Logical Foundations of Peer-To-Peer Data Integration

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

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

ABSTRACT

In peer-to-peer data integration, each peer exports data in terms of its own schema, and data interoperation is achieved by means of mappings among the peer schemas. Peers are autonomous systems and mappings are dynamically created and changed. One of the challenges in these systems is answering queries posed to one peer taking into account the mappings. Obviously, query answering strongly depends on the semantics of the overall system. In this paper, we compare the commonly adopted approach of interpreting peer-to-peer systems using a first-order semantics, with an alternative approach based on epistemic logic. We consider several central properties of peer-to-peer systems: modularity, generality, and decidability. We argue that the approach based on epistemic logic is superior with respect to all the above properties. In particular, we show that, in systems in which peers have decidable schemas and conjunctive mappings, but are arbitrarily interconnected, the first-order approach may lead to undecidability of query answering, while the epistemic approach always preserves decidability. This is a fundamental property, since the actual interconnections among peers are not under the control of any actor in the system.

1. INTRODUCTION

In 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 [**?**], the Semantic Web [**?**], Peer-to-Peer and Grid computing [**?**, **?**], service oriented computing and distributed agent systems [**?**, **?**]. 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.

PODS 2004, June 14–16, 2004, Paris, France.

Here, we study data integration in such Peer-to-Peer (P2P) systems. Each peer in the system provides part of the overall information available from a distributed environment, without relying on a single global view, and acts both as a client and as a server in the system. Moreover, the various nodes adopt a suitable infrastructure for managing information. The P2P paradigm was made popular by Napster, which employed a centralized database with references to the information items (files) on the peers. Gnutella, another well-known P2P system, has no central database, and is based on a communication-intensive search mechanism. More recently, a Gnutella-compatible P2P system, called Gridella [**?**], has been proposed, which follows the so-called Peer-Grid (P-Grid) approach. A P-Grid is a virtual binary tree that distributes replication over community of peers and supports efficient search. P-Grid's search structure is completely decentralized, supports local interactions between peers, uses randomized algorithms for access and search, and ensures robustness of search against node failures.

As pointed out in [**?**], current P2P systems focus strictly on handling semantic-free, large-granularity requests for objects by identifier, which both limits their utility and restricts the techniques that might be employed to distribute the data. These current sharing systems are largely limited to applications in which objects are described by their name, and exhibit strong limitations in establishing complex links between peers. To overcome these limitations, data-oriented approaches to P2P have been proposed recently [**?**, **?**, **?**]. For example, in the Piazza system [**?**], data origins serve original content, peer nodes cooperate to store materialized views and answer queries, nodes are connected by bandwidth-constrained links and advertise their materialized views to share resources with other peers.

Differently from the traditional setting, integration in data-oriented P2P systems is not based on a global schema. Instead, each peer represents an autonomous information system, and information integration is achieved by establishing P2P mappings, i.e., mappings among the various peers. Queries are posed to one peer, and the role of query processing is to exploit both the data that are internal to the peer, and the mappings with other peers in the system. To stress the data-oriented nature of the framework, we assume that the various peers export data in terms of a suitable schema, and mappings are established among such peer schemas. A peer schema is therefore intended to export the semantics of information as viewed from the peer.

One of the main issue in formalizing data oriented P2P systems is the semantic characterization of P2P mappings. In this paper, we argue that, although correct from a formal point of view, the usual approach of resorting to a first-order logic interpretation of P2P mappings (followed e.g. by [**?**, **?**, **?**]), has several drawbacks, both from the modeling and from the computational perspective. In particular we analyze three central desirable properties of P2P systems:

- *Modularity*: i.e., how autonomous are the various peers in a P2P system with respect to the semantics. Indeed, since each peer is autonomously built and managed, it should be clearly interpretable both alone and when involved in interconnections with other peers. In particular, interconnections with other peers should not radically change the interpretation of the concepts expressed in the peer.
- *Generality*: i.e., how free we are in placing connections (P2P mappings) between peers. This is a fundamental property, since actual interconnections among peers are not under the control of any actor in the system.
- *Decidability*: i.e., are sound, complete and terminating query answering mechanisms available? If not, it becomes critical to establish basic quality assurance of the answers returned by the system.

We show that these desirable properties are weakly supported by approaches based directly on FOL semantics. Indeed, such approaches essentially consider the P2P system as a single flat logical theory. As a result, the structure of the system in terms of peers is actually lost and remote interconnections may propagate constraints that have a deep impact on the semantics of a peer (see Section 4). Moreover, under arbitrary P2P interconnections, query answering under the first-order semantics is undecidable, even when the single peers have an extremely restricted structure. Motivated by these observations, several authors proposed suitable limitations to the form of P2P mappings, such as acyclicity, thus giving up generality to retain decidability [**?**, **?**, **?**].

To overcome the above drawbacks, we propose a new semantics for P2P systems, with the following aims:

- We want to take into account that peers in our context are to be considered autonomous sites that exchange information. In other words, peers are modules, and the modular structure of the system should be explicitly reflected in the definition of its semantics.
- We do not want to limit a-priori the topology of the mapping assertions among the peers in the system. In particular, we do not want to impose acyclicity of assertions.
- We seek for a semantic characterization that leads to a setting where query answering is decidable, and possibly, polynomially tractable.

