answering, context concerns information and/or parameters that may influence the result given to users in response to their queries.

Besides, we use the Description Logics language ALC (Attribute Language with Complement) [6] in order to formalize ontologies as well as queries. In ALC, the 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. In this light, we state the concept of ontology as a triple $O = \langle C, R, I \rangle$, where C is a set of ALC-concepts, R is a set of ALC-role definitions and I is a set of individuals. Next, we provide an overview of how these semantic elements are used in our approach.

2.1 Using a Domain Ontology to Define Semantic Correspondences

Domain Ontologies (DO) contain concepts and properties belonging to a particular knowledge domain and may be used as background knowledge in some tasks. In our PDMS, we consider DO as reliable references that are available on the internet. Particularly, we use them in order to bridge the conceptual differences or similarities between two ontologies O_1 and O_2 representing the schemas of neighbor peers. In this sense, first concepts and properties from the two peer 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. Figure 1 shows an overview of our approach for specifying the correspondences between peer ontologies. 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 $O_1{:}x \sqsubseteq\!\!\Rightarrow O_2{:}y$.

In order to specify the correspondences, we consider four aspects: (i) the semantic knowledge found in the DO; (ii) whether the peer ontologies' concepts share super-concepts in the DO; (iii) if these super-concepts are different from the root and; (iv) the depth between two 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), which has been designed to formalize multiple ontologies interconnected by mappings [7]. Since our approach is not concerned with proposing new algorithms for DL or DDL, we rely on existing equivalence and subsumption ones [6] as the basis for our definitions.

Definition 1 - Semantic Correspondence. A semantic correspondence is represented by one of the following expressions:

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

2. O_1:x $\sqsubseteq\!\!\Rightarrow$ O_2:y, an *isSubConceptOf* correspondence
3. O_1:x $\sqsupseteq\!\!\Rightarrow$ O_2:y, an *isSuperConceptOf* correspondence
4. O_1:x $\rhd\!\!\Rightarrow$ O_2:y, an *isPartOf* correspondence
5. O_1:x $\lhd\!\!\Rightarrow$ O_2:y, an *isWholeOf* correspondence
6. O_1:x $\approx\!\!\Rightarrow$ O_2:y, an *isCloseTo* correspondence
7. O_1:x $\perp\!\!\Rightarrow$ O_2:y, an *isDisjointWith* correspondence

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

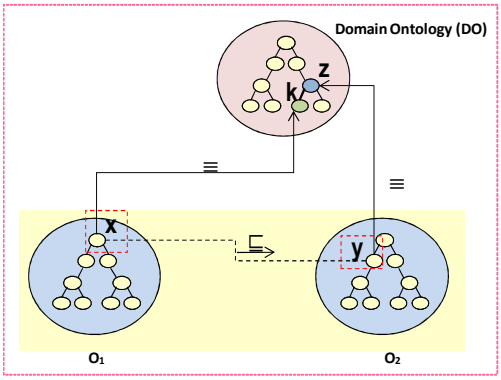

Fig. 1. Specifying Semantic Correspondences between Peer Ontologies

Each correspondence type is defined as follows:

Equivalence: 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 $\equiv\!\!\Rightarrow$ O_2:y and means that both concepts/properties (x and y) describe the same real world concept/property.

Specialization: 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 $\sqsubseteq\!\!\Rightarrow$ O_2:y and means that O_1:x is less general than O_2:y.

Generalization: An element O_1:x *isSuperConceptOf* O_2:y if O_1:x ≡ DO:k and O_2:y ≡ DO:z and DO:k ⊒ DO:z. This correspondence is represented by O_1:x $\sqsupseteq\!\!\Rightarrow$ O_2:y and expresses that O_1:x is more general than O_2:y.

Closeness: 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 ≠ ⊤ and (depth(DO:a, DO:⊤) ⩾ *tresholdRoot*) and ¬(DO:k ⊥ DO:z) and (depth(DO:k,DO:a) ⩽ *thresholdCommonAncestor*) and depth(DO:z,DO:a) ⩽ *thresholdCommonAncestor*). This correspondence is represented by O_1:x $\approx\!\!\Rightarrow$ O_2:y. In this case, considering the DO, the nearest common ancestor of two concepts is used to determine the closeness degree between them. Thus, two concepts are close if they are perceived as belonging to a common relevant meaning, i.e., they are under the same real world concept. The *thresholdRoot* and the *thresholdCommonAncestor* are dependent on the current domain ontology's granularity and size: the former provides a limit for the position of the common ancestor in relation to the root; the latter provides a limit for the position of each matching concept in relation to the common ancestor. Thereby, we state 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 ($\top$); (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. For instance, considering that the concepts *cat* and *lion* (belonging to ontologies O_1 and O_2, respectively) are sub-concepts of a common ancestor *feloidae* (in a given DO), and the closeness conditions are evaluated to true (e.g., the depth between each concept and the common ancestor is lower than the corresponding threshold), we can infer that *cat* and *lion* are close concepts, denoted as O_1:cat $\overset{\cong}{\Rightarrow}$ O_2:lion.

Aggregation – PartOf: An element O_1:x *isPartOf* O_2:y if O_1:x $\equiv$ DO:k and O_2:y $\equiv$ DO:z and DO:k $\triangleright$ DO:z (isPartOf). This correspondence is represented by O_1:x $\overset{\triangleright}{\Rightarrow}$ O_2:y and states that O_1:x is a part or component of O_2:y.

Aggregation – WholeOf: An element O_1:x *isWholeOf* O_2:y if O_1:x $\equiv$ DO:k and O_2:y $\equiv$ DO:z and DO:k $\triangleleft$ DO:z (isWholeOf). This correspondence is represented by O_1:x $\overset{\triangleleft}{\Rightarrow}$ O_2:y and means that, O_1:x is composed by O_2:y. The *WholeOf* correspondence is the inverse of the *PartOf* one, i.e., the composed concept is called "whole" and the components are called "parts". For instance, a *team* is composed by *employees*, denoted as O_1:team $\overset{\triangleleft}{\Rightarrow}$ O_2:employee.

Disjointness: An element O_1:x *isDisjointWith* O_2:y if O_1:x $\equiv$ DO:k and O_2:y $\equiv$ DO:z and DO:k $\perp$ DO:z. This is represented by O_1:x $\overset{\perp}{\Rightarrow}$ O_2:y and states that O_1:x does not overlap with O_2:y.

To clarify matters, in Section 3.2, we provide an example which presents two peer ontologies and a set of identified semantic correspondences between them.

2.2 Using Context in Query Reformulation

In a PDMS, for each submitted query, the whole query execution process instantiation changes completely according to the context of the query, the peers, the schema semantic correspondences and the user [7]. In our approach, we use three types of context: of the user, of the query and of the environment.

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. The 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.
- *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 user wishes an enriched reformulation. These variables help to guide the execution of the query reformulation algorithm (presented in next section). Whenever users want, they can redefine the variables.

The context of the query is obtained by analyzing its semantics and through the query reformulation mode (defined by the user). In the former, the query concepts and

constructors are identified. The latter relates to the way the reformulation algorithm operates: *expanded*, where both exact and enriched reformulations may be provided, or *restricted*, where the priority is to produce an exact reformulation, although if it results empty, then an enriched reformulation may be provided.

The context of the environment is acquired at two different moments: (i) at query session configuration time, when the user defines the variable *Path_Length*, which limits the number of subsequent reformulations in the set of neighbor peers; and (ii) at query submission time, when the system identifies in which peer the query has been submitted and also establishes the context of the submission peer neighbors (e.g., peer's availability). Based on this set of contextual elements, the system defines where to route and to reformulate the queries.

Most contextual information 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 to operate. Others are intended to be inferred on the fly according to the environment's conditions. We have represented contextual information by a context ontology [8]. However, in this paper, we only deal with the context acquired from the user preferences.

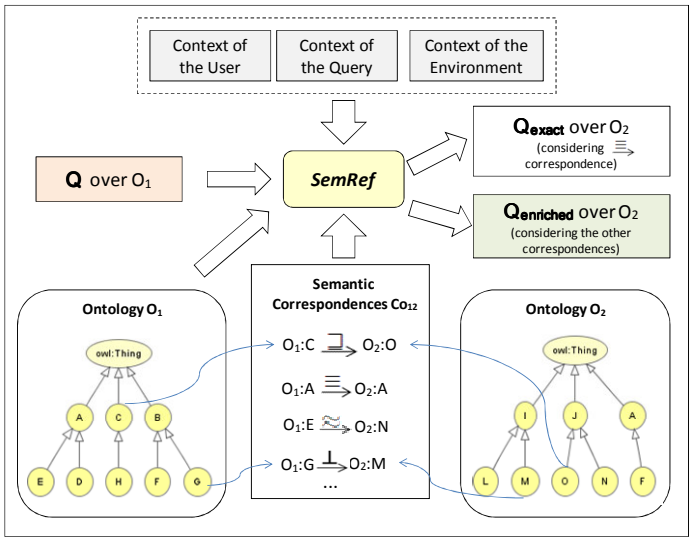

Fig. 2. Semantic-based Query Reformulation Approach

2.3 General Principle of Applying Semantics to Query Reformulation

The principle of our approach is to enhance query reformulation by using semantic correspondences between schema ontologies and contextual information in such a way that we can provide users with a set of expanded answers. These answers are consistent with what the user has defined as relevant through his/her preferences, according to the current status of the environment (i.e., the context of the user, of the

query and of the environment). Exact and enriched query reformulations will be produced as a means to obtain these answers. Figure 2 shows an overview of our approach, considering two neighboring peers P_1 and P_2, their respective ontologies O_1 and O_2, and a set of correspondences between them $\{Co_{12}\}$.

3 The *SemRef* Approach

In this section, we present the Semantic Query Reformulation algorithm, named *SemRef*. Furthermore, we provide some examples showing how *SemRef* works.

3.1 The *SemRef* Algorithm

Our query reformulation approach has been encoded in ALC-DL [6]. Thus, a query Q over a peer ontology O_1 is a concept expression, $Q = C$, where C is an ALC concept. An ALC concept may be an atomic concept or a complex concept including roles, quantifiers, conjunctions or disjunctions. In our work, we consider that a query Q is a formula consisting of a disjunction of queries which are themselves conjunctions of ALC concepts $C_1, \ldots, C_n$ where $n \geq 1$, as follows.

Definition 2 - Query. A query Q expressed over P_1's ontology, has the following form: $Q = Q_1 \sqcup Q_2 \sqcup \ldots \sqcup Q_m$, where $Q_i = C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$, and where each C_j is an atomic concept, a negated atomic concept of a quantified atomic concept (C_j, $\neg C_j$, $\forall R.C_j$ or $\exists R.C_j$).

Supposing a peer ontology concerning an academic research center, some query examples are:

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

Q_2 = [Teacher $\sqcap$ Researcher] $\sqcup$ [Student $\sqcap$ Researcher], which asks for people who are teachers and researchers or students that are also researchers.

We argue that when posing a query, users must be aware that not only exact answers, but also those that meet or complement their initial intention, can be relevant for them. The algorithm verifies users' preferences and if it is not possible to produce an exact reformulation to a given query or if users' preferences enable enriching reformulation, it produces an enriched one, providing users with answers at different levels of closeness. 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. Query reformulations are defined as follows:

Definition 3 - Exact Reformulation. A reformulation Q' of a query Q is said to be exact (denoted as *Qexact*) if each concept (or property) C' of Q' is related to a concept (or property) C of Q by a Co correspondence, where $Co \in \{\equiv\Rightarrow\}$ (equivalence).

Definition 4 - Enriched Reformulation. A reformulation Q' of a query Q is said to be enriched (*Qenriched*) if each concept (or property) C' of Q' is related to a concept (or property) C of Q by a Co correspondence, where $Co \in \{\sqsubseteq\Rightarrow, \sqsupseteq\Rightarrow, \approx\Rightarrow, \rhd\Rightarrow, \lhd\Rightarrow, \perp\Rightarrow\}$.

The *SemRef* algorithm always tries to produce the set of exact reformulations, although sometimes the produced set is empty. Enriched reformulations will be produced in two situations: (i) if the user requires them through the set of enriching variables (*approximate*, *specialize*, *generalize* and *compose*) and *expanded* query execution mode or; if the user has set the enriching variables, the execution mode was defined as restricted, but the exact reformulation resulted empty. Such combination of possibilities is described in Table 1. In this light, the *SemRef* algorithm receives as input a query Q, submitted in a peer P_1, the target peer P_2, the set of semantic correspondences between them and the context that has been set by the user (enriching variables and *mode* values). As output, it produces one or two reformulated queries (*Qexact* and/or *Qenriched*). A high level view of *SemRef* is sketched in Figure 3. The complete *SemRef* algorithm is detailed in Figure 4.

Table 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

***SemRef*(Q, P_1, P_2, Co[O_1,O_2], MODE, REF_VAR, Q_{exact}, $Q_{enriched}$)**
Input: Q, P_1, P_2, Co[O_1,O_2], MODE, REF_VAR
Output: Q_{exact}, $Q_{enriched}$

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 $Q_{k\_exact} \neq \emptyset$)
8. Then
9. Build final exact reformulation Q_{exact} of Q
10. Else $Q_{exact} \leftarrow \emptyset$
11. If ((MODE is expanded) or (MODE is restricted and Q_{exact} is empty)) and
12. (at least one of $Q_{k\_enriched} \neq \emptyset$)
13. Then
14. Build final enriched reformulation $Q_{enriched}$ of Q
15. Else $Q_{enriched} \leftarrow \emptyset$
16. End *SemRef*;

Fig. 3. High Level View of *SemRef* Algorithm

```
SemRef (Q, P1, P2, Co[P1,P2], MODE, REF_VAR, Qexact, Qenriched)
For each Qk in Q      /* for each conjunctive query in Q */
    B ← TRUE    /* used to stop the search if some concept has no correspondent one in Pj */
    While (there is still a concept Cj in Qk to process) and (B=TRUE)
        S1Cj ← ∅  /* set of concepts that are equivalent to Cj */
        S2Cj ← ∅  /* concepts related to Cj by other correspondence except disjointness*/
        Neg_S2cj ← ∅  /* set of concepts related to Cj by disjointness correspondence */
        For each isEquivalentTo assertion between Cj and a concept C'
            Add C' to S1Cj
        End For;  /* End of the loop related to the assertions equal to ≡> */
        For each other kind of assertion involving Cj
            If (MODE is expanded) or (MODE is restricted and S1Cj is empty) Then
                If APPROXIMATE = TRUE Then
                    If there is a concept C' in P' such that C' ≈> Cj Then
                        Add C' to S2Cj
                If SPECIALIZE = TRUE Then
                    If there is a concept C' in P' such that C' ⊏> Cj Then
                        Add C' to S2Cj
                If GENERALIZE = TRUE Then
                    If there is a concept C' in P' such that C' ⊐> Cj Then
                        Add C' to S2Cj
                If COMPOSE = TRUE Then
                    If there is a concept C' in P' such that C' ▷> Cj or C' ◁> Cj Then
                        Add C' to S2Cj
                If Cj is negated Then
                    If there is a concept C' in P' such that C' ⊥> Cj Then
                        Add C' to Neg_S2Cj
                        BNeg ← TRUE
        End For;   /* End of the loop related to the assertions different from ≡> */
        If (S1Cj = ∅ and S2Cj  = ∅ and Neg_S2C2 = ∅) Then B ← FALSE
    End While;   /* End of the loop processing concepts */
    B1 ← TRUE;  If (one of S1Cj = ∅) Then B1 ← FALSE    /* the conjunction fails */
    B2 ← TRUE;  If (one of S2Cj = ∅) Then B2 ← FALSE    /* the conjunction fails */
    If B1 = TRUE Then Qk_exact ← Build_Exact_Reformulation (Qk, S1C1, S1C2, …, S1Cp)
                 Else Qk_exact ← ∅
    If B2 = TRUE or BNeg = TRUE  Then Qk_enriched ← Build_Enriched_Reformulation(Qk,
                                       S2C1, … S2Cp,Neg_S2C1, … Neg_S2Cp)
                                  Else Qk_enriched ← ∅
End For;   /* End of the loop processing the conjunctive queries Qk */
If (at least one of Qk_exact ≠ ∅)    /* at least one of Qk's exact is not empty  */
   Then Qexact ← Build_Final_Exact_Reformulation (Q, Q1_exact, …, Qm_exact)
   Else Qexact  ← ∅
If ((MODE is expanded) or (MODE is restricted and Qexact is empty)) and
    (at least one of Qk_enriched ≠ ∅)
    Then Qenriched  ← Build_Final_Enriched_Reformulation (Q, Q1_enriched,…,
                        Qm_ enriched)
   Else Qenriched  ← ∅
End_SemRef;
```

Fig. 4. The *SemRef* Algorithm

In order to obtain the reformulations, the algorithm performs the following tasks:

I. It receives query Q (a disjunction of conjunctions of ALC concepts). For each conjunctive query Q_k in Q, while there are concepts C_j in Q_k to process, it adds the corresponding concepts to one of three 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 kinds 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.
   - $Neg\_S_2C_j$: 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 disjointness 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 existing concepts in the conjunction had corresponding ones). If so, it builds the exact reformulation for the current conjunctive query Q_k.

III. 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 *Qexact*, as the disjunction of the resulting exact conjunctions and the final *Qenriched* as the disjunction of the resulting enriched conjunctions.

As a way to present *SemRef*'s main steps execution, we provide some examples in next section.

3.2 *SemRef* in Practice

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 work (e.g., research). Each peer is described by one ontology – O_1 (*Semiport.owl*) and O_2 (*UnivBench.owl*). Furthermore, we have considered as background knowledge a public DO named *UnivCSCMO.owl*[1].

In order to identify the semantic correspondences between O_1 and O_2, the set of rules described in Section 2.1 was applied [9]. As a result, the set of semantic correspondences between O_1 and O_2 was identified. Since the correspondences are unidirectional, we present examples of this set concerning the concept FullProfessor (from O_1) with some related concepts in O_2: O_1:FullProfessor $\equiv\!\!>$ O_2:FullProfessor, O_1:FullProfessor $\sqsubset\!\!>$ O_2:Professor, O_1:FullProfessor $\approx\!\!>$ O_2:VisitingProfessor, O_1:FullProfessor $\perp\!\!>$ O_2:AssociateProfessor, O_1:FullProfessor $\rhd\!\!>$ O_2:Course. In this illustrative set, we can see the equivalence correspondence between FullProfessor in

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

O_1 and O_2. This is the most commonly identified correspondence type in traditional query reformulation approaches. On the other hand, by using the semantics underlying the DO, we can identify other unusual correspondences. In this fragment, FullProfessor has been identified as: (i) *sub-concept* of Professor; (ii) *close* to VisitingProfessor; (ii) *disjoint* with AssociateProfessor; and; (iii) *part* of Course.

We present the *SemRef* main steps in practice through the query Q = FullProfessor, submitted in P_1. *SemRef* starts by initializing the sets S_1C_1, S_2C_1 and Neg_S_2C_1. The first set receives the concepts that are equivalent to FullProfessor, i.e., S_1C_1 = {FullProfessor}. The second set receives the concepts resulting from the other correspondences (except disjointness), i.e., S_2C_1 = {VisitingProfessor, Professor, Course}, considering that the user has set all four enriching variables to TRUE and the reformulation mode to EXPANDED. 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 it, the algorithm verifies that both sets (S_1C_1 and S_2C_1) are not empty and consequently builds both exact and enriched reformulations. The final exact reformulation is Qexact = [FullProfessor]. The enriched one is Qenriched = [VisitingProfessor ⊔ Professor ⊔ Course].

Another example regards the submitted query Q = ¬UndergraduateStudent (in P_1). Suppose that the user has set the *specialize* variable to TRUE, and the *restricted* option for the reformulation mode. Also, assume that there is no equivalent concept of UndergraduateStudent in P_2, what implies in S_1C_1 = { }. Since query execution mode was defined as restricted, but S_1C_1 was empty, enrichment is considered by *SemRef*. Thus, S_2C_1 set receives {Monitor}, according to the specialization correspondence. Because there is a negation over the concept UndergraduateStudent, the third set Neg_S_2C_1 is set to {Worker, GraduateStudent}, according to disjointness correspondences. Thereby, although the algorithm does not build an exact reformulation, it produces an enriched one, negating over the concept *Monitor*, and providing a union of such negation with the concepts *Worker* and *GraduateStudent* (from Neg_S_2C_1 set). The final exact reformulation is Qexact = { }, but the final enriched one is Qenriched = [¬Monitor ⊔ Worker ⊔ GraduateStudent].

4 Experiments and Results

We have developed the *SemRef* approach within a query submission module (implemented in Java) for our PDMS. Figure 5 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[2] or ALC-DL. In this paper, we present our experiments using queries expressed in DL.

Considering the peers presented in the previous example, we identified the set of correspondences between their ontologies which was stored in a RDF file. A fragment of such file is shown in Figure 6, where the concept undergraduatestudent is stated as *disjointwith* worker, *partof* course and *subconceptof* student. Then, we ran

[2] http://www.w3.org/TR/rdf-sparql-query/

several queries from P_1 to P_2 and vice-versa. For each query, we evaluated its reformulation, considering semantics and not considering semantics, i.e., with different enriching variables and restricted/expanded mode and without enriching variables and restricted/expanded mode. As a result, 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. An example of one reformulated query where *Qexact* is empty is shown in Figure 7.

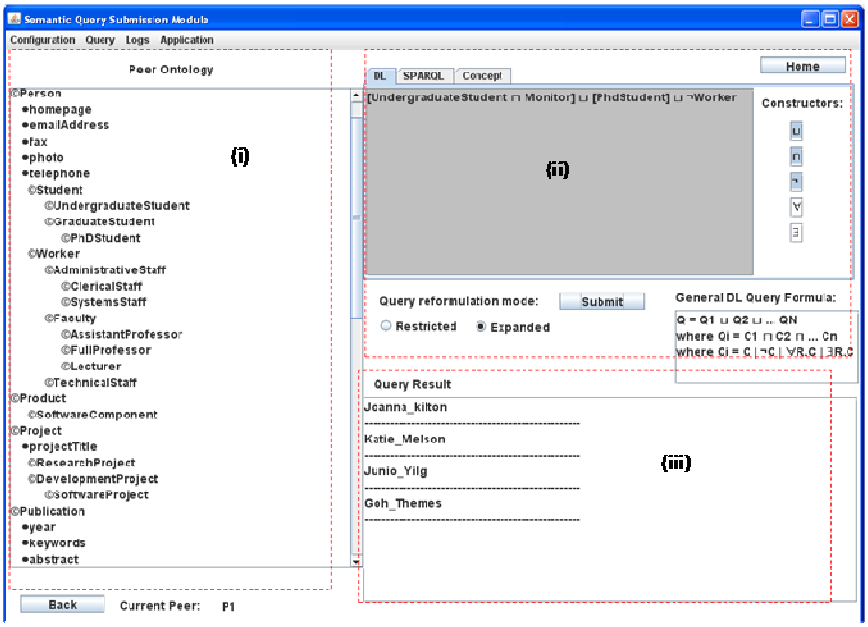

Fig. 5. Query Interface

```
<rdf:Description rdf:about="http://swrc.ontoware.org/ontology/portal#UndergraduateStudent">
 <j.0:isDisjointWith>http://www.lehigh.edu/~zhp2/univbench.owl#Worker</j.0:isDisjointWith>
 <j.0:isPartOf>http://www.lehigh.edu/~zhp2/univbench.owl#Course</j.0:isPartOf>
 <j.0:isSubConceptOf>http://www.lehigh.edu/~zhp2/univbench.owl#Student</j.0:isSubConceptOf>
```

Fig. 6. Some Correspondences between P_1 and P_2

Even enabling only one of the enriching variables has shown to entail 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 related answers. Figure 8 presents one query example, considering only *approximate* = *TRUE*, where *SemRef* produces both exact and enriched reformulations (in this case, only

GraduateStudent has an equivalent concept in P_2). 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 (GraduateStudent *isdisjointwith* UndergraduateStudent).

```
Query Mode: Expanded
Using Reformulation Variables: Yes
Selected Variables: Approximate, Generalize, Compose, Specialize
Original Query (Source Peer): PhDStudent
Exact Query (Target Peer):
Enriched Query (Target Peer): [[MasterStudent ⊔ GraduateStudent]]
```

Fig. 7. Empty Exact Reformulation and Not Empty Enriched Reformulation

```
Query Mode: Expanded
Using Reformulation Variables: Yes
Selected Variables: Approximate
Original Query (Source Peer): ¬GraduateStudent ⊔ Lecturer ⊔ TechnicalStaff
Exact Query (Target Peer): [[¬GraduateStudent]]
Enriched Query (Target Peer): [[UndergraduateStudent]] ⊔ [[PostDoc ⊔ Professor]] ⊔ [[Assistant ⊔ Faculty]]
```

Fig. 8. Both Exact and Enriched Reformulations

In summary, the experimental results support 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. Since we have conducted our experiments considering only two peers (a source and a target), we have not used the variable *Path_length*. Besides, 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.

5 Related Work

Query reformulation techniques have been tackled in diverse environments. The works of Necib and Freytag [10] and Kostadinov [11] use semantic knowledge to enrich queries. The former uses knowledge provided by a DO and correspondences to a single database. The latter uses knowledge from user profiles in a mediator-based system. Regarding PDMS, Piazza is one that performs query reformulation by means of equivalence and inclusion mappings [1]. Other approaches have some peculiarities: Xiao and Cruz [12] consider integrity constraints specified on data sources; Calvanese and his group [13] present the "What-To-Ask" problem (WTA) and compute its solution through three kinds of mappings - subsumption between classes, participation of classes in roles and mandatory participation of classes in roles; the work of Adjiman et al. [2] shows query reformulation in SomeRDFS reduced to the problem

of consequence finding over logical propositional theories and, finally, Stuckenschmidt et al. [14'] provide query reformulation using concept approximation and query relaxation.

Comparing these works with ours, correspondences in most of them are restricted to equivalence and subsumption (SomeRDFS also considers disjunction). We go one step further as we also use other kinds of semantic reformulation rules (e.g., closeness and aggregation) which are obtained from the set of semantic correspondences between the peers. 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 3, when users enable approximation, closeness correspondences may 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, the mentioned works do not deal with context. Differently, our work produces reformulated queries considering the context of the user (preferences), of the query (mode) and of the environment (relevant peers). Moreover, our work prioritizes the generation of exact reformulations, but, depending on the context, it also generates an enriched version, which may avoid empty reformulations, providing a larger set of reformulated queries, and, consequently, non-empty expanded answers to users.

6 Conclusions and Further Work

In environments which are highly dynamic, the semantics surrounding queries are rather important to produce results with relevance according to users' needs and environment's capabilities. This work has presented a semantic-based query reformulation approach instantiated in a PDMS that brings together both query enrichment and query reformulation. What differentiates *SemRef* approach from other ones is that it goes beyond traditional correspondences usage, by means of some extended semantic correspondences (e.g., closeness) identification as well as by considering context. In this sense, *SemRef* prioritizes the generation of exact reformulations but, depending on the current context (e.g., reformulation mode), it also generates an enriched version. As a result, users benefit from such enrichment, getting additional and/or close answers, if so they choose.

Experiments carried out have shown that, if we consider only equivalent correspondences (without semantics), some reformulations result empty. Considering semantics, enriched reformulations are generated, providing additional expanded reformulations and query answers.

Currently, we are developing rules to allow reasoning over the contextual information already instantiated in a specific context ontology [8]. This reasoning might improve the query reformulation and routing processes. As further work, we will instantiate additional query reformulation scenarios which may allow us to work with other different contextual settings and with larger datasets.

References

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

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

[3] Dey, A.: Understanding and Using Context. Personal and Ubiquitous Computing Journal, Volume 5 (1), pp. 4-7. (2001).

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

[5] Levy A., Rajaraman A., and Ordille J.: Querying heterogeneous information sources using source descriptions. In Proceedings of the International Conference on Very Large Databases (VLDB), pages 251{262. (1996).

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

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

[8] Souza, D., Belian, R., Salgado, A. C., Tedesco, P.: 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, Auckland, New Zealand. (2008).

[9] Pires, C. E. S., Souza, D., Pachêco, T., and Salgado, A. C. (2009a) "A Semantic-based Ontology Matching Process for PDMS", Submitted to 2nd International Conference on Data Management in Grid and P2P Systems, Linz, Austria.

[10] Necib, C. B. and Freytag, J.: Query Processing Using Ontologies. CAiSE 2005: 167-186. (2005).

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

[12] Xiao, H., Cruz, I.: Ontology-based Query Rewriting in Peer-to-Peer Networks. In Proc. of the 2nd International Conference on Knowledge Engineering and Decision Support. (2006).

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

[14] Stuckenschmidt H., Giunchiglia F., and van Harmelen F.: Query processing in ontology-based peer-to-peer systems. In V. Tamma, S. Crane¯eld, T. Finin, and S. Willmott, editors, Ontologies for Agents: Theory and Experiences. Birkhuser. (2005).