The What-To-Ask Problem for Ontology-Based Peers

Diego Calvanese[1(✉)], Giuseppe De Giacomo[2], Domenico Lembo[2], Maurizio Lenzerini[2], and Riccardo Rosati[2]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
calvanese@inf.unibz.it
[2] Sapienza Università di Roma, Rome, Italy
{degiacomo,lembo,lenzerini,rosati}@diag.uniroma1.it

Abstract. The issue of cooperation, integration, and coordination between information peers has been addressed over the years both in the context of the Semantic Web and in several other networked environments, including data integration, Peer-to-Peer and Grid computing, service-oriented computing, distributed agent systems, and collaborative data sharing. One of the main problems arising in such contexts is how to exploit the mappings between peers in order to answer queries posed to one peer. We address this issue for peers managing data through ontologies and in particular focus on ontologies specified in logics of the *DL-Lite* family. Our goal 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 both a positive and a negative result. Namely, we first prove that a solution to this problem always exists when the ontology is specified in $DL\text{-}Lite_{\mathcal{R}}$, and we provide an algorithm to compute it. Then, we show that for the case of $DL\text{-}Lite_{\mathcal{F}}$ the problem may have no solution. We finally illustrate that a solution to our problem can still be found even for more general networks of peers, and for any language of the *DL-Lite* family, provided that we interpret mappings according to an epistemic semantics, rather than the usual first-order semantics.

1 Introduction

In the era towards a data-driven society, the issue of cooperation, integration, and coordination between data stored in different nodes of a network is of paramount importance. Indeed, recent years have shown the need to deal with networked data in large-scale, distributed settings, and it is not surprising that the abstraction of networked data systems appears in many disciplines, including Web Science and Peer-to-Peer computing [3,8,26], Semantic Web [1,42], Data Management [12,27,31,37,44], and Knowledge Representation [25,30,42,46].

C. Lutz et al. (Eds.): Baader Festschrift, LNCS 11560, pp. 187–211, 2019.
https://doi.org/10.1007/978-3-030-22102-7_9

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. A mapping is a statement specifying that some relationship exists between pieces of information held by one peer and pieces of information held by another peer. The whole knowledge of the system is fully distributed, without any central entity holding a global view of information, or controlling the overall operation of the system.

The basic problems arising in this architecture include the following:

– how to discover, express, and compose the mappings between peers (see, for instance, [8,23,26,33,39]),
– how to exchange data between peers based on the specified mappings (see, for instance, [24,31,32]),
– how to exploit the mappings in order to answer queries posed to one peer [28, 37,40].

The latter is the problem studied in this paper. Although several interesting results have been reported in each of the above mentioned 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 an ontology.

Our goal is to present some basic, fundamental results on this problem. Given the fundamental nature of our investigation, we consider a simplified setting where the whole system is constituted by only two peers, called local and remote, respectively. Information in the remote peer is related to the information in the local peer by means of suitable mappings (cf. Fig. 1). Interestingly, despite the fact that this setting might look elementary, it will nevertheless allow us to uncover various subtleties of an interoperating ontology-based peer system.

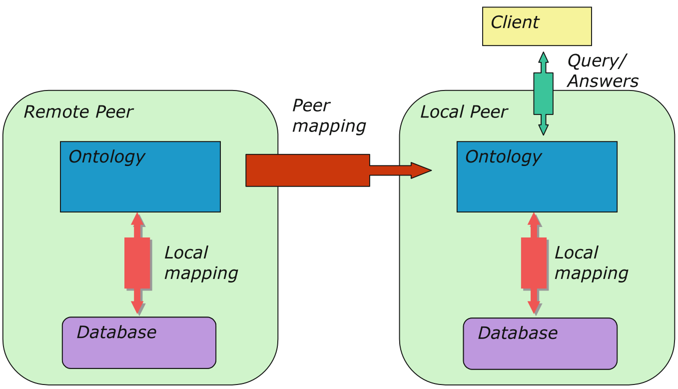

Fig. 1. Ontology-to-ontology: a simple form of interoperation among peer ontologies

In our study, we make several assumptions, that are made explicit here:

- In contrast with most of the papers in peer-to-peer data management, we assume that each peer does not simply store data, but holds a knowledge base. In particular, we explore the context where each peer models its knowledge base by means of an ontology.
- The ontology at each peer specifies both intensional knowledge (general rules) and extensional one (individual facts). Actually, the latter may be managed through a relational DBMS, and therefore represented by a database connected to the ontology via local mappings, as shown in Fig. 1. So, if we have data sources linked to our ontology through mappings, they are seen as internal components within a peer. In other words, each peer can be seen as an Ontology-based data access (OBDA) system [13], and the novelty with respect to the usual notion of OBDA is represented by the fact that mappings connect peers, and not simply data sources to ontologies.
- We concentrate our attention to the issue of answering queries posed to the local peer.
- We assume that each of the two peers provides the service of answering queries expressed over its underlying ontology. Note that answering a query for a peer requires reasoning over the ontology by means of deduction, rather than simply evaluating the query expression over a database.
- We assume that query answering is the only basic service provided by each peer. In other words, while processing a query posed to the local peer, the query answering services provided by each of the two peers are the only basic services that can be relied upon.
- In order to address the problem in the most general way, we assume that the local peer can only collect the answers received by the remote peers, and add them to the answers obtained by accessing its own data. In other words, no computational power is available at the local peer to process the tuples returned by the remote peer, except for just adding them to the result of the whole query.

We believe that the above assumptions faithfully capture the modular structure of a peer-to-peer system, and generalize the existing investigation of peer-to-peer architectures to the case where each peer is seen as an agent holding complex knowledge, instead of simply data.

In this context, the basic problem we address is the following: given a query posed to the local peer, find a way to answer the query by relying only on the two query answering services available at the two peers. Thus, when answering the query posed to the local peer, we have to figure out which queries to send to the remote peer in order for the local peer to be able to return the correct and complete set of answers to the original query. This is why we call this problem the "What-To-Ask" problem (cf. [14]).

Example 1. Consider a music sharing system, and assume that the peer SongUniverse stores its own information about songs, and has a mapping specifying that other songs, in particular live rock songs, can be retrieved from the remote peer RockPlanet. Now, suppose that Carol interacts with the SongUniverse peer, and asks for all live songs of U.K. artists. What this peer can do in order to answer Carol's query at best is to: *(i)* directly provide her with the live songs of U.K. artists that it stores locally, *(ii)* use its general knowledge about music to deduce that also live rock songs suit Carol's needs, *(iii)* use the mapping to reformulate Carol's request in terms of RockPlanet knowledge, in particular asking to the remote peer the right query to retrieve all live rock songs of U.K. artists. ◁

In this paper, we study the What-To-Ask problem in a setting where the two peers hold an ontology expressed in a Description Logic of the *DL-Lite* family [16]. Specifically, we present the following contributions.

1. We formalize the above mentioned two-peer architecture, we define its semantics, and we give a precise characterization of the semantics of query answering (Sect. 3).
2. We provide both the intuition and 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, only the query answering services available at the two peers can be relied upon (Sect. 3).
3. We show that in the case of ontologies specified in $\textit{DL-Lite}_{\mathcal{R}}$ there is an algorithm that allows us to solve any instance of "What-To-Ask", i.e., that allows us to compute what we should ask to the remote peer in order to answer a query posed to the local peer. One of the basic ingredients of the algorithm is the ability of reformulating the query on the basis of the local peer ontology and the mappings, so as to deduce the correct queries to send to the remote peer (Sect. 4).
4. We show that in the case of $\textit{DL-Lite}_{\mathcal{F}}$, the "What-To-Ask" problem may not admit any solution. This shows that particular attention should be devoted to the trade-off between the expressive power of the ontology language and the complexity/feasibility of reasoning (Sect. 5).
5. We finally discuss how to overcome the limitation above by making use of mappings that explicitly take into account that ontologies are autonomous agents that provide as query answering service the (independent) generation of certain answers. This calls for the usage of (auto-)epistemic operators (Sect. 6).

To complete the description of the organization of the paper, Sect. 2 illustrates some preliminary notions that will be used in the technical development, and Sect. 7 presents some concluding remarks. Finally, we note that this paper is a revised and extended version of [14].

2 Preliminaries

We introduce now the ontology languages on which we base the technical development in the next sections. Specifically, we rely on Description Logics (DLs) [6], which are logic's that represent the domain of interest in terms of *concepts*, denoting sets of objects, and *roles*, denoting binary relations between objects. Complex concept and role expressions are constructed by applying suitable constructs, starting from a set of atomic concepts and roles.

2.1 The *DL-Lite* Family

We focus here on a family of lightweight DLs, called the *DL-Lite family* [16], and introduce three prominent logics of this family, namely *DL-Lite*, *DL-Lite*$_\mathcal{R}$ and *DL-Lite*$_\mathcal{F}$. In the core language of the family, called *DL-Lite*, (basic) *concepts* C and *roles* R are formed according to the following syntax:

$$C \longrightarrow A \mid \exists R \qquad\qquad R \longrightarrow P \mid P^-$$

where A denotes an atomic concept, P an atomic role, P^- the *inverse* of P, and $\exists R$ an unqualified existential quantification. Intuitively, P^- denotes the inverse of the binary relation denoted by P, while $\exists R$ denotes the domain of (the binary relation denoted by) R, i.e., the projection of R on its first component.