We base our proposal of a new semantics for P2P systems on epistemic logic, and we show that the new semantics is clearly superior to the usual FOL semantics with respect to all three properties mentioned above. In particular, for fairly general P2P systems, we devise a top-down query answering algorithm that is based on a recursive (Datalog) reformulation of the query posed to one of the peer of the P2P system, and that is polynomial time with respect to the size of data stored in the peers. Notably, our technique can be applied every time peers are able to do query answering based on reformulation; this ability is at the base of several recent results on data integration [**?**, **?**, **?**].

The paper is organized as follows. In Section 2 we introduce a framework that captures a very general architecture for P2P systems, and then in Section 3 we define both the first-order and the epistemic semantics of such a framework. in Section **??** we discuss the issues of modularity, generality, and decidability under the two semantics, and in Section **??** we study decidability and complexity of query processing in P2P systems under the epistemic semantics. Finally, we draw some conclusions in Section **??**.

2. FRAMEWORK

In this section, we set up the general framework for peer-to-peer (P2P) systems. We refer to a fixed, infinite, denumerable, set Γ of constants. Such constants are shared by all peers, and are the constants that can appear in the P2P system. Moreover, given a relational alphabet A, we denote with $\mathcal{L}_A$ the set of function-free first-order logic (FOL) formulas whose relation symbols are in A and whose constants are in Γ.

A *conjunctive query* (CQ) of arity n over an alphabet A is written in the form

$$\{\mathbf{x} \mid \exists \mathbf{y}\ body_{cq}(\mathbf{x}, \mathbf{y})\}$$

where $body_{cq}(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms of $\mathcal{L}_A$ involving the free variables (also called the *distinguished* variables of the query) $\mathbf{x} = x_1, \ldots, x_n$, the existentially quantified variables (also called the *non-distinguished* variables of the query) $\mathbf{y} = y_1, \ldots, y_m$, and constants from Γ.

A *P2P system* $\mathcal{P}$ is constituted by a set of peers, each of which includes a set of mappings that specify the semantic relationships with the data exported by other peers. Formally, each peer $P \in \mathcal{P}$ (cf. [**?**]) is defined as a tuple $P = (G, S, L, M)$, where:

- G is the *schema* of P, which is a finite set of formulas of $\mathcal{L}_{A_G}$, where A_G is a relational alphabet (disjoint from the other alphabets in $\mathcal{P}$) called the *alphabet* of P.
- S is the *(local) source schema* of P, that is simply a finite relational alphabet (again disjoint from the other alphabets in $\mathcal{P}$), which is called the *local alphabet* of P.
- L is a set of *(local) mapping assertions* between G and S. Each local mapping assertion is an expression of the form

$$cq_S \rightsquigarrow cq_G$$

where cq_S and cq_G are two conjunctive queries of the same arity, over the source schema S and over the peer schema G, respectively.

- M is a set of *P2P mapping assertions*, each of which is an expression of the form

$$cq' \rightsquigarrow cq$$

The query cq, called the *head* of the assertion, is a conjunctive query over the peer (schema of) P, while the query cq', called the *tail* of the assertion, is a conjunctive query of the same arity as cq, over (the schema of) one of the other peers in $\mathcal{P}$.

Intuitively, the source schema describes the structure of the data sources of the peer (possibly obtained by wrapping physical sources), where the real data managed by the peer are stored, while the peer schema provides a virtual view of the information managed by, and exported by the peer. The local mapping assertions establish the connection between the elements of the source schema and those of the peer schema. In particular, an assertion of the form $cq_S \rightsquigarrow cq_G$ specifies that all the data satisfying the query cq_S over the sources also satisfy the concept in the peer schema represented by the query cq_G. This form of mapping is one of the most expressive among those studied in the data integration literature. Indeed, in terms of the terminology used in data integration, except for the P2P mapping assertions, a peer in our setting corresponds to a GLAV *data integration system* [**?**] managing a set of sound data sources S defined in terms of a (virtual) global schema G.[1] Finally, a P2P mapping assertion $cq' \rightsquigarrow cq$, where cq is a query over the schema of the peer P, expresses the fact that P can use, besides the data in its local sources, also the data retrieved by cq' from the peer P' over (the schema of) which cq' is expressed. Such data are mapped to the schema of P according to what is specified by the query cq.

Observe that no limitation is imposed on the topology of the whole set of P2P mapping assertions in the peer system $\mathcal{P}$, and hence the set of all P2P mappings may be cyclic.

Finally, we assume that *queries* that are posed to the P2P system $\mathcal{P}$ are in fact posed to one of the peers P of $\mathcal{P}$. Such queries are expressed in a certain relational query language $\mathcal{L}_P$ (e.g., conjunctive queries, or Datalog – see later) over the schema of P. For now, we make no specific assumption on the query language $\mathcal{L}_P$, except that the peer P can indeed process queries belonging to $\mathcal{L}_P$, and we say that the queries in $\mathcal{L}_P$ are those *accepted by* P.

3. SEMANTICS

We assume that the peers are interpreted over a fixed infinite domain Δ. We also fix the interpretation of the constants in Γ (cf. previous section) so that: (i) each $c \in \Gamma$ denotes an element $d \in \Delta$; (ii) different constants in Γ denote different elements of Δ; (iii) each element in Δ is denoted by a constant in Γ.[2] It follows that Γ is actually isomorphic to Δ, so that we can use (with some abuse of notation) constants in Γ whenever we want to denote domain elements.

[1]Although we assume sources to be sound, our approach to semantics is perfectly valid if we consider complete and exact sources.

[2]In other words the constants in Γ act as *standard names* [**?**].

3.1 Semantics of one peer

We focus first on the semantics of a single peer $P = (G, S, L, M)$. Let us call *peer theory of* P the FOL theory T_P defined as follows. The alphabet of T_P is obtained as union of the alphabet $\mathcal{A}_G$ of G and the alphabet of the local sources S of P. The axioms of T_P are the formulas in G plus one formula of the form

$$\forall \mathbf{x}\ (\exists \mathbf{y}\ (body_{cq_S}(\mathbf{x}, \mathbf{y})) \supset \exists \mathbf{z}\ body_{cq_G}(\mathbf{x}, \mathbf{z}))$$

for each local mapping assertion $cq_S \rightsquigarrow cq_G$ in L.

Observe that the P2P mapping assertions of P are not considered in T_P, and that T_P is an "open theory", since for the sources in P we only have the schema, S, and not the extension. We call *local source database* for P, a database D for the source schema S, i.e., a finite relational interpretation of the relation symbols in S. An interpretation $\mathcal{I}$ of T_P is a *model of* P *based on* D if it is a model of the FOL theory T_P such that for each relational symbol $s \in S$, we have that $s^{\mathcal{I}} = s^D$.

Finally, consider a query q of arity n, expressed in the query language $\mathcal{L}_P$ accepted by P. Given an interpretation $\mathcal{I}$ of T_P, we denote with $q^{\mathcal{I}}$ the set of n-tuples of constants in Γ obtained by evaluating q in $\mathcal{I}$ (viewed as a database over the relations in G), according to the semantics of $\mathcal{L}_P$. We define the *certain answers* $ans(q, P, D)$ to q (accepted by P) based on a local source database D for P, as the set of tuples $\mathbf{t}$ of constants in Γ such that for all models $\mathcal{I}$ of P based on D, we have that $\mathbf{t} \in q^{\mathcal{I}}$.

We now turn our attention to assigning a semantics to the whole P2P system. We distinguish two different approaches.

3.2 FOL semantics for P2P systems

The first approach we discuss is what we may call the FOL approach, followed by [**?**, **?**, **?**]. In this approach, one associates to a P2P system $\mathcal{P}$ a *single* (open) FOL theory $T_{\mathcal{P}}$, obtained as the disjoint union of the various peer theories (P2P mappings are not considered in $T_{\mathcal{P}}$).

By following the approach used for a single peer, we consider a *source database* $\mathcal{D}$ *for* $\mathcal{P}$, simply as the (disjoint) union of one local source database D for each peer P in $\mathcal{P}$. We call *FOL model of* $T_{\mathcal{P}}$ *based on* $\mathcal{D}$ an interpretation $\mathcal{I}$ of the FOL theory $T_{\mathcal{P}}$ such that for each relational symbol s of the source schemas in the peers of $\mathcal{P}$, we have that $s^{\mathcal{I}} = s^{\mathcal{D}}$. Then we call *FOL model of* P *based on* $\mathcal{D}$ a model $\mathcal{I}$ of $T_{\mathcal{P}}$ based on $\mathcal{D}$ that is also a model of the formula

$$\forall \mathbf{x}\ (\exists \mathbf{y}\ (body_{cq_1}(\mathbf{x}, \mathbf{y})) \supset \exists \mathbf{z}\ body_{cq_2}(\mathbf{x}, \mathbf{z}))$$

for each P2P mapping assertion $cq_1 \rightsquigarrow cq_2$ in the peers of $\mathcal{P}$.

Finally, given a query q over one of the peers P in $\mathcal{P}$ and a source database $\mathcal{D}$ for $\mathcal{P}$, we define the *certain answers* $ans_{fol}(q, P, \mathcal{P}, \mathcal{D})$ to q in $\mathcal{P}$ based on $\mathcal{D}$ under FOL semantics, as the set of tuples $\mathbf{t}$ of constants in Γ such that for every FOL model $\mathcal{I}$ of $\mathcal{P}$ based on $\mathcal{D}$, we have that $\mathbf{t} \in q^{\mathcal{I}}$.

3.3 A new semantics for P2P systems based on epistemic logic

We base our proposal of a new semantics for P2P systems on epistemic logic[3]. We briefly remind the basic notions of epistemic logic [**?**, **?**]. In epistemic logic, the language is the one of FOL, