What to Ask to a Peer: Ontology-based Query Reformulation

Diego Calvanese
Faculty of Computer Science
Free University of Bolzano/Bozen
Piazza Domenicani 3, I-39100 Bolzano, Italy
calvanese@inf.unibz.it

Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it

Abstract

In the recent years, the issue of cooperation, integration, and coordination between information peers in a networked environment has been addressed in different contexts, including data integration, the Semantic Web, Peer-to-Peer and Grid computing, service-oriented computing and distributed agent systems. One of the main problems that arises in such systems is how to exploit the mappings between peers in order to answer queries posed to one peer. The goal of this paper is to present some basic, fundamental results on this problem. In particular, we focus on a simplified setting based on just two interoperating peers and we investigate how to solve the so-called "What-To-Ask" 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. We show that a solution to this problem exists in the case of peers based on a basic ontology language and provide an algorithm to compute it. We also show that, by slightly enriching the ontology language, the problem may become unsolvable.

Introduction

In the recent years, the issue of cooperation, integration, and coordination between information nodes in a networked environment has been addressed in different contexts, including data integration (Halevy 2000; Lenzerini 2002), the Semantic Web (Heflin & Hendler 2001), Peer-to-Peer and Grid computing (Bernstein *et al.* 2002; Halevy *et al.* 2003), service oriented computing and distributed agent systems (Papazoglou, Kramer, & Yang 2003; Hull *et al.* 2003).

Put in an abstract way, all these systems are characterized by an architecture constituted by various autonomous nodes (called sites, sources, agents, or, as we call them here, peers) which hold information, and which are linked to other nodes by means of mappings. Two basic problems arising in this architecture are: how to discover, express, and compose the mappings between peers (Halevy *et al.* 2003; Bernstein *et al.* 2002; Madhavan & Halevy 2003; Fagin *et al.* 2004), and how to exploit the mappings in order to answer queries posed to one peer (Halevy 2001; Lenzerini 2002; Mattos 2003). The latter is the problem studied in this paper.

Although several interesting results have been reported in each of the above contexts, we argue that a deep understanding of the problem of answering queries in a networked environment is still lacking, in particular when the information in each peer is modelled in terms of a knowledge base.

The goal of this paper is to present some basic, fundamental results on this problem. We consider a simplified setting where the whole system is constituted by only two peers, called local and remote, respectively. Suitable mappings relate information in the remote peer to the information in the local peer. We assume that the queries to be answered are posed to the local peer, and that each of the two peers provides the service of answering queries expressed over its underlying knowledge base, with these two services being the only basic services that we can rely upon. Thus, the formal problem we address in the paper, called the "What-To-Ask" problem, is to find a way to answer queries posed to the local peer by relying only on the two query answering services available at the peers.

Example 1 Consider, for example, a music sharing system, and assume that the peer SongUniverse exports both actual songs and knowledge about various types of music, e.g., the fact that live rock songs are live performance songs. Assume now that SongUniverse stores live songs and also knows that other live rock songs can be retrieved from the remote peer RockPlanet. Now, when Carol visits SongUniverse to get live songs of U.K. artists, what this peer can do is: (*i*) directly provide her with the live songs of U.K. artists that it stores locally, and (*ii*) deduce that also live rock songs suit Carol's needs, and reformulate Carol's request by asking RockPlanet about live rock songs of U.K. artists. ■

In this paper, we study the What-To-Ask problem in a first-order logic (FOL) context. Specifically, we present the following contributions.

1. We formalize the above mentioned architecture, we define its semantics, and we give a precise characterization of the semantics of query answering. We provide the formal definition of the "What-To-Ask" problem, taking into account both the semantics of query answering and the fact that, when answering a query posed to the local peer, we can only rely on the two query answering services available at the peers.

2. We specialize the general framework to the case where a basic ontology language is used to express the knowledge bases of the two peers. We show that in this case there is an algorithm that allows us to solve any instance of "What-To-Ask", i.e., that allows to compute what we should ask to the remote peer in order to answer a query posed to the local peer.
3. We show that, if we slightly enrich the expressive power of the ontology language, there are instances of the "What-To-Ask" problem that do not admit solutions.

The rest of the paper is organized as follows. First, we present a general formal framework for knowledge-based peer interoperation and provide the definition of the What-To-Ask problem. Then, we consider a particular instantiation of the general framework where the knowledge base is expressed in a basic ontology language allowing for the definition of classes, roles, and subsumption between classes, and provide for such a case a solution to the What-To-Ask problem. Next, we show that by enriching the ontology language with subsumption between roles, the What-To-Ask problem may admit no solutions. Finally, we discuss some related work, and conclude the paper.

The Framework

In this section we set up a formal framework for *knowledge-based peer* interoperation.

Each knowledge-based peer contains a knowledge base K that it can use to make logical inferences. The peer exports a suitable fragment V of K to the agents willing to use the peer, here called *clients*. Clients can *ask* to the peer only queries that are *accepted by* the peer, i.e., expressed in some given query language over the exported fragment V of the knowledge base K that the peer supports. The peer answers such queries by exploiting inference from its knowledge base K.

Apart from using its knowledge base K, each peer can be connected with other knowledge-based peers to which it can *ask* queries that can be accepted by them. Suitable *mappings* between the peers give the means to interpret the answers to queries posed to the remote peer. Basically, mappings establish the relationship between given queries posed over the peer knowledge base K and queries accepted by a remote peer.

In this paper we focus on a system made up by two interoperating knowledge-based peers. One of them, called local peer, is the one with whom the client interacts, i.e., asks queries. The other peer will be referred to as the remote peer, and the knowledge contained in it will be exploited indirectly by the local peer through the mappings to enhance its answer to the client. We further assume that, while the local peer exploits the remote peer through the mappings, the remote peer does not use in any way the knowledge of the local one. This very simple setting will already allow us to uncover various subtleties of an interoperating knowledge-based peer system.

Next, we formalize such a framework. We assume that the domain of interpretation is a fixed denumerable set Δ of elements and that every such element is denoted uniquely by a constant, called its *standard name* (Levesque & Lakemeyer 2001). We denote by Γ the set of standard names and we assume that Γ is part of the alphabet of the local knowledge base in each peer. With this assumption in place, we turn our attention to the notion of knowledge-based peer.

Definition 2 A *knowledge-based peer* is a tuple of the form $P = \langle K, V, M\rangle$ where:

- K is a knowledge base written in some subset of first-order logic (FOL) on the alphabet formed by the standard names as constants, and a set of relation names (we do not consider functions in this paper);
- V is the exported fragment of K (i.e., a knowledge base formed as a subset of K);
- M is a set of mapping assertions whose form will be shown below. ■

Clients pose their queries over the exported fragment V of a peer P. Queries that we consider in our framework are expressed in a language which amounts to a fragment of FOL. We remind the reader that a *FOL query* is an open formula of the form

$$\{x_1, \ldots, x_n \mid \phi(x_1, \ldots, x_n)\}$$

where $x_1, \ldots, x_n$ are the free variables of ϕ, and n is the *arity* of the query. In general, a peer P supports queries expressed in some subset of FOL, and clients can ask queries to P as long as such queries are accepted by P. Formally, this is captured as follows.

Definition 3 Given a knowledge-based peer $P = \langle K, V, M\rangle$, a query q is *accepted by* P if q is a query over the alphabet of V expressed in the subset of FOL (possibly with equalities) that is supported by P. ■

A knowledge-based peer system is formed by several peers sharing the domain of interpretation and the set of standard names. Here we concentrate on knowledge-based systems of a very specific form. They consist of only two peers, 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 $P_r = \langle K_r, V_r, \emptyset\rangle$, called *remote peer*. Observe that the remote peer does not contain mapping assertions at all.

The mapping M_ℓ in the local peer is constituted by a finite set of *assertions* of the form

$$q_r \rightsquigarrow q_\ell,$$

where q_r and q_ℓ are two queries of the same arity, called local and remote query, respectively. The local query q_ℓ is expressed in some FOL query language over K_ℓ, while the remote query q_r must be a query accepted by P_r.

A mapping assertion $q_r \rightsquigarrow q_\ell$ has an immediate interpretation in FOL: it states that

$$\forall x_1, \ldots, x_n . \phi_r(x_1, \ldots, x_n) \supset \phi_\ell(x_1, \ldots, x_n),$$

where ϕ_ℓ and ϕ_r are the open formulas constituting the queries q_ℓ and q_r, respectively.[1]

[1]This form of mappings is often refereed to as GLAV sound mappings in data integration (Lenzerini 2002).

Let us now turn our attention to the notion of *answer* to a query posed to our knowledge-based peer system, i.e., a query posed to the local peer P_ℓ of the system. First of all, we recall the notion of certain answers to a query posed over a first-order theory.

Definition 4 Given a FOL query q of arity n over a FOL theory $\mathcal{T}$, the *certain answers* $cert(q,\mathcal{T})$ to q over $\mathcal{T}$ is the set of tuples of constants of Γ such that:

$$cert(q,\mathcal{T}) = \{(c_1,\ldots,c_n) \mid (c_1,\ldots,c_n) \in q^{\mathcal{I}} \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \mathcal{T}\},$$

where $q^{\mathcal{I}}$ denotes the result of evaluating q in the interpretation $\mathcal{I}$. (Recall that the interpretation of constants, i.e., standard names, is the same in every interpretation.) ∎

In our framework, we assume that each peer $P = \langle K, V, M\rangle$ is only able to provide the certain answers $cert(q,K)$, inferable from its knowledge base K, to queries accepted by P itself.

Now, ideally, we would like, given a client's query q accepted by the local peer P_ℓ, to return all certain answers that are inferable from all the knowledge in the system. That is, we aim at computing $cert(q, K_\ell \cup K_r \cup M_\ell)$. However, to do so we can rely only on the query answering services available at the peers. In other words, we need to exploit the kind of certain answers that peers can actually compute, that is certain answers with respect to their knowledge base. First, we can directly use the certain answers $cert(q,K_\ell)$ provided by the local peer P_ℓ. Then, to use the certain answers provided by the remote peer P_r, we need to reformulate in some way the query q into a finite set $\{q_r^1,\ldots,q_r^n\}$ of queries, each accepted by the remote peer P_r, and require that the union of $cert(q,K_\ell)$ with the certain answers computed by P_r for each q_r^i gives us $cert(q, K_\ell \cup K_r \cup M_\ell)$. Formally, we define the *What-To-Ask* problem as follows.

Definition 5 Consider a knowledge-based peer system consisting of a local peer $P_\ell = \langle K_\ell, V_\ell, M_\ell\rangle$ and a remote peer $P_r = \langle K_r, V_r, \emptyset\rangle$, and let q be a client's query accepted by P_ℓ. The *What-To-Ask* problem, $WTA(q, P_\ell, P_r)$, is defined as follows: *Given as input P_ℓ and q, find a finite set $\{q_r^1,\ldots,q_r^n\}$ of queries, each accepted by the remote peer P_r, such that*[2]

$$cert(q, K_\ell \cup K_r \cup M_\ell) = cert(q, K_\ell) \cup \bigcup_{i=1}^{n} cert(q_r^i, K_r).$$

∎

Notice that, in general, several solutions to the What-To-Ask problem may exist, i.e., there may be several sets of queries for which the conditions described above hold. However, it is easy to see that, according to Definition 5, all solutions are equivalent from a semantic point of view, i.e., each of them allows us to obtain all certain answers inferable from the knowledge managed by the peer system. Nonetheless, syntactic differences might exist between different solutions that could lead one to prefer one solution to another, e.g., if the set of queries in the former is contained in the set of queries in the latter. However, we focus here on solving the What-To-Ask problem (i.e., finding *any* solution that satisfies Definition 5) in the specific setting described in the next section, whereas the problem of characterizing which solution is "better", and of finding the "best" solutions is outside the scope of this paper.

[2]Note that in finding q_r^i we can exploit neither K_r nor V_r, since P_r is not an input of the problem, but is only used as a parameter to the problem necessary to formulate the notion of "*accepted by* P_r".

The What-To-Ask Problem in an Ontology-Based Framework

We now consider a particular instantiation of the formal framework described above, and provide for a such case a solution to the What-To-Ask problem.

Specialized Framework

To specialize the formal framework for knowledge-based peers described above, we consider specific choices for the language in which a peer knowledge base is expressed, for the language of queries accepted by peers, and for the query language to be used in the local query of mapping assertions.

We concentrate first on the language in which to express the peer knowledge base. The language we use, called L_K^O in the following, is a subset of FOL that captures the fundamental features of frame-based knowledge representation formalisms and of ontology languages for the Semantic Web. The alphabet of L_K^O consists of constants from Γ, and of unary and binary predicates, called *classes* and *roles* respectively. Classes denote sets of objects, while roles denote binary relationships between classes. The language L_K^O consists of two components, to represent respectively intensional and extensional knowledge in the peer knowledge base K.

The intensional component of L_K^O allows for capturing typical ontology constructs, namely typing of roles, mandatory participation to roles for the objects in a class, functionality of direct and inverse roles, and subsumption between classes. To keep the presentation simple, we represent the constructs of L_K^O using a graphical notation, and specify their semantics in FOL. Specifically, the intensional component of K is a directed graph whose nodes are classes and whose edges represent either roles or subsumption relationships.

Classes of K, in the following denoted by the letter C, possibly with subscripts, are represented by means of a rectangle containing the name of the class. Roles of K, in the following denoted by the letter R, possibly with subscripts, are represented by means of a (thin) arrow, labeled with the name of the role, connecting two classes, called respectively the first and second component of the role. Each role is also labeled with participation and functionality constraints for both components, as depicted in Figure 1, where

- m_1 and m_2 may be either
  - 0, which means there is no constraint on the participation, or

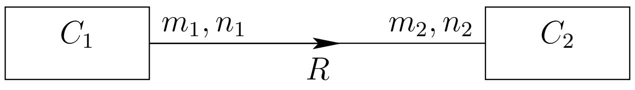


Figure 1: A role in the intensional component of a peer knowledge base

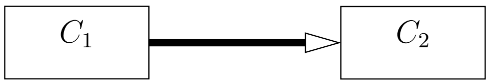


Figure 2: Subsumption in the intensional component of a peer knowledge base

  - 1, which means mandatory participation;
- n_1 and n_2 may be either
  - 1, which means functionality, or
  - ∞, which means there is no constraint.

In the graphical representation we omit constraints of the form $(0, \infty)$.

The FOL assertions that specify the semantics of the fragment of knowledge base shown in Figure 1 are the following:

- assertions specifying the typing of the two components of the role:
  - an assertion specifying the typing of the first component of the role:
  $$\forall x, y. R(x, y) \supset C_1(x);$$
  - an assertion specifying the typing of the second component of the role:
  $$\forall x, y. R(x, y) \supset C_2(y);$$
- possibly, assertions specifying the mandatory participation to the role:
  - if $m_1 = 1$, then:
  $$\forall x. C_1(x) \supset \exists y. R(x, y);$$
  - if $m_2 = 1$, then:
  $$\forall y. C_2(y) \supset \exists x. R(x, y);$$
- possibly, assertions specifying functionality of the role:
  - if $n_1 = 1$, then:
  $$\forall x, y_1, y_2. R(x, y_1) \wedge R(x, y_2) \supset y_1 = y_2;$$
  - if $n_2 = 1$, then:
  $$\forall x_1, x_2, y. R(x_1, y) \wedge R(x_2, y) \supset x_1 = x_2.$$

L_K^O is equipped with a subsumption relationship between classes, denoted by a thick hollow arrow from the subsumed class to the subsuming class, as depicted in Figure 2. The corresponding FOL formula specifying the semantics is:
$$\forall x. C_1(x) \supset C_2(x).$$

The extensional component of L_K^O contains facts and existential formulas, possibly involving constants of Γ. Specifically, each such assertion has one of the forms

$$\begin{array}{ll} C(a), & \exists x. C(x), \\ R(a_1, a_2), & \exists x_1, x_2. R(x_1, x_2), \\ \exists x. R(a, x), & \exists x. R(x, a), \end{array}$$

where C and R are respectively a class and a role of K, and a, a_1, and a_2 are constants of Γ.

To summarize, a peer knowledge base K expressed in L_K^O consists of a set of FOL assertions representing intensional information, and of a set of facts and existential formulas, representing extensional information.[3] In the following, we view the whole knowledge base of a peer as a set of FOL assertions (those that specify the semantics of the intensional component, together with the extensional component) expressed over an alphabet of classes and roles.

Let us now turn the attention to the language of queries accepted by a peer. We adopt the language of conjunctive queries (Abiteboul, Hull, & Vianu 1995), which offers a reasonable trade-off between expressive power and complexity of query processing, and is thus adopted in most data integration proposals.

Definition 6 A *conjunctive query* (CQ) q is a formula of the form

$$\{z_1, \ldots, z_n \mid \exists y_1, \ldots, y_m. \phi(z_1, \ldots, z_n, y_1, \ldots, y_m)\},$$

where $z_1, \ldots, z_n$ are (not necessarily pairwise distinct) variables or constants of Γ, and $\phi(z_1, \ldots, z_n, y_1, \ldots, y_m)$ is a conjunction of atoms, possibly containing constants of Γ, whose predicates are classes or roles, and whose free variables are the variables in $z_1, \ldots, z_n, y_1, \ldots, y_m$. We call $(z_1, \ldots, z_n)$ the *head* of q. ■

Note that a CQ written in the form above corresponds to a FOL query

$$\{x_1, \ldots, x_n \mid \exists y_1, \ldots, y_m. \phi(x_1, \ldots, x_n, y_1, \ldots, y_m) \wedge \mathit{eqs}\},$$

where $x_1, \ldots, x_n$ are pairwise distinct variables, and *eqs* is a conjunction of equalities, with one equality $x_i = c$ whenever z_i is a constant c, and one equality $x_i = x_j$, whenever z_i is the same variable as z_j.

With regard to the specification of mappings, we need only to define which is the query language used for expressing local queries in the mapping assertions. Indeed, remote queries are queries accepted by the remote peer $P_r = \langle K_r, V_r, \emptyset \rangle$, and thus are CQs over the exported fragment V_r. We consider here mapping assertions that allow for establishing a basic form of correspondence between knowledge in different peers, namely to map a single element of

[3]Note that L_K^O can be seen as a specific description logic (Baader *et al.* 2003).

the knowledge base of the local peer to a CQ over V_r. Hence, each local query in a mapping assertion is just a single atom (different from equality), and each mapping assertion (in the local peer) has one of the forms

$$q_r \rightsquigarrow \{x \mid C(x)\} \text{ or}$$
$$q'_r \rightsquigarrow \{x_1, x_2 \mid R(x_1, x_2)\},$$

where q_r (resp., q'_r) is a CQ of arity 1 (resp., 2) over the exported fragment of the remote peer, and C (resp., R) is a concept (resp., role) of the local peer. Moreover, we assume that for each concept C or role R of the local peer, there is at most one mapping assertion in which C (resp., R) is used in the local query.

We note that, in data integration terminology, the mappings we have considered here would correspond to a form of mapping called global-as-view (GAV), where the local knowledge base corresponds to a global schema of a data integration system, the remote knowledge base corresponds to a set of data sources, and each concept of the global schema is defined by means of a CQ over the data sources.

Example 7 Consider a peer $P_\ell = \langle K_\ell, V_\ell, M_\ell \rangle$ in which the intensional component for the local knowledge base K_ℓ is the one shown in Figure 3(a).

The graphical representation corresponds to the following set of FOL assertions:

$$\begin{array}{l}
\forall x, y. \mathit{Member}(x, y) \supset \mathit{Employee}(x) \\
\forall x, y. \mathit{Member}(x, y) \supset \mathit{Dept}(y) \\
\forall x. \mathit{Employee}(x) \supset \exists y. \mathit{Member}(x, y) \\
\forall x, y_1, y_2. \mathit{Member}(x, y_1) \wedge \mathit{Member}(x, y_2) \supset y_1 = y_2 \\
\forall x, y. \mathit{Director}(x, y) \supset \mathit{Manager}(x) \\
\forall x, y. \mathit{Director}(x, y) \supset \mathit{Dept}(y) \\
\forall y. \mathit{Dept}(y) \supset \exists x. \mathit{Director}(x, y) \\
\forall x_1, x_2, y. \mathit{Director}(x_1, y) \wedge \mathit{Director}(x_2, y) \supset x_1 = x_2 \\
\forall x. \mathit{Manager}(x) \supset \mathit{Employee}(x)
\end{array}$$

Similarly, the intentional component for the remote knowledge base shown in Figure 3(b) corresponds to the following set of FOL assertions:

$$\begin{array}{l}
\forall x, y. \mathit{MemberR}(x, y) \supset \mathit{EmployeeR}(x) \\
\forall x, y. \mathit{MemberR}(x, y) \supset \mathit{DeptR}(y) \\
\forall x, y. \mathit{BossR}(x, y) \supset \mathit{EmployeeR}(x) \\
\forall x, y. \mathit{BossR}(x, y) \supset \mathit{ManagerR}(y)
\end{array}$$

Assuming that the exported portion of the remote knowledge base coincides with the set of concepts and roles appearing in it, a possible set of mapping assertions for the local peer is shown in Figure 4. ■

Computing What-To-Ask

We now present an algorithm that solves the What-To-Ask problem in the setting described above. We consider a local peer $P_\ell = \langle K_\ell, V_\ell, M_\ell \rangle$, a remote peer $P_r = \langle K_r, V_r, \emptyset \rangle$, and a client's conjunctive query q that is accepted by P_ℓ. In a nutshell, our 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 that is relevant for answering q; 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.

Before proceeding further, we point out that we assume in the following that the theory $K_\ell \cup K_r \cup M_\ell$ is consistent, and we do not consider the case where the knowledge bases at the peers are mutually inconsistent.[4]

Let us now formally describe the algorithm. To this aim, we need a preliminary definition.

Definition 8 Given a CQ q, we say that a variable x is *unbound* in q if it occurs only once in q, otherwise we say that x is *bound* in q. Notice that variables occurring in the head of the query are all bound. A *bound term* is either a bound variable or a constant. ■

In Figure 5 we define the algorithm computeWTA. In the algorithm, each unbound variable is represented by the symbol $\_$. Also, $q[g/g']$ denotes the query obtained from q by replacing the atom g with a new atom g'.

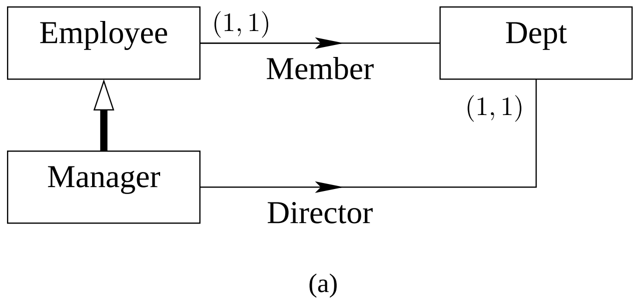
For each query $q \in Q$, the algorithm first checks if there exists an assertion stating a semantic relationship among classes and roles of K_ℓ that can be used to produce a new query to be added to the set Q (steps (a), (b), and (c)). Three kinds of assertions are taken into account: (*i*) subsumption between classes, (*ii*) participation of classes in roles, and (*iii*) mandatory participation of classes in roles. computeWTA makes use of these assertions as rewriting rules that allow to reformulate the original query q into a set of queries, by compiling away the knowledge specified by K_ℓ that is relevant for computing $\mathit{cert}(q, K_\ell \cup K_r \cup M_\ell)$. More precisely, the algorithm produces from q a new query for each atom g in q whose symbol corresponds to a class or role involved in an assertion and such that the assertion propagates all bound terms occurring in g. The new query is obtained from q by replacing g with a reformulation g' of g, constructed according to the assertion.

Then, computeWTA checks if q contains two atoms g_1 and g_2 that unify (step (d)). In this case, the algorithm computes the query $\mathit{reduce}(q, g_1, g_2)$, which is obtained by applying to the query q the *most general unifier* between g_1 and g_2 (Lloyd 1987). Such a new query is then transformed by the function τ, which replaces with $\_$ each variable symbol x such that there is a single occurrence of x in q. The use of τ is necessary in order to guarantee that each unbound variable is represented by the symbol $\_$. The query is then added to Q.

Finally, computeWTA reformulates the queries produced in the above steps into a set of queries accepted by the remote peer P_r, by means of the procedure Mref. Such a procedure implements a standard unfolding technique (Ullman 1997): roughly speaking, mapping assertions are used as rewriting rules for translating the initial set of queries into a set of queries accepted by the remote peer.

In the following we show termination of computeWTA, and soundness of the algorithm with respect to the What-To-Ask problem.

[4]For an analysis on the inconsistency problem in the context of database and knowledge base integration see, for example, (Calì, Lembo, & Rosati 2003; Subrahmanian 1994).



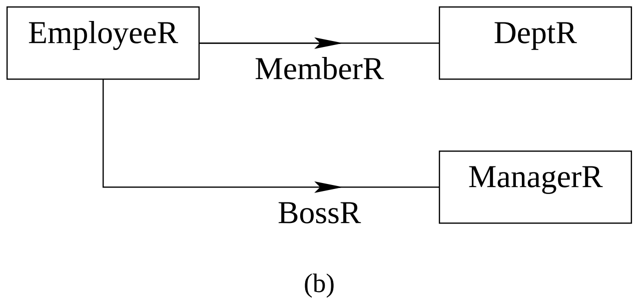


Figure 3: Intensional component of the local and remote knowledge bases for Example 7

$$\begin{array}{rcl}
\{x \mid DeptR(x)\} & \rightsquigarrow & \{x \mid Dept(x)\} \\
\{x \mid EmployeeR(x)\} & \rightsquigarrow & \{x \mid Employee(x)\} \\
\{x \mid ManagerR(x)\} & \rightsquigarrow & \{x \mid Manager(x)\} \\
\{x, y \mid \exists z. BossR(x,z) \wedge MemberR(z,y)\} & \rightsquigarrow & \{x, y \mid Director(x,y)\} \\
\{x, y \mid MemberR(x,y)\} & \rightsquigarrow & \{x, y \mid Member(x,y)\}
\end{array}$$

Figure 4: Mapping assertions for the local peer for Example 7

Theorem 9 *Let $P_\ell = \langle K_\ell, V_\ell, M_\ell\rangle$ be a local peer, $P_r = \langle K_r, V_r, \emptyset\rangle$ a remote peer, and q a CQ accepted by P_ℓ. Then,* computeWTA(q, P_ℓ) *terminates.*

Proof (sketch). Termination of the algorithm follows immediately from the fact that the number of conjunctive queries that can be generated by the algorithm is finite. Indeed, each replacement performed by the algorithm replaces an atom by a single atom, and hence the length (i.e., the number of atoms) of a generated query is at most equal to the length of the query q. Moreover, since the number of assertions in K_ℓ is finite, the number of different atoms that can be generated by the algorithm is finite. □

Theorem 10 *Let $P_\ell = \langle K_\ell, V_\ell, M_\ell\rangle$ be a local peer, $P_r = \langle K_r, V_r, \emptyset\rangle$ a remote peer, and q a CQ accepted by P_ℓ. Then,* computeWTA(q, P_ℓ) *returns a solution for $WTA(q, P_\ell, P_r)$.*

Proof (sketch). To prove the claim we need to show that

$$cert(q, K_\ell \cup K_r \cup M_\ell) = cert(q, K_\ell) \cup \bigcup_{i=1}^{n} cert(q_r^i, K_r)$$

where $\{q_r^1, \ldots, q_r^n\}$ are the queries accepted by P_r that are returned by computeWTA. We proceed in three steps.

In the first step we show that

$$cert(q, K_\ell \cup K_r \cup M_\ell) = \bigcup_{i=1}^{n} cert(q_\ell^i, K_\ell^E \cup K_r \cup M_\ell),$$

where:

- K_ℓ^E is the knowledge base obtained from K_ℓ by deleting all assertions of the intentional component (i.e., subsumption between classes, typing of roles, mandatory participation to roles, and functionality of roles), keeping only the extensional component (together with the alphabet of classes and roles).
- $\{q_\ell^1, \ldots, q_\ell^n\}$ is the set of queries Q (see Figure 5) produced by computeWTA at the end of the repeat-until cycle (i.e., before the final reformulation by means of the operator *Mref*).

Roughly speaking, the above property states that the algorithm first compiles away the intensional component of K_ℓ into the queries $\{q_\ell^1, \ldots, q_\ell^n\}$ in such a way that

$$cert(q, K_\ell) = \bigcup_{i=1}^{n} cert(q_\ell^i, K_\ell^E).$$

In the second step, we notice that K_ℓ^E cannot contribute in inferring new tuples in the answer to each q_ℓ^i apart for those retrievable from its extensional knowledge. Hence, we get such tuples by simply relying on the certain answers service provided by the local peer, i.e.,

$$cert(q, K_\ell \cup K_r \cup M_\ell) = cert(q, K_\ell) \cup \bigcup_{i=1}^{n} cert(q_l^i, K_r \cup M_\ell)$$

Finally, in order to get tuples inferred by $K_r \cup M_\ell$ we apply the procedure *Mref* (unfolding) to $\{q_\ell^1, \ldots, q_\ell^n\}$ (last step of the algorithm). From soundness and completeness of unfolding, we easily obtain

$$cert(q, K_\ell \cup K_r \cup M_\ell) = cert(q, K_\ell) \cup \bigcup_{i=1}^{n} cert(q_r^i, K_r),$$

thus proving the claim. □

Notice that computeWTA does not consider functionality constraints asserted on the components of the roles in K_ℓ for computing the set Q. Indeed, as the above theorem shows, functionality constraints do not interact with the other dependencies enforced by the knowledge base, thus allowing computeWTA to proceed as if they would not be specified.

```
Algorithm computeWTA$(q, P_\ell)$

Input: CQ $q$, ontology-based peer $P_\ell = \langle K_\ell, V_\ell, M_\ell \rangle$
Output: set of conjunctive queries $Mref(Q, M_\ell)$ over $P_r$

begin
  $Q \leftarrow \{q\}$;
  repeat
    $Q_{aux} \leftarrow Q$;
    for each $q \in Q_{aux}$ do
    (a) for each atom $C(x)$ in $q$ do
          for each assertion in $K_\ell$ of the form $\forall x. C_1(x) \supset C_2(x)$
                  (i.e., subsumption) do
            $Q \leftarrow Q \cup \{\, q[C_1(x)/C_2(x)] \,\}$;
          for each assertion in $K_\ell$ of the form $\forall x, y. R(x, y) \supset C(x)$
                  (i.e., typing of the first role) do
            $Q \leftarrow Q \cup \{\, q[C(x)/R(x, \_ \,)] \,\}$
          for each assertion in $K_\ell$ of the form $\forall x, y. R(x, y) \supset C(y)$
                  (i.e., typing of the second role) do
            $Q \leftarrow Q \cup \{\, q[C(x)/R(\_, x)] \,\}$;
    (b) for each atom $R(x, \_ \,)$ in $q$ do
          for each assertion in $K_\ell$ of the form $\forall x. C(x) \supset \exists y. R(x, y)$
                  (i.e., mandatory participation to the first component) do
            $Q \leftarrow Q \cup \{\, q[R(x, \_ \,)/C(x)] \,\}$;
    (c) for each atom $R(\_, y)$ in $q$ do
          for each assertion in $K_\ell$ of the form $\forall y. C(y) \supset \exists x. R(x, y)$
                  (i.e., mandatory participation to the second component) do
            $Q \leftarrow Q \cup \{\, q[R(\_, y)/C(y)] \,\}$;
    (d) for each pair of atoms $g_1, g_2$ in $q$ do
          if $g_1$ and $g_2$ unify
          then $Q \leftarrow Q \cup \{\tau(reduce(q, g_1, g_2))\}$
  until $Q_{aux} = Q$;

  $Q_{res} = Mref(Q, M_\ell)$;
  return $Q_{res}$
end
```

Figure 5: Algorithm computeWTA

We point out that this property cannot be always generalized to more powerful languages for specifying ontologies. In the next section we will discuss one of such cases, and provide a significant example in which this behavior does not hold.

Next, we turn to computational complexity of the algorithm and provide the following result.

Theorem 11 *Let $P_\ell = \langle K_\ell, V_\ell, M_\ell \rangle$ be a local peer, $P_r = \langle K_r, V_r, \emptyset \rangle$ a remote peer, and q a CQ accepted by P_ℓ. Then, the computational complexity of* $\mathsf{computeWTA}(q, P_\ell)$ *is exponential in the size of the query q and polynomial in the size of K_ℓ and M_ℓ.*

Proof. We observe that the maximum number of atoms of a generated query is equal to the number k of atoms of the initial query q, and that for each atom of the query, the number of different atoms that can be generated by the algorithm is bound by the number of assertions that can be specified in K_ℓ and that can be applied in the computation. It is immediate to verify that such a number is $O(n^2)$, where n is the size of K_ℓ. Therefore, the computational complexity of the algorithm is $n^{O(k)}$. Hence, if k is assumed to be constant, the computational complexity of the algorithm is polynomial. □

We point out that, since the number of queries generated by computeWTA is exponential only in the size of the initial query, and typically such a size can be assumed to be small, the exponential blow-up is not likely to be a problem in practice.

Example 7 (continued) Consider the query

$$q_0 = \{x \mid \mathit{Employee}(x)\}$$

that is accepted by the local peer P_ℓ, and execute $\mathsf{computeWTA}(q_0, P_\ell)$. Since $\mathit{Manager}$ is subsumed by $\mathit{Employee}$, then the algorithm produces the query

$$q_1 = \{x \mid \mathit{Manager}(x)\},$$

and since $\mathit{Employee}$ is the first component of the role Member, then the algorithm produces the query

$$q_2 = \{x \mid \mathit{Member}(x, \_)\}.$$

It should be easy to see that the algorithm does not generate any other reformulation. Then, computeWTA applies the operator Mref to the set $Q = \{q_0, q_1, q_2\}$, thus producing the queries

$$\{x \mid \mathit{EmployeeR}(x)\},$$
$$\{x \mid \mathit{ManagerR}(x)\}, \text{and}$$
$$\{x \mid \mathit{MemberR}(x, y)\},$$

where the symbol $\_$ has been replaced by the new fresh variable symbol y. ■

Adding Subsumption between Roles

As shown by Theorem 10, in the specialized framework of the previous section, the What-To-Ask problem has always a solution (which can be computed by the algorithm computeWTA). In this section we show that, if we only add to the ontology language L_K^O the possibility of specifying subsumption relations between roles, such a property does not hold anymore, i.e., What-To-Ask may have no solutions.

We define an new ontology language L_K^{O+}, containing the same constructs as L_K^O, except that, in addition, we allow for expressing subsumption relations between roles. We represent an assertion stating that a role R_1 is subsumed by a role R_2 by using a thick hollow arrow from R_1 to R_2. The corresponding FOL formula specifying the semantics is:

$$\forall x, y. R_1(x, y) \supset R_2(x, y).$$

We now prove that, for peers using the ontology language L_K^{O+}, the What-To-Ask problem does not always have a solution.

Theorem 12 *There exists a pair of knowledge-based peers P_ℓ and P_r with $K_\ell, V_r \in L_K^{O+}$ and a CQ q accepted by P_ℓ such that $WTA(q, P_\ell, P_r)$ has no solutions whenever the language accepted by P_r is a subset of the language of FOL queries.*

Proof. We exhibit an example of a pair of knowledge-based peers P_ℓ and P_r such that the claim holds. More precisely, let us consider $P_\ell = \langle K_\ell, V_\ell, M_\ell \rangle$, $P_r = \langle K_r, V_r, \emptyset \rangle$ where:

- V_ℓ is the ontology displayed in Figure 6, i.e., it comprises the classes C_1, C_2, C_3, C_4, the roles R_1 (between C_2 and C_1), R_2 (between C_1 and C_2), and R_3 (between C_3 and C_4), the mandatory participation of C_2 to R_1 and of C_1 to R_2, the functional participation of C_3 to R_3, the subsumption between roles R_1 and R_2 and between roles R_2 and R_3;
- $K_\ell = V_\ell \cup \{R_1(a, b)\}$;
- M_ℓ consists of the single assertion
$$\{x, y \mid R_r(x, y)\} \rightsquigarrow \{x, y \mid R_3(x, y)\};$$
- $V_r = \{R_r\}$;
- K_r is simply a set of facts for R_r.

We prove that the answer to the boolean query $\{R_1(c, d)\}$ over V_ℓ is true if and only if the following condition holds:

> *[Cond] There exist $n + 1$ constants $a_1, \ldots, a_{n+1}$ (with n even) such that $a_1 = b$, $a_n = c$, $a_{n+1} = d$ and $R_r(a_i, a_{i+1}) \in K_r$ for $1 \le i \le n$.*

Indeed, if condition [Cond] holds, then it is immediate to verify that, due to (*i*) the functionality of the participation of C_3 to R_3, (*ii*) the two subsumption relation between the three roles, and (*iii*) the two mandatory participations of C_1 and C_2, in each model $\mathcal{I}$ for $K_\ell \cup K_r \cup M$ each tuple of the form $\langle a_i, a_{i+1} \rangle$ must belong to $R_1^{\mathcal{I}}$ if i is even and to $R_2^{\mathcal{I}}$ if i is odd, which implies that $\langle c, d \rangle \in R_1^{\mathcal{I}}$. Conversely, if condition [Cond] does not hold, then it is immediate to exhibit a model $\mathcal{I}$ for $K_\ell \cup K_r \cup M$ in which the tuple $\langle c, d \rangle$ is not in $R_1^{\mathcal{I}}$.

Then, observe that verifying the above condition [Cond] requires to compute the transitive closure of R_r, which in general cannot be done through a finite number of FOL queries over P_r. Therefore, the claim follows. □

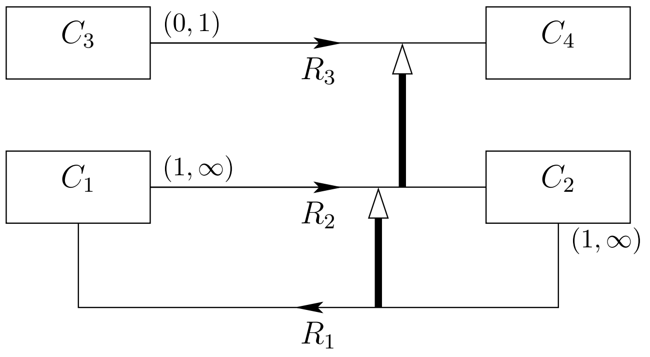


Figure 6: Ontology of peer P_ℓ in the proof of Theorem 12

Notice that the above theorem states that the What-To-Ask problem may have no solutions (and its proof exhibits an example of peers for which the problem has no solution) even under the assumption that the remote peer accepts arbitrary FOL queries over the ontology V_r. That is, even if we empower the answering abilities of the remote peer to the full FOL language, the What-To-Ask problem may have no solutions. This fact highlights the crucial role played by the expressiveness of the language for specifying the local peer knowledge base K_ℓ in the What-To-Ask problem: indeed, by simply adding the possibility of expressing role subsumption to our specialized framework, we are not guaranteed anymore that a solution to our problem always exists.

Related Work

The problem studied in this paper is crucial in several contexts.

First, our work provides new results to data integration. Indeed, the algorithm computeWTA can be seen as a new query rewriting algorithm, in particular in the case where the global schema is expressed as an ontology. Although recent papers address the issue of query rewriting under integrity constraints (see, for example, (Calì *et al.* 2003; Calì, Lembo, & Rosati 2003)), the constraints deriving by our ontology languages have not been considered in the data integration literature.

The What-To-Ask problem is of clear relevance for the Semantic Web, even if research on the Semantic Web has focused more on the problem of ontology matching (i.e., finding the mapping between peers). The problem of reformulating queries over ontologies has been investigated in (Calvanese, De Giacomo, & Lenzerini 2002; Tzitzikas, Constantopoulos, & Spyratos 2001). To the best of our knowledge, the present work is the first one that deals with query reformulation when one is forced to rely only on the query answering services of the peers.

The research on P2P and Grid computing is also related to our approach. However, the works on query answering in such architectures either assume that peers are databases (and not knowledge bases, nor ontologies), or deal with query languages that are much less expressive than those considered in our investigation. A notable exception is (Serafini & Ghidini 2000), where peers are assumed to be knowledge bases. However, the problem studied there is different from our reformulation problem.

Finally, our work can be seen as providing a service-oriented architecture, where the algorithm presented in Section aims at computing the "composition" of the query answering services provided by the peers. We are aware of only one paper with similar objectives, namely (Thakkar, Ambite, & Knoblock 2003). However, in that paper, peers do not have constraints, and the query reformulation problem is therefore easier than the one addressed here.

Conclusions

In this paper we have formally defined the What-To-Ask problem, which captures a fundamental issue in a networked environment based on information exchange. We have seen that even small changes in the representation formalism may affect seriously the ability of dealing with this problem, and hence particular care must be put into choosing the form of knowledge used by interoperating systems. To show this, it has been sufficient to look at a simplified setting with only two interoperating peers.

The impact of having more than two peers (each with its own mappings) remains to be investigated. Also, it is worth studying query answering in the case where the knowledge bases at the peers are mutually inconsistent (in the line of (Calì, Lembo, & Rosati 2003)), since this is of relevance in real domains.

Acknowledgments

This research has been partially supported by the Projects INFOMIX (IST-2001-33570) and SEWASIE (IST-2001-34825) funded by the EU, by MIUR - Fondo Speciale per lo Sviluppo della Ricerca di Interesse Strategico — project "Società dell'Informazione", subproject SP1 "Reti Internet: Efficienza, Integrazione e Sicurezza", and by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts.

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

Bernstein, P. A.; Giunchiglia, F.; Kementsietsidis, A.; Mylopoulos, J.; Serafini, L.; and Zaihrayeu, I. 2002. Data management for peer-to-peer computing: A vision. In *Proc. of the 5th Int. Workshop on the Web and Databases (WebDB 2002)*.

Calì, A.; Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2003. Data integration under integrity constraints. *Information Systems*. To appear.

Calì, A.; Lembo, D.; and Rosati, R. 2003. Query rewriting and answering under constraints in data integration systems. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*.

Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2002. A framework for ontology integration. In Cruz, I.; Decker, S.; Euzenat, J.; and McGuinness, D., eds., *The Emerging Semantic Web — Selected Papers from the First Semantic Web Working Symposium*. IOS Press. 201–214.

Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W.-C. 2004. Composing schema mappings: Second-order dependencies to the rescue. In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*.

Halevy, A.; Ives, Z.; Suciu, D.; and Tatarinov, I. 2003. Schema mediation in peer data management systems. In *Proc. of the 19th IEEE Int. Conf. on Data Engineering (ICDE 2003)*, 505–516.

Halevy, A. Y. 2000. Theory of answering queries using views. *SIGMOD Record* 29(4):40–47.

Halevy, A. Y. 2001. Answering queries using views: A survey. *Very Large Database J.* 10(4):270–294.

Heflin, J., and Hendler, J. 2001. A portrait of the semantic web in action. *IEEE Intelligent Systems* 16(2):54–59.

Hull, R.; Benedikt, M.; Christophides, V.; and Su, J. 2003. E-services: a look behind the curtain. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, 1–14. ACM Press and Addison Wesley.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, 233–246.

Levesque, H. J., and Lakemeyer, G. 2001. *The Logic of Knowledge Bases*. The MIT Press.

Lloyd, J. W. 1987. *Foundations of Logic Programming (Second, Extended Edition)*. Berlin, Heidelberg: Springer.

Madhavan, J., and Halevy, A. Y. 2003. Composing mappings among data sources. In *Proc. of the 29th Int. Conf. on Very Large Data Bases (VLDB 2003)*, 572–583.

Mattos, N. M. 2003. Integrating information for on demand computing. In *Proc. of the 29th Int. Conf. on Very Large Data Bases (VLDB 2003)*, 8–14.

Papazoglou, M. P.; Kramer, B. J.; and Yang, J. 2003. Leveraging Web-services and peer-to-peer networks. In *Proc. of the 15th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2003)*, 485–501.

Serafini, L., and Ghidini, C. 2000. Using wrapper agents to answer queries in distributed information systems. In *Proc. of the 1st Int. Conf. on Advances in Information Systems (ADVIS-2000)*, volume 1909 of *Lecture Notes in Computer Science*. Springer.

Subrahmanian, V. S. 1994. Amalgamating knowledge bases. *ACM Trans. on Database Systems* 19(2):291–331.

Thakkar, S.; Ambite, J.-L.; and Knoblock, C. A. 2003. A view integration approach to dynamic composition of Web services. In *Proc. of 2003 ICAPS Workshop on Planning for Web Services*.

Tzitzikas, Y.; Constantopoulos, P.; and Spyratos, N. 2001. Mediators over ontology-based information sources. In *Proc. of the 2nd Int. Conf. on Web Information Systems Engineering (WISE 2001)*, 31–40.

Ullman, J. D. 1997. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, 19–40. Springer.