A DL *ontology* $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ encodes the knowledge about the domain of interest in two distinct components: the *TBox* (for terminological box) $\mathcal{T}$ specifies general knowledge about the conceptual elements of the domain, while the *ABox* (for assertional box) $\mathcal{A}$, specifies extensional knowledge about individual elements of the domain.

In *DL-Lite*, a TBox is formed by a finite set of *inclusion* and *disjointness assertions* between concepts, respectively of the form

$$B_1 \sqsubseteq B_2 \qquad\qquad B_1 \sqsubseteq \neg B_2$$

where B_1 and B_2 are basic concepts. The first assertion expresses that every instance of concept B_1 is also an instance of concept B_2, while the second assertion expresses that the two sets of instances are disjoint. An ABox consists of concept and role membership assertions, respectively of the form

$$A(c) \qquad\qquad P(c, c')$$

where A is an atomic concept, P an atomic role, and c, c' two constants. The first assertion expresses that the individual denoted by c is an instance of concept A, while the second assertion expresses that the two individuals denoted by c and c' are in relation P.

In *DL-Lite*$_\mathcal{R}$, a TBox may additionally contain *role inclusion* and *disjointness assertions*, respectively of the form

$$R_1 \sqsubseteq R_2 \qquad\qquad R_1 \sqsubseteq \neg R_2$$

where R_1 and R_2 are arbitrary roles. The meaning of such assertions is analogous to the one for concepts.

Instead, in *DL-Lite*$_\mathcal{F}$, a TBox may contain also *functionality assertions* of the form

$$(\mathsf{funct}\ R)$$

asserting that R is a functional role. Such a role R can relate each object to at most one other object.

The semantics of a DL is given in terms of first-order logic interpretations, where an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty *interpretation* *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each role R a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$, in such a way that the following conditions hold. In particular, for the constructs of *DL-Lite* we have:

$$\begin{array}{rcl@{\qquad\qquad}rcl} A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} & P^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\ (\exists R)^{\mathcal{I}} & = & \{o \mid \exists o'.\, (o,o') \in R^{\mathcal{I}}\} & (P^-)^{\mathcal{I}} & = & \{(o_2,o_1) \mid (o_1,o_2) \in P^{\mathcal{I}}\} \\ (\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & (\neg R)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}} \end{array}$$

To specify the semantics of membership assertions, we extend interpretations to constants, by assigning to each constant c a *distinct* object $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Note that this implies that we enforce the *unique name assumption* on constants [6]. Then, to assign semantics to an ontology, we first define when an interpretation $\mathcal{I}$ *satisfies an assertion* α, denoted $\mathcal{I} \models \alpha$, as follows:

- $\mathcal{I} \models E_1 \sqsubseteq E_2$, if $E_1^{\mathcal{I}} \subseteq E_2^{\mathcal{I}}$;
- $\mathcal{I} \models E_1 \sqsubseteq \neg E_2$, if $E_1^{\mathcal{I}} \cap E_2^{\mathcal{I}} = \emptyset$;
- $\mathcal{I} \models (\mathsf{funct}\ R)$, if whenever $\{(o,o_1),\ (o,o_2)\} \subseteq R^{\mathcal{I}}$, then $o_1 = o_2$;
- $\mathcal{I} \models A(c)$, if $c^{\mathcal{I}} \in A^{\mathcal{I}}$;
- $\mathcal{I} \models P(c,c')$, if $(c^{\mathcal{I}}, c'^{\mathcal{I}}) \in P^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ that satisfies all assertions of an ontology $\mathcal{O}$ is called a *model* of $\mathcal{O}$, and is denoted as $\mathcal{I} \models \mathcal{O}$. An ontology that admits a model is called *satisfiable*. Finally, we say that an ontology $\mathcal{O}$ *logically implies* an assertion α, denoted $\mathcal{O} \models \alpha$, if every model of $\mathcal{O}$ satisfies α. Analogous definitions hold when we replace the ontology $\mathcal{O}$ with a TBox $\mathcal{T}$ or an ABox $\mathcal{A}$.

We observe that, despite the simplicity of the language, the logics of the *DL-Lite* family are able to capture the main elements of conceptual modeling formalisms used in databases and software engineering (e.g., Entity-Relationship and UML class diagrams), cf. [13]. Furthermore, *DL-Lite* is one of the classes of DLs for which conjunctive query answering is tractable in data complexity. Other DLs showing this property are $\mathcal{EL}$ [4,5], and all Horn DLs [41]. Moreover, query answering remains tractable in the DL $\mathcal{FL}_0$ for instance queries (whereas answering conjunctive queries in this logic is coNP-complete), as shown in [7].

2.2 Queries over a DL Ontology

We start with a general notion of queries in first-order logic, and then we move to the definition of queries over a DL ontology.

In general, a *query* is an open formula of first-order logic with equalities (FOL in the following). We denote a (FOL) query q as follows

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

where $\phi(x_1, \ldots, x_n)$ is a FOL formula with free variables $x_1, \ldots, x_n$. We call n the *arity* of the query q. Given an interpretation $\mathcal{I}$, $q^{\mathcal{I}}$ is the set of tuples of domain elements that, when assigned to the free variables, make the formula ϕ true in $\mathcal{I}$ [2].

A query over an ontology is a FOL query as above, in which the predicates in ϕ are concepts and roles of the ontology. Among the various queries, we are interested in conjunctive queries, which provide a reasonable trade-off between expressive power and complexity of query processing.

A *conjunctive query* (CQ) q of arity n over an ontology $\mathcal{O}$ is a FOL query of the form

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

where $x_1, \ldots, x_n$ are pairwise distinct variables[1], and $\phi(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a conjunction of atoms whose predicates are concept and roles of $\mathcal{O}$, and whose free variables are the variables in $x_1, \ldots, x_n, y_1, \ldots, y_m$. We call $\exists y_1, \ldots, y_m . \phi(x_1, \ldots, x_n, y_1, \ldots, y_m)$ the *body* of q, $x_1, \ldots, x_n$ the *distinguished variables* of q, and $y_1, \ldots, y_m$ the *non-distinguished variables* of q.

In the following we will not indicate existential variables in queries when not explicitly needed, i.e., we will use $\phi(x_1, \ldots, x_n)$ to indicate $\exists y_1, \ldots, y_m . \phi(x_1, \ldots, x_n, y_1, \ldots, y_m)$.

When a query is posed to an ontology, the ontology should answer the query by returning all tuples of constants from the alphabet Γ that satisfy the query in every interpretation that is a model of the ontology. This is formalized by the following notion of certain answers,

Given a CQ q of arity n over an ontology $\mathcal{O}$, the *certain answers* $cert(q, \mathcal{O})$ to q over $\mathcal{O}$ is the set of tuples of constants:

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

3 What-To-Ask

In this section we set up a formal framework for interoperation between ontology-based peers, and we formally define the What-To-Ask problem.

3.1 Ontology-Based Peer Framework

As already said, the extensional level of the ontology can be virtually generated by means of local mappings connecting the intensional level of the ontology to a database. In this case, a peer is actually an autonomous ontology-based data access system, or an ontology-based data integration system in the case where the underlying database is federated [43]. For the sake of simplicity, in this paper we consider a peer ontology of a more plain form, in which both the intensional and the extensional knowledge are represented in a first-order logic theory, and more precisely as a DL ontology. All the results we present in fact apply almost straightforwardly to peers that are ontology-based data integration systems.

[1] For simplicity of presentation, we have assumed here that conjunctive queries contain neither constants nor repeated variables among $x_1, \ldots, x_n$, but all our results extend to the case where this restrictions do not apply.

Each peer contains an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ that it can use to make logical inferences. Agents willing to use the peer, here called *clients*, can *ask* the peer queries specified over the peer ontology (i.e., over its TBox).

Besides using its ontology $\mathcal{O}$ for answering queries, each peer can be connected with other peers by means of *mappings*. Mappings establish the relationship between the concepts represented in the peers. When answering a query, each peer can also *ask* queries to the other peers based on such mappings.

In this paper we focus on a system made up by two interoperating peers. One of them, called local peer, is the one the client interacts with by asking queries. The other peer will be referred to as the remote peer, and the knowledge contained in it can be exploited by the local peer through the mappings, so as to enhance the capability of the local peer to provide answers to queries posed by the client. We further assume that, while the local peer exploits the remote peer through the mappings, the remote peer has no information about the local peer, and thus it cannot use in any way the knowledge of the local peer.

Next, we move to the formalization of the framework. We assume that all peers share the same set of constants, denoted by Γ, and we assume that Γ is part of the alphabet of the ontology in each peer. We also assume that in every interpretation different constants are interpreted with different domain elements, i.e., we adopt the *unique name assumption*. With this assumption in place, we turn our attention to the definition of ontology-based peers.

Definition 1. An *ontology-based peer* (or simply *peer*) is a pair $P = \langle \mathcal{O}, M \rangle$ where:

- $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ is the peer ontology, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox;
- M is a set of mapping assertions, whose form will be illustrated below.

We also call the pair $\langle \mathcal{T}, M \rangle$ the *specification* of P, denoted by P^S, and call the ABox $\mathcal{A}$ the *instance* of P. $\triangleleft$

Queries posed to a peer are specified over its TBox $\mathcal{T}$. The queries that we consider are conjunctive queries (cf. Sect. 2). We concentrate on systems consisting of two peers, namely $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$, called *local peer*, which is the peer to which the client may connect, and $P_r = \langle \mathcal{O}_r, \emptyset \rangle$, called *remote peer*. The alphabets of $\mathcal{O}_\ell$ and $\mathcal{O}_r$ share the set of constants Γ, but contain disjoint sets of relation names. Observe that the remote peer does not contain any mapping assertion. We also assume that both peers may process conjunctive queries posed over them, i.e., they are able to compute certain answers to CQs specified over $\mathcal{O}_\ell$ and $\mathcal{O}_r$, respectively. We say that the class of CQs is *accepted by P_ℓ and P_r*, and we call the pair $\langle P_\ell, P_r \rangle$ an *ontology-to-ontology system*.

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

$$q_r \rightsquigarrow \{x \mid C(x)\} \quad \text{or}$$

$$q'_r \rightsquigarrow \{x_1, x_2 \mid R(x_1, x_2)\},$$

where q_r is a CQ of arity 1 and q'_r a CQ of arity 2 over the remote peer, C is a concept and R a role of the local peer, x is a variable, and x_1 and x_2 are distinct variables.

A mapping assertion $q_r \rightsquigarrow \{x \mid C(x)\}$ has an immediate interpretation as an implication in FOL: it states that

$$\forall x. \phi_r(x) \rightarrow C(x),$$

where ϕ_r is the open formula constituting the query q_r. Analogously, the mapping assertion $q'_r \rightsquigarrow \{x_1, x_2 \mid R(x_1, x_2)\}$ is interpreted as

$$\forall x_1, x_2. \phi_r(x_1, x_2) \rightarrow R(x_1, x_2).$$

We note that, in data integration terminology, the mappings we have considered here would correspond to a form of mappings called global-as-view (GAV), where the local ontology corresponds to a global schema of a data integration system, the remote ontology 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.

3.2 The What-To-Ask Problem

A natural task to consider, given a client's query q specified over the local peer P_ℓ, is to return the answers that can be inferred from all the knowledge in the system, that is, return the certain answers $cert(q, \mathcal{O}_\ell \cup M_\ell \cup \mathcal{O}_r)$[2]. Clearly, such a task is meaningful in the case where the axioms in $\mathcal{O}_\ell$, M_ℓ, and $\mathcal{O}_r$ are known and usable by the query answering algorithm.

Here, however, we consider a different setting, in which we assume that the remote peer can only be used by invoking its query answering service, and the local peer has minimal computational capabilities to perform post-processing of the answers provided by the remote peer. More precisely, we assume that:

- each peer $P = \langle \mathcal{O}, M \rangle$ is able to provide the certain answers $cert(q, \mathcal{O})$ to queries q specified over P itself, and
- each peer does not have additional computation capabilities, and is only able to redirect its own answers and those produced by the other peer to the output[3].

Under these assumptions, computing the certain answers to a query posed to the local peer requires to determine the set of queries to send to the remote peer in such a way that the union of such answers with the certain answers computed locally provides the certain answers to the query. This challenge is formalized in what we call the *What-To-Ask* problem.

Definition 2. Consider a local peer $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$, a remote peer specification $P_r^S = \langle \mathcal{T}_r, \emptyset \rangle$, and a query q specified over P_ℓ. The *What-To-Ask* problem, $WTA(q, P_\ell, P_r^S)$, is defined as follows: *Given as input q, P_ℓ and P_r^S, find a finite set $\{q_r^1, \ldots, q_r^n\}$ of queries, each specified over the remote peer P_r, such that for every instance of the remote peer $\mathcal{A}_r$:*

$$cert(q, \mathcal{O}_\ell \cup M_\ell \cup \mathcal{O}_r) = cert(q, \mathcal{O}_\ell) \cup cert(q_r^1, \mathcal{O}_r) \cup \cdots \cup cert(q_r^n, \mathcal{O}_r).$$

where $\mathcal{O}_r = \langle \mathcal{T}_r, \mathcal{A}_r \rangle$. ◁

[2] Whenever we refer to M_ℓ as part of an ontology, we consider its FOL formulation.
[3] This formally corresponds to computing the *union* of the two sets of answers.

The above definition clearly points out the specific nature of the What-To-Ask problem, where the answers coming from the remote peer are combined using *union only*. In particular, it clarifies the difference with other data interoperability architectures, such as data federation. Indeed, in data federation, the mediator has to decide how to send the query to the various federated databases, and then in principle it can use the whole power of SQL (or relational algebra) to combine the answers returned by the data sources.

Notice that, in general, several solutions to the What-To-Ask problem may exist. However, it is easy to see that all solutions are equivalent from a semantic point of view, i.e., each of them allows us to obtain all certain answers that can be inferred from the knowledge managed by the peer system. 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 one is contained in the set of queries in the latter one. However, we focus here on solving the What-To-Ask problem, i.e., finding *any* solution that satisfies Definition 2, in the specific setting described in the next section, whereas the problem of characterizing when a solution is "better" than another, or finding the "best" solutions with respect to some criteria, is outside the scope of this paper.

In the following, for simplicity, we consider only systems of peers that are *consistent*, i.e., such that their FOL formalization admits at least one model. We will then briefly come back to the issue of (in)consistency in the conclusions.

4 What-To-Ask Problem: Positive Results

We now consider a particular instantiation of the formal framework described in Sect. 3, i.e., we consider specific choices for both the language in which a peer ontology is expressed, and the queries appearing in the mapping assertions. We then study the What-To-Ask problem in the specialized framework. We first present an algorithm, called computeWTA, for the What-To-Ask problem, and then we both prove its termination and correctness, and establish its computational complexity. We also comment on the relationship between the What-To-Ask problem and the task of computing the answers to queries posed to the local peer.

4.1 *DL-Lite*$_{\mathcal{R}}$ Peer Ontologies

We concentrate first on the ontology language in which to express the peer ontology. The language we use for this purpose is *DL-Lite*$_{\mathcal{R}}$.

Example 2. Consider a local peer specification $P_\ell = \langle \mathcal{T}_\ell, M_\ell \rangle$ such that $\mathcal{T}_\ell$ is the following *DL-Lite*$_{\mathcal{R}}$ TBox:

$$
\begin{array}{rclcrcl}
\exists\mathsf{member} & \sqsubseteq & \mathsf{Employee} & \quad & \exists\mathsf{director} & \sqsubseteq & \mathsf{Manager}\\
\exists\mathsf{member}^- & \sqsubseteq & \mathsf{Dept} & & \exists\mathsf{director}^- & \sqsubseteq & \mathsf{Dept}\\
\mathsf{Employee} & \sqsubseteq & \exists\mathsf{member} & & \mathsf{Dept} & \sqsubseteq & \exists\mathsf{director}^-\\
\mathsf{Manager} & \sqsubseteq & \mathsf{Employee} & & \mathsf{director} & \sqsubseteq & \mathsf{member}
\end{array}
$$

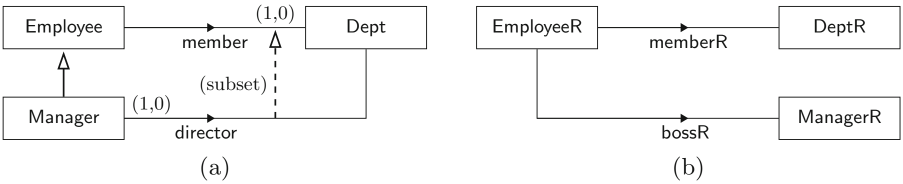

Fig. 2. Intensional component of the local and remote ontologies for Example 2

In this case, such TBox can be directly represented by means of a UML class diagram [45]. Indeed, concepts and roles correspond to UML classes and binary associations, respectively, and role typing assertions are represented in UML by the participation of classes to associations. ISA assertions between concepts correspond to sub-classing, while mandatory participation to roles can be specified in UML by means of multiplicity constraints. Also, ISA assertions between roles can be specified by means of association subsetting. The UML representation of $\mathcal{T}_\ell$ is shown in Fig. 2(a).

Similarly, the following set of *DL-Lite*$_\mathcal{R}$ assertions, providing the representation of the TBox $\mathcal{T}_r$ of a remote peer P_r, corresponds to the UML class diagram shown in Fig. 2(b)[4]:

$$\begin{array}{rclcrcl} \exists\mathsf{memberR} & \sqsubseteq & \mathsf{EmployeeR} & \quad & \exists\mathsf{bossR} & \sqsubseteq & \mathsf{EmployeeR} \\ \exists\mathsf{memberR}^- & \sqsubseteq & \mathsf{DeptR} & & \exists\mathsf{bossR}^- & \sqsubseteq & \mathsf{ManagerR} \end{array}$$

Possible ABoxes $\mathcal{A}_\ell$ and $\mathcal{A}_r$ for the TBoxes given above are represented by the *DL-Lite*$_\mathcal{R}$ assertions below:

$$\begin{array}{ll} \mathsf{Manager}(\mathtt{Mary}) & \quad \mathsf{memberR}(\mathtt{Mary}, \mathtt{D2}) \\ \mathsf{Dept}(\mathtt{D1}) & \quad \mathsf{DeptR}(\mathtt{D3}) \end{array}$$

Finally, a possible set of assertions for the mapping M_ℓ of the local peer P_ℓ is the following:

$$\begin{array}{rcl} \{x \mid \mathsf{DeptR}(x)\} & \rightsquigarrow & \{x \mid \mathsf{Dept}(x)\} \\ \{x \mid \mathsf{EmployeeR}(x)\} & \rightsquigarrow & \{x \mid \mathsf{Employee}(x)\} \\ \{x \mid \mathsf{ManagerR}(x)\} & \rightsquigarrow & \{x \mid \mathsf{Manager}(x)\} \\ \{x, y \mid \exists z.\mathsf{bossR}(x, z) \wedge \mathsf{memberR}(z, y)\} & \rightsquigarrow & \{x, y \mid \mathsf{director}(x, y)\} \\ \{x, y \mid \mathsf{memberR}(x, y)\} & \rightsquigarrow & \{x, y \mid \mathsf{member}(x, y)\} \end{array} \quad \triangleleft$$

4.2 The Algorithm ComputeWTA

Consider a local peer $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$ and a remote peer specification $P_r = \langle \mathcal{T}_r, \emptyset \rangle$, and a client's conjunctive query q that is specified over P_ℓ. In a nutshell, our

[4] Note that, differently from classical UML semantics, we do not consider as disjoint those classes that in the class diagram do not have a common ancestor.

algorithm first reformulates the client's query q into a set Q of conjunctive queries expressed over $\mathcal{T}_\ell$, in which it compiles the knowledge of the local peer that is relevant for answering q; then the algorithm reformulates the queries of Q into a new set of queries specified over the remote peer P_r.

In the following, given a remote instance $\mathcal{A}_r$, we assume that the theory $\mathcal{O}_\ell \cup M_\ell \cup \mathcal{O}_r$, where $\mathcal{O}_r = \langle \mathcal{T}_r, \mathcal{A}_r \rangle$, is consistent, i.e., there exists at least one first-order interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{O}_\ell \cup M_\ell \cup \mathcal{O}_r$. Notice that when the theory is inconsistent, the certain answers to a query q of arity n over $\mathcal{O}_\ell \cup M_\ell \cup \mathcal{O}_r$ are all the n-tuples constructible from constants of Γ. Therefore, computing the certain answers to q in this situation does not lead to a meaningful result[5].

Algorithm $\mathsf{computeWTA}(q, P_\ell)$
Input: CQ q, local peer $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$, where $\mathcal{O}_\ell = \langle \mathcal{T}_\ell, \mathcal{A}_\ell \rangle$ is a *DL-Lite*$_\mathcal{R}$ ontology
Output: set of conjunctive queries
begin
    $Q_{\mathsf{Pref}} \leftarrow \mathsf{PerfectRef}(q, \mathcal{T}_\ell)$;
    $Q \leftarrow \mathsf{Mref}(Q_{\mathsf{Pref}}, M_\ell, \mathcal{O}_\ell)$;
    return Q
end

Fig. 3. Algorithm computeWTA

In Fig. 3, we define the algorithm computeWTA. The algorithm makes use of two main procedures: the first one, called PerfectRef, reformulates the query in accordance with the local TBox $\mathcal{T}_\ell$, whereas the second procedure, called Mref, is concerned with the reformulation based on the mapping.

The algorithm PerfectRef is the query rewriting algorithm for *DL-Lite*$_\mathcal{R}$ defined in [16,18,43]. Intuitively, it compiles the knowledge of the local TBox $\mathcal{T}_\ell$ needed to answer the input query q into a set of conjunctive queries over $\mathcal{T}_\ell$.

Example 3. Continuing Example 2, consider the query

$$q_0 = \{y \mid \exists x. \mathsf{Manager}(x) \land \mathsf{member}(x, y)\}$$

that is specified over the local peer P_ℓ, and execute $\mathsf{computeWTA}(q_0, P_\ell)$. Since the first component of the role director is typed by the concept Manager (assertion $\exists \mathsf{director} \sqsubseteq \mathsf{Manager}$ in $\mathcal{T}_\ell$), the algorithm rewrites the first atom of q_0 and produces the query $q_1 = \{y \mid \exists x. \mathsf{director}(x, \_) \land \mathsf{member}(x, y)\}$[6]. Since the role director is subsumed by the role member (assertion $\mathsf{director} \sqsubseteq \mathsf{member}$ in $\mathcal{T}_\ell$), the algorithm rewrites the second atom of q_1 and produces the query $q_2 = \{y \mid \exists x. \mathsf{director}(x, \_) \land \mathsf{director}(x, y)\}$. It is not possible to directly rewrite the query

[5] For an analysis on the inconsistency problem in the context of database and ontology integration see, for example, [9,11,36,47].

[6] We use the symbol '_' to denote non-shared variables that are existentially quantified.

q_2 by exploiting the TBox assertions. However, the two atoms in q_2 unify, and hence PerfectRef "reduces" q_2, thus producing the query $q_3 = \{y \mid \mathsf{director}(\_, y)\}$. Actually, the reduction transforms the bound variable x of q_2 in an unbound variable in q_3. Therefore, the algorithm can now rewrite q_3 by means of the assertion $\mathsf{Dept} \sqsubseteq \exists\mathsf{director}^-$, and produces the query $q_4 = \{y \mid \mathsf{Dept}(y)\}$. Then, by the TBox assertion $\exists\mathsf{member}^- \sqsubseteq \mathsf{Dept}$, the algorithm produces $q_5 = \{y \mid \mathsf{member}(\_, y)\}$. Notice also that due to the role subsumption assertion in $\mathcal{T}_\ell$, from the query q_0, the algorithm produces also the query $q_6 = \{y \mid \exists x.\mathsf{Manager}(x) \wedge \mathsf{director}(x, y)\}$. The algorithm does not generate other reformulations. ◁

Algorithm $\mathsf{Mref}(Q, M_\ell, \mathcal{O}_\ell)$
Input: set of CQs Q, mapping M_ℓ, local ontology $\mathcal{O}_\ell$
Output: set of CQs Q over P_r
begin
    $Q_{aux} \leftarrow \emptyset$; $Q_{ris} \leftarrow \emptyset$;
    for each $q \in Q$ **do**
        $Q_{aux} = Q_{aux} \cup \mathit{unfold}(q, M_\ell)$
    for each $q \in Q_{aux}$ **do**
        if q is a mixed query
        then $Q_{ris} \leftarrow Q_{ris} \cup R_{ref}(q, \mathcal{O}_\ell)$
        else $Q_{ris} \leftarrow Q_{ris} \cup q$
    return Q_{ris}
end

Fig. 4. Algorithm Mref

We now turn our attention to the algorithm Mref, shown in Fig. 4, which reformulates the queries over the local TBox $\mathcal{T}_\ell$ returned by PerfectRef into a new set of queries specified over the remote peer P_r. To this aim, Mref makes use of two operators, *unfold* and R_{ref}. Informally, the former reformulates a query q that is specified over the local TBox $\mathcal{T}_\ell$ by replacing atoms of q with the queries over the remote peer P_r associated to such atoms by the mapping M_ℓ. The latter operator computes a set of queries specified over the remote peer for each query that is specified over both the local and the remote TBox. Notice that queries of this form cannot be directly evaluated in our framework. In the following, we formally describe the two operators.

Definition 3. Let $P = \langle \mathcal{T}, M \rangle$ be a peer, let $R(z_1, z_2)$ be an atom, and let m be a mapping assertion $q_r \rightsquigarrow q_\ell$ in M such that

$$\begin{aligned} q_\ell &= \{x_1, x_2 \mid R(x_1, x_2)\}, \qquad \text{and} \\ q_r &= \{x_1', x_2' \mid \exists y_1, \ldots, y_m.\phi(x_1', x_2', y_1, \ldots, y_m)\}. \end{aligned}$$

Then $\mathit{unfold}(R(z_1, z_2), m) = \phi(z_1, z_2, y_1, \ldots, y_m)$. Similarly, let $C(z)$ be an atom, and let m be a mapping assertion $q_r \rightsquigarrow q_\ell$ in M such that

$$\begin{aligned} q_\ell &= \{x \mid C(x)\}, \qquad \text{and} \\ q_r &= \{x' \mid \exists y_1, \ldots, y_m.\phi(x', y_1, \ldots, y_m)\}. \end{aligned}$$

Then $unfold(C(z), m) = \phi(z, y_1, \ldots, y_m)$.

If there is no mapping assertion $q_r \rightsquigarrow q_\ell$ in M such that $q_\ell = \{x_1, x_2 \mid R(x_1, x_2)\}$ (resp., $q_\ell = \{x \mid C(x)\}$), then $R(z_1, z_2)$ (resp., $C(z)$) is said to be *non unfoldable* in M, otherwise it is said to be *unfoldable* in M. ◁

The above notion is extended below to unfolding of conjunctive queries. The following definition generalizes the well-known concept of query unfolding [48].

Definition 4. Let $P = \langle \mathcal{T}, M \rangle$ be a peer, and let $q = \{z_1, \ldots, z_n \mid \phi(z_1, \ldots, z_n)\}$ be a conjunctive query specified over P. The *unfolding of* q *w.r.t.* M is the set of conjunctive queries $unfold(q, M)$ defined as follows:

$$\begin{array}{l} unfold(q, M) = \{ \\ \quad \{z_1, \ldots, z_n \mid unfold(g_1, m_1) \wedge \cdots \wedge unfold(g_h, m_h) \wedge g_{h+1} \wedge \cdots \wedge g_k\} \ \mid \\ \quad m_1, \ldots, m_h \in M,\ \{g_1, \ldots, g_h\} \text{ is a non-empty subset of the unfoldable} \\ \quad\ \text{atoms of q, and } g_{h+1}, \ldots, g_k \text{ are the remaining atoms of } q\}. \end{array}$$ ◁

Note that, if no atom in a query q is unfoldable in the mapping M, then $unfold(q, M) = \emptyset$. Therefore, the unfolding operator produces either CQs completely specified over the alphabet of $\mathcal{T}_r$, and hence specified over P_r, or CQs specified over both the alphabets of $\mathcal{T}_\ell$ and $\mathcal{T}_r$. Such queries are called *mixed queries* (we recall that queries specified over P_ℓ are called local queries, whereas queries specified over P_r are called remote queries). It is easy to see that mixed queries are not queries specified over either the local or remote peer, and therefore there is no means in our framework for directly evaluating them. To solve this problem, the algorithm Mref reformulates each mixed query in a set of remote queries, in such a way that the set of answers to the reformulated queries with respect to an instance for the remote peer $\mathcal{A}_r$, computed by the remote peer, coincides with the set of answers that we would have obtained by directly evaluating mixed queries over $\mathcal{O}_\ell \cup \mathcal{O}_r$, where $\mathcal{O}_r = \langle \mathcal{T}_r, \mathcal{A}_r \rangle$. Since each tuple in the answer to a mixed query is partially supported by extensional assertions provided by the local ontology, the idea at the basis of such a reformulation is to cast into the new remote queries those constants occurring in $\mathcal{O}_\ell$ that support the answers to the mixed query. Such a mechanism is realized trough the operator R_{ref}, formally described below.

Definition 5. Let $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$ be a local peer, $P_r^S = \langle \mathcal{T}_r, \emptyset \rangle$ a remote peer specification, and let

$$q = \{x_1, \ldots, x_n, y_1, \ldots, y_m \mid \exists z_1, \ldots, z_i, w_1, \ldots, w_j . g_\ell^1 \wedge \cdots \wedge g_\ell^h \wedge g_r^1 \wedge \cdots \wedge g_r^k\}$$

be a mixed conjunctive query (i.e., it is such that $h \neq 0$ and $k \neq 0$), where $g_\ell^1, \ldots, g_\ell^h$ are local atoms, $g_r^1, \ldots, g_r^k$ are remote atoms, $x_1, \ldots, x_n$ are the distinguished variables that occur in $g_\ell^1, \ldots, g_\ell^h$ (and possibly also in $g_r^1, \ldots, g_r^k$), $y_1, \ldots, y_m$ are the distinguished variables that occur only in $g_r^1, \ldots, g_r^k$, $z_1, \ldots, z_i$ are the non-distinguished variables that occur both in $g_\ell^1, \ldots, g_\ell^h$ and in $g_r^1, \ldots, g_r^k$, and $w_1, \ldots, w_j$ are the remaining non-distinguished variables of q.

Then, the *remote reformulation of* q w.r.t. $\mathcal{O}_\ell$ is the set $R_{ref}(q, \mathcal{O}_\ell)$ of conjunctive queries specified over P_r defined as follows:

$$\begin{aligned} R_{ref}(q,\mathcal{O}_\ell) = \{\, & \{d_1,\dots,d_n,y_1,\dots,y_m \mid \exists w_1,\dots,w_j.\sigma(g_r^1)\wedge\cdots\wedge\sigma(g_r^k)\} \mid \\ & \langle d_1,\dots,d_n,c_1,\dots,c_i\rangle \in \\ & \quad cert(\{x_1,\dots,x_n,z_1,\dots,z_i \mid \exists w_1,\dots,w_j.g_\ell^1\wedge\cdots\wedge g_\ell^h\},\mathcal{O}_\ell), \\ & \text{and } \sigma = \{x_1\to d_1,\dots,x_n\to d_n, z_1\to c_1,\dots,z_i\to c_i\}\}. \end{aligned}$$

◁

Roughly speaking, R_{ref} first computes "local answers" to the mixed query q "projected" on its local component, selecting as distinguished variables $z_1,\dots,z_i$, i.e., the variables that are non-distinguished in q and that also occur in the remote component of the query. Then, for each computed tuple t, R_{ref} constructs a new remote query by projecting the body of q on its remote component, and substituting $z_1,\dots,z_i$ and $x_1,\dots,x_n$ with the corresponding constants in t (notice that in such a way the remote peer receives through the reformulated query those extensional information of the local ontology which is needed to answer the mixed query). Obviously, if no local answers to the mixed query exists, the remote reformulation of q is empty.

Example 4. We continue Example 2. The procedure Mref is executed with the set $Q = \{q_0,q_1,q_2,q_3,q_4,q_5,q_6\}$ as input. Let's focus on the query q_0. It is unfolded in the remote query $\{y \mid \exists x.\mathsf{ManagerR}(x)\wedge\mathsf{memberR}(x,y)\}$, and in two mixed queries. Since there are no facts in the local peer for the predicate member, the mixed mentioning this predicate can be ignored. We thus consider the mixed query $q_m = \{y \mid \exists x.\mathsf{Manager}(x)\wedge\mathsf{memberR}(x,y)\}$. Since $cert(\{x \mid \mathsf{Manager}(x)\},\mathcal{O}_\ell) = \{\mathtt{Mary}\}$, the remote reformulation of q_m produced by R_{ref} is $\{y \mid \mathsf{memberR}(\mathtt{Mary},y)\}$. We proceed analogously for the other queries in Q[7]. The result returned by computeWTA is the following set of queries:

$$\begin{aligned}
&\{y \mid \exists x.\mathsf{ManagerR}(x)\wedge\mathsf{memberR}(x,y)\},\\
&\{y \mid \mathsf{memberR}(\mathtt{Mary},y)\},\\
&\{y \mid \exists x,z,w.\mathsf{bossR}(x,z)\wedge\mathsf{memberR}(z,w)\wedge\mathsf{memberR}(x,y)\},\\
&\{y \mid \exists x,z.\mathsf{bossR}(x,z)\wedge\mathsf{memberR}(z,y)\},\\
&\{y \mid \mathsf{DeptR}(y)\},\\
&\{y \mid \exists x,z.\mathsf{ManagerR}(x)\wedge\mathsf{bossR}(x,z)\wedge\mathsf{memberR}(z,y)\},\\
&\{y \mid \exists z.\mathsf{bossR}(\mathtt{Mary},z)\wedge\mathsf{memberR}(z,y)\},\\
&\{y \mid \exists x.\mathsf{memberR}(x,y)\}.
\end{aligned}$$

The set of certain answers returned by the remote peer is then $\{\mathtt{D3},\mathtt{D2}\}$. Furthermore, the set of certain answers to q_0 computed by the local peer is $\{\mathtt{D1}\}$. It is easy to see that the union of the above sets is exactly the set that we would have obtained by computing $cert(q_0,\mathcal{O}_\ell\cup M_\ell\cup\mathcal{O}_r)$, i.e., the algorithm computeWTA returned a solution to the What-To-Ask problem for our ongoing example. ◁

[7] We do not reformulate q_2 since it is contained in q_3.

As for the correctness of our technique, it is possible to show that the algorithm computeWTA provides a solution to the What-To-Ask problem in our specialized setting, based on the following properties:

(i) From a client's query q over the local ontology $\mathcal{O}_\ell$, the algorithm PerfectRef is able to compute a set of CQs over $\mathcal{O}_\ell$ that can be evaluated in order to provide the certain answers to q, without taking into account the TBox of $\mathcal{O}_\ell$.
(ii) The unfolding operator used in the algorithm Mref allows us to obtain, from a query q specified over the local peer P_ℓ, a set of CQs over $\mathcal{O}_\ell$ and $\mathcal{O}_r$, which can be evaluated in order to compute the certain answers to q, without taking into account the mapping $\mathcal{M}_\ell$.
(iii) In order to compute the certain answers to a mixed CQ q, i.e., referring to at least one predicate of $\mathcal{O}_\ell$ and one predicate of $\mathcal{O}_r$, we can resort to the remote reformulation of q which produces only queries over the remote ontology $\mathcal{O}_r$.

Theorem 1. *Let $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$ be a local peer such that $\mathcal{O}_\ell$ is a DL-Lite$_\mathcal{R}$ ontology, let $P_r^S = \langle \mathcal{T}_r, \emptyset \rangle$ be a remote peer specification such that $\mathcal{T}_r$ is a DL-Lite$_\mathcal{R}$ TBox, and let q be a CQ over P_ℓ. Then,* computeWTA(q, P_ℓ) *returns a solution for WTA(q, P_ℓ, P_r^S).* $\lhd$

Next, we turn to computational complexity of the algorithm and provide the following result, which follows from the fact that the algorithm PerfectRef runs in polynomial time with respect to the size of the input TBox $\mathcal{T}_\ell$ [16], and from the fact that Mref runs in polynomial time with respect to the size of $\mathcal{O}_\ell$.

Theorem 2. *Let $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$ be a local peer such that $\mathcal{O}_\ell$ is a DL-Lite$_\mathcal{R}$ ontology, let $P_r^S = \langle \mathcal{T}_r, \emptyset \rangle$ be a remote peer specification such that $\mathcal{T}_r$ is a DL-Lite$_\mathcal{R}$ TBox, and let q be a CQ over P_ℓ. Then, the computational complexity of* computeWTA(q, P_ℓ) *is polynomial in the size of $\mathcal{O}_\ell$ and M_ℓ.* $\lhd$

We point out that, in general, the size of the set of queries generated by computeWTA may be exponential in the size of the initial query, which obviously implies that the algorithm runs in exponential time in the query size. However, since typically the input query size can be assumed to be small, this exponential blow-up is not likely to be a problem in practice.

5 What-To-Ask Problem: Negative Result

In this section we consider peers equipped with ontologies specified in *DL-Lite*$_\mathcal{F}$, the other basic language of the *DL-Lite* family, which does not admit role inclusions, as in *DL-Lite*$_\mathcal{R}$, but allows for functionalities on roles, without any restriction (cf. Sect. 2). Interestingly, despite the fact that, as in *DL-Lite*$_\mathcal{R}$, conjunctive query answering in *DL-Lite*$_\mathcal{F}$ can be solved through query rewriting into a set of conjunctive queries (cf. [16]), the What-To-Ask problem in this case may not admit a solution.

To prove this result, we first provide a complexity lower bound for the problem of instance checking in our framework when both the remote and local peer hosts ontologies specified in *DL-Lite*$_{\mathcal{F}}$.

Theorem 3. *The instance checking (and thus query answering) problem in an ontology-to-ontology system $\langle P_\ell, P_r\rangle$ where the ontologies of both P_ℓ and P_r are expressed in DL-Lite$_{\mathcal{F}}$ is* NLogSpace*-hard in data complexity.* ◁

Proof. We prove this result by a reduction from reachability in directed graphs.

Let $G = (N, E)$ be a directed graph, where N is the set of its nodes and E is the set of its edges, i.e., pairs (n_i, n_j) such that n_i and n_j belongs to N. We consider the problem of verifying whether a node $d \in N$ is reachable from a node $s \in N$. We define the remote peer $\mathcal{P}_r = \langle \mathcal{O}_r, \emptyset\rangle$, where $\mathcal{O}_r = \langle \mathcal{T}_r, \mathcal{A}_r\rangle$, as follows:

– the alphabet of the predicates of P_r contains the atomic concept A, the atomic role P, and the atomic role $\hat{P}$, and $\mathcal{T}_r$ consists of the inclusion assertions

$$A \sqsubseteq \exists P \qquad\qquad \exists P^- \sqsubseteq A$$

– the ABox $\mathcal{A}_r$ is the set of facts

$$\{A(s)\} \cup \{\hat{P}(n_i, n_j) \mid (n_i, n_j) \in E \text{ is an edge of } G\}$$

We then construct the local peer $\mathcal{P}_\ell = \langle \mathcal{O}_\ell, M_\ell\rangle$, with $\mathcal{O}_\ell = \langle \mathcal{T}_\ell, \mathcal{A}_\ell\rangle$, as follows:

– the alphabet of the predicates of P_ℓ consists of the atomic concept C and the atomic role Q, and the TBox $\mathcal{T}_\ell$ contains the assertion

$$(\mathsf{funct}\ Q)$$

– the ABox $\mathcal{A}_\ell$ is empty;
– the mapping M_ℓ contains the following assertions

$$\begin{array}{c}\{x, y \mid P(x, y)\} \rightsquigarrow \{x, y \mid Q(x, y)\}\\ \{x, y \mid \hat{P}(x, y)\} \rightsquigarrow \{x, y \mid Q(x, y)\}\\ \{x \mid A(x)\} \rightsquigarrow \{x \mid C(x)\}\end{array}$$

It is then easy to see that there is a path in G from s to d if and only if $d \in \mathit{cert}(q, \mathcal{O}_\ell \cup \mathcal{M}_\ell \cup \mathcal{O}_r)$, where $q = \{x \mid C(x)\}$. □

From the complexity characterization given above, it follows that peer query answering in the setting considered requires at least the power of linear recursive Datalog (NLogSpace). The following result is therefore a straightforward consequence of Theorem 3.

Theorem 4. *There exists a local peer $P_\ell = \langle \mathcal{O}_\ell, M_\ell\rangle$, where $\mathcal{O}_\ell$ is a DL-Lite$_{\mathcal{F}}$ ontology, a remote peer specification $P_r^S = \langle \mathcal{T}_r, \emptyset\rangle$, where $\mathcal{T}_r$ is a DL-Lite$_{\mathcal{F}}$ TBox, and a CQ q (in fact an instance query) specified over P_ℓ such that $\mathit{WTA}(q, P_\ell, P_r^S)$ has no solution.* ◁

We finally remark that for *DL-Lite*$_{\mathcal{F}}$ peers we miss the property that a solution to the What-To-Ask problem exists even if we empower the local peer with the ability of combining the certain answers from the remote peer through FOL rather than simply union, since query answering in this setting requires to go beyond a FOL processing of the data.

6 Towards a Different Semantic Interpretation of Peer Mappings

We have seen above that the What-To-Ask problem admits solutions for two *DL-Lite*$_{\mathcal{R}}$ ontology-based peers where the local ontology contains mappings towards the remote ontology, but not vice-versa. In fact, it is immediate to extend this result to any number of remote ontologies as long as this hierarchical topology on the mapping is maintained, i.e., the remote ontologies contain no mappings between them nor towards the local ontology. Instead, if we allow for a network of peers with arbitrary topology of the mappings, even for ontologies with no TBox, peer query answering becomes undecidable [19,26]. On the other hand, we have just shown above that even if we maintain a hierarchical structure of the mapping, but include functionalities, in fact replacing *DL-Lite*$_{\mathcal{R}}$ with *DL-Lite*$_{\mathcal{F}}$, the What-To-Ask problem becomes unsolvable even if we allow for arbitrary FOL combinations of the certain answers returned by the remote peer.

These results together question the use of first-order mappings, i.e., mappings whose interpretation is an implication between FOL formulas, typically adopted in data peer frameworks [8,19,26].

A radical solution to this is adopting an (auto) epistemic view of the mappings, as suggested in [19]. According to this view each peer is seen as an autonomous agent that interacts with other autonomous agents through peer mappings, and the entire network of peers is not interpreted as a single first-order logic theory, obtained as the disjoint union of the various peer theories, but it is rather considered as a set of different modules, each with its own knowledge about the world and about the other peers in the network. We formalize these ideas below.

6.1 The Logic K

We present a logical formalization of a peer-to-peer network of peer-ontologies based on the use of epistemic logic [10,20,22,29]. In particular, we adopt a *multi-modal* epistemic logic, based on the premise that each peer in the system can be seen as a rational agent. More precisely, the formalization we provide is based on $\mathbf{K}$, the multi-modal version of the well-known modal logic of knowledge/belief *K45* [20] (a.k.a. *weak-S5* [29], see also [38]).

The language $\mathcal{L}(\mathbf{K})$ of $\mathbf{K}$ is obtained from first-order logic by adding a set $\mathbf{K_1}, \ldots, \mathbf{K_n}$ of modal operators, for the forming rule: if ϕ is a (possibly open) formula, then also $\mathbf{K_i}\phi$ is so, for $1 \leq i \leq n$ for a fixed n. In $\mathbf{K}$, each modal operator is used to formalize the epistemic state of a different agent. Informally, the formula $\mathbf{K_i}\phi$ should be read as "ϕ is known to hold by the agent i". The semantics of $\mathbf{K}$ is such that what is known by an agent must hold in the real world: in other words, the agent cannot have inaccurate knowledge of what is true, i.e., believe something to be true although in reality it is false. Moreover, $\mathbf{K}$ states that the agent has complete information on what it knows, i.e., if agent i knows ϕ then it knows of knowing ϕ, and if agent i does not know ϕ, then it knows that it does not know ϕ. In other words, the following assertions hold for every $\mathbf{K}$ formula ϕ:

$$
\begin{array}{ll}
\mathbf{K_i}\phi \rightarrow \phi, & \text{known as the axiom schema T} \\
\mathbf{K_i}\phi \rightarrow \mathbf{K_i}(\mathbf{K_i}\phi), & \text{known as the axiom schema 4} \\
\neg\mathbf{K_i}\phi \rightarrow \mathbf{K_i}(\neg\mathbf{K_i}\phi), & \text{known as the axiom schema 5}
\end{array}
$$

To define the semantics of $\mathbf{K}$, we start from first-order interpretations. We restrict our attention to first-order interpretations that share a fixed infinite domain Δ and assume that constants of the set Γ act as standard names for Δ.

Formulas of $\mathbf{K}$ are interpreted over $\mathbf{K}$-structures. A $\mathbf{K}$-*structure* is a Kripke structure E of the form $(W, \{R_1, \dots R_n\}, V)$, where: W is a set whose elements are called *possible worlds*; V is a function assigning to each $w \in W$ a first-order interpretation $V(w)$; and each R_i, called the *accessibility relation* for the modality $\mathbf{K_i}$, is a binary relation over W, with the following constraints:

$$
\begin{array}{l}
\text{if } w \in W \text{ then } (w, w) \in R_i, \text{ i.e., } R_i \text{ is reflexive} \\
\text{if } (w_1, w_2) \in R_i \text{ and } (w_2, w_3) \in R_i \text{ then } (w_1, w_3) \in R_i, \text{ i.e., } R_i \text{ is transitive} \\
\text{if } (w_1, w_2) \in R_i \text{ and } (w_1, w_3) \in R_i \text{ then } (w_2, w_3) \in R_i, \text{ i.e., } R_i \text{ is euclidean.}
\end{array}
$$

An $\mathbf{K}$-*interpretation* is a pair E, w, where $E = (W, \{R_1, \dots R_n\}, V)$ is an $\mathbf{K}$-structure, and w is a world in W. We inductively define when a sentence (i.e., a closed formula) ϕ *is true in an interpretation* E, w (or, is true on world $w \in W$ in E), written $E, w \models \phi$, as follows:[8]

$$
\begin{array}{lll}
E, w \models P(c_1, \dots, c_n) & \text{iff} & V(w) \models P(c_1, \dots, c_n) \\
E, w \models \phi_1 \wedge \phi_2 & \text{iff} & E, w \models \phi_1 \text{ and } E, w \models \phi_2 \\
E, w \models \neg\phi & \text{iff} & E, w \not\models \phi \\
E, w \models \exists x.\psi & \text{iff} & E, w \models \psi^x_c \text{ for some constant } c \\
E, w \models \mathbf{K_i}\phi & \text{iff} & E, w' \models \phi \text{ for every } w' \text{ such that } (w, w') \in R_i
\end{array}
$$

We say that a sentence ϕ is *satisfiable* if there exists an $\mathbf{K}$-*model* for ϕ, i.e., an $\mathbf{K}$-interpretation E, w such that $E, w \models \phi$, *unsatisfiable* otherwise. A *model* for a set Σ of sentences is a model for every sentence in Σ. A sentence ϕ is *logically implied* by a set Σ of sentences, written $\Sigma \models_{\mathbf{K}} \phi$, if and only if in every $\mathbf{K}$-model E, w of Σ, we have that $E, w \models \phi$.

Notice that, since each accessibility relation of a $\mathbf{K}$-structure is reflexive, transitive and Euclidean, all instances of axiom schemas T, 4 and 5 are satisfied in every $\mathbf{K}$-interpretation.

[8] We use ψ^x_c to denote the formula obtained from ψ by substituting each free occurrence of the variable x with the constant c.

6.2 The What-To-Ask Problem Under the Epistemic Semantics

Due to the characteristics mentioned above, see also [15], **K** is well-suited to formalize mappings between peers. We recall that an ontology-based peer ontology P_i has the form $P_i = \langle \mathcal{O}_i, M_i \rangle$, where $\mathcal{O}_i$ is an ontology, and M_i is a set of peer mapping assertions of the form (cf. Sect. 3.1)

$$\{x \mid \exists \boldsymbol{y}. conj(x, \boldsymbol{y})\} \leadsto \{x \mid C(x)\} \text{ or}$$
$$\{x_1, x_2 \mid \exists \boldsymbol{y}. conj(x_1, x_2, \boldsymbol{y})\} \leadsto \{x_1, x_2 \mid R(x_1, x_2)\},$$

where $conj(x, \boldsymbol{y})$ and $conj(x_1, x_2, \boldsymbol{y})$ are specified over another peer P_j.

For a peer P_i, we define the theory $\mathcal{T}_K(P_i)$ in **K** as the union of the following sentences:

– Ontology $\mathcal{O}_i$ of P_i: for each sentence ϕ in $\mathcal{O}_i$, we have

$$\mathbf{K_i}\phi$$

Observe that ϕ is a first-order sentence expressed in the alphabet of P_i, which is disjoint from the alphabet of all the other peers.

– peer mapping assertions M_i: for each peer mapping assertion from peer P_j to peer P_i in M, we have

$$\forall x. \mathbf{K_j}(\exists \boldsymbol{y}. conj(x, \boldsymbol{y})) \rightarrow \mathbf{K_i}(C(x))$$
$$\forall x_1, x_2. \mathbf{K_j}(\exists \boldsymbol{y}. conj(x_1, x_2, \boldsymbol{y})) \rightarrow \mathbf{K_i}(R(x_1, x_2)).$$

In words, the first sentence specifies the following rule: for each object a, if peer P_j knows the sentence $\exists \boldsymbol{y}. conj(a, \boldsymbol{y})$, then peer P_i knows the assertion $C(a)$. Similarly, the second sentence specifies that for each pair of objects a, b, if peer P_j knows the sentence $\exists \boldsymbol{y}. conj(a, b, \boldsymbol{y})$, then peer P_i knows the assertion $R(a, b)$.

Given a network of peer-ontologies $\mathcal{P} = \{P_1, \ldots, P_n\}$, we denote by $\mathcal{T}_K(\mathcal{P})$ the theory corresponding to the network of peer-ontologies $\mathcal{P}$, i.e., $\mathcal{T}_K(\mathcal{P}) = \bigcup_{i=1,\ldots,n} \mathcal{T}_K(P_i)$.

The semantics of a (conjunctive) query q posed to a peer $P_i = \langle O_i, M_i \rangle$ of $\mathcal{P}$ is defined as the set of tuples

$$cert_{\mathbf{K}}(q, P_i, \mathcal{P}) = \{\boldsymbol{t} \mid \mathcal{T}_K(\mathcal{P}) \models_{\mathbf{K}} \mathbf{K_i} q(\boldsymbol{t})\}$$

where $q(\boldsymbol{t})$ denotes the sentence obtained from the open formula $q(\boldsymbol{x})$ by replacing all occurrences of the free variables in $\boldsymbol{x}$ with the corresponding constants in $\boldsymbol{t}$.

Let us now turn our attention to ontology-to-ontology systems of the form defined in Sect. 3.1. It is immediate to apply the epistemic-based interpretation given above to systems of this kind, which contain only a remote peer and a local peer. Then, we can rephrase the What-To-ask problem under the epistemic semantics as follows.

Definition 6. Let $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$ be a local peer, $P_r^S = \langle \mathcal{T}_r, \emptyset \rangle$ a remote peer specification, and q a client's query specified over P_ℓ. The *What-To-Ask* problem under the epistemic interpretation of peer mappings, $WTA_e(q, P_\ell, P_r^S)$, is defined as follows: Given as input q, P_ℓ, and P_r^S, find a finite set $\{q_r^1, \ldots, q_r^n\}$ of queries, each specified over the remote peer P_r, such that for every instance $\mathcal{A}_r$ of the remote peer:

$$cert_{\mathbf{K}}(q, P_\ell, \mathcal{P}) = cert(q, \mathcal{O}_\ell) \cup cert(q_r^1, \mathcal{O}_r) \cup \ldots \cup cert(q_r^n, \mathcal{O}_r)$$

where $\mathcal{O}_r = \langle \mathcal{T}_r, \mathcal{A}_r \rangle$ and $\mathcal{P} = \{P_\ell, P_r\}$, with $P_r = \langle \mathcal{O}_r, \emptyset \rangle$. $\triangleleft$

Notably, it is possible to show that under this interpretation of the system, the What-To-Ask problem admits solutions when ontologies are specified in *DL-Lite*$_\mathcal{A}$ [43], which is the logic combining the features of both *DL-Lite*$_\mathcal{R}$, and *DL-Lite*$_\mathcal{F}$, but where the functionality axiom can be asserted only on roles that have no specializations.

Theorem 5. *Let $P_\ell = \langle \mathcal{O}_\ell, M_\ell \rangle$ be a local peer, such that $\mathcal{O}_\ell$ is a DL-Lite$_\mathcal{A}$ ontology, let $P_r^S = \langle \mathcal{T}_r, \emptyset \rangle$ be a remote peer specification, such that $\mathcal{T}_r$ is a DL-Lite$_\mathcal{A}$ TBox, and let q be a CQ specified over P_ℓ. Then,* $\mathsf{computeWTA}(q, P_\ell)$ *returns a solution for $WTA_e(q, P_\ell, P_r)$.* $\triangleleft$

Finally, we point out that when the ability of the local peer of combining certain answers returned by the remote peer goes beyond the simple union, peer query answering can be solved also through mechanisms that are different from the algorithm $\mathsf{computeWTA}$. For example, when the local peer is able to combine tuples coming from the remote peer with local tuples for computing joins in mixed queries, the procedure Mref in the algorithm $\mathsf{computeWTA}$ might be substituted with a more efficient procedure, based for example on the (partial) local materialization of remote data accessible through mapping assertions [34, 35]. Some smart strategies can be adopted in this case to limit materialization only to data relevant for answering the query at hand.

6.3 Epistemic Semantics for Networks of Peer-Ontologies

Interestingly, by virtue of the epistemic interpretation of the peer mappings, techniques for query answering as the one discussed above can be generalized to peer-ontologies networks of arbitrary topology, provided that each peer has the ability of reformulating queries posed over the local ontology in queries to be posed to the other peers in the network (e.g., via the algorithm given in [19] where the external database system can be seen as an autonomous peer in the network). These techniques have been studied in the relational setting in [21].

7 Conclusions

The peer-to-peer paradigm represents an abstraction that captures several types of system studied in different disciplines, such as Multi-agent systems, Semantic

Web, Data Management, Knowledge Representations, and others. In this paper, we have carried out a fundamental study on data-intensive peer-to-peer systems in the case where the whole system is constituted by two peers connected by mappings, and each peer is structured as a knowledge base expressed in a Description Logic of the *DL-Lite* family. In particular, we have addressed the so-called "What-To-Ask" problem, which, given a query q on a local peer P_ℓ, requires to figure out which queries to send to the remote peer in order for P_ℓ to be able to return the correct and complete set of answers to q.

The investigation discussed in this paper can be continued along several interesting directions. In particular, it would be interesting to explore methods for dealing with inconsistencies between peers, a problem that has been ignored by the present paper (see, for instance, [17]). Finally, another relevant problem is to design methods for update propagation between peers, so that all relevant data from the remote peers can be stored in the local peer, thus avoiding asking queries at run time.

References

1. Aberer, K.: Peer-to-Peer Data Management. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, San Rafael (2011). https://doi.org/10.2200/S00338ED1V01Y201104DTM015
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publishing Co., Boston (1995)
3. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Distributed reasoning in a peer-to-peer setting: application to the Semantic Web. J. Artif. Intell. Res. **25**, 269–314 (2006)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI), pp. 364–369 (2005)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: Clark, K., Patel-Schneider, P.F. (eds.) Proceedings of the 4th International Workshop on OWL: Experiences and Directions (OWLED DC) (2008)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
7. Baader, F., Marantidis, P., Pensel, M.: The data complexity of answering instance queries in $\mathcal{FL}_0$. In: Proceedings of the 27th International World Wide Web Conferences (WWW), pp. 1603–1607 (2018)
8. Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I.: Data management for peer-to-peer computing: a vision. In: Proceedings of the 5th International Workshop on the Web and Databases (WebDB) (2002)
9. Bienvenu, M., Bourgaux, C., Goasdoué, F.: Computing and explaining query answers over inconsistent DL-Lite knowledge bases. J. Artif. Intell. Res. **64**, 563–644 (2019). https://doi.org/10.1613/jair.1.11395
10. Blackburn, P., van Benthem, J.F.A.K., Wolter, F.: Handbook of Modal Logic. Elsevier, New York (2006)

11. Bravo, L., Bertossi, L.: Disjunctive deductive databases for computing certain and consistent answers to queries from mediated data integration systems. J. Appl. Logic **3**(2), 329–367 (2005). Special Issue on Logic-based Methods for Information Integration
12. Calvanese, D., Damaggio, E., De Giacomo, G., Lenzerini, M., Rosati, R.: Semantic data integration in P2P systems. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) DBISP2P 2003. LNCS, vol. 2944, pp. 77–90. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24629-9_7
13. Calvanese, D., et al.: Ontologies and databases: the *DL-Lite* approach. In: Tessaris, S., et al. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03754-2_7
14. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: What to ask to a peer: ontology-based query reformulation. In: Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR), pp. 469–478 (2004)
15. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: effective first-order query processing in description logics. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), pp. 274–279 (2007)
16. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the *DL-Lite* family. J. Autom. Reason. **39**(3), 385–429 (2007)
17. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Inconsistency tolerance in P2P data integration: an epistemic logic approach. Inf. Syst. **33**(4–5), 360–384 (2008)
18. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artif. Intell. **195**, 335–360 (2013). https://doi.org/10.1016/j.artint.2012.10.003
19. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Logical foundations of peer-to-peer data integration. In: Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS), pp. 241–251 (2004)
20. Chellas, B.F.: Modal Logic: An introduction. Cambridge University Press, Cambridge (1980)
21. De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: On reconciling data exchange, data integration, and peer data management. In: Proceedings of the 26th ACM Symposium on Principles of Database Systems (PODS), pp. 133–142 (2007)
22. van Ditmarsch, H., Halpern, J.Y., van der Hoek, W., Kooi, B. (eds.): Handbook of Epistemic Logic. College Publications, Kolkata (2015)
23. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing schema mappings: second-order dependencies to the rescue. In: Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS) (2004)
24. Fuxman, A., Kolaitis, P.G., Miller, R., Tan, W.C.: Peer data exchange. In: Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS), pp. 160–171 (2005)
25. Ghidini, C., Serafini, L.: Distributed first order logic. Artif. Intell. **253**, 1–39 (2017). https://doi.org/10.1016/j.artint.2017.08.008
26. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE), pp. 505–516 (2003)

27. Halevy, A.Y.: Theory of answering queries using views. SIGMOD Rec. **29**(4), 40–47 (2000)
28. Halevy, A.Y.: Answering queries using views: a survey. Very Large Database J. **10**(4), 270–294 (2001)
29. Hughes, G.E., Cresswell, M.J.: A Companion to Modal Logic. Methuen, London (1984)
30. Hull, R., Benedikt, M., Christophides, V., Su, J.: E-services: a look behind the curtain. In: Proceedings of the 22nd ACM Symposium on Principles of Database Systems (PODS), pp. 1–14. ACM Press and Addison Wesley (2003). https://doi.org/10.1145/773153.773154
31. Ives, Z.G.: Updates and transactions in peer-to-peer systems. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, 2nd edn. Springer, New York (2018). https://doi.org/10.1007/978-1-4614-8265-9_1222
32. Karvounarakis, G., Green, T.J., Ives, Z.G., Tannen, V.: Collaborative data sharing via update exchange and provenance. ACM Trans. Database Syst. **38**(3), 19:1–19:42 (2013). https://doi.org/10.1145/2500127
33. Kolaitis, P.G., Pichler, R., Sallinger, E., Savenkov, V.: Limits of schema mappings. Theory Comput. Syst. **62**(4), 899–940 (2018). https://doi.org/10.1007/s00224-017-9812-7
34. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in *DL-Lite*. In: Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning (KR), pp. 247–257 (2010)
35. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), pp. 2656–2661 (2011)
36. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant query answering in ontology-based data access. J. Web Semant. **33**, 3–29 (2015). https://doi.org/10.1016/j.websem.2015.04.002
37. Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS), pp. 233–246 (2002). https://doi.org/10.1145/543613.543644
38. Levesque, H.J., Lakemeyer, G.: The Logic of Knowledge Bases. The MIT Press, Cambridge (2001)
39. Madhavan, J., Halevy, A.Y.: Composing mappings among data sources. In: Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), pp. 572–583 (2003)
40. Mattos, N.M.: Integrating information for on demand computing. In: Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), pp. 8–14 (2003)
41. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), pp. 1039–1044. IJCAI/AAAI (2011)
42. Papazoglou, M.P., Krämer, B.J., Yang, J.: Leveraging web-services and peer-to-peer networks. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 485–501. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45017-3_33
43. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semant. **10**, 133–173 (2008). https://doi.org/10.1007/978-3-540-77688-8_5

44. Roth, A., Skritek, S.: Peer data management. In: Data Exchange, Integration, and Streams, Dagstuhl Follow-Ups, vol. 5, pp. 185–215. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2013). https://doi.org/10.4230/DFU.Vol5.10452.185
45. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison Wesley Publishing Co., Boston (1998)
46. Serafini, L., Ghidini, C.: Using wrapper agents to answer queries in distributed information systems. In: Yakhno, T. (ed.) ADVIS 2000. LNCS, vol. 1909, pp. 331–340. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-40888-6_32
47. Staworko, S., Chomicki, J., Marcinkowski, J.: Prioritized repairing and consistent query answering in relational databases. Ann. Math. Artif. Intell. **64**(2–3), 209–246 (2012). https://doi.org/10.1007/s10472-012-9288-8
48. Ullman, J.D.: Information integration using logical views. In: Afrati, F., Kolaitis, P. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 19–40. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62222-5_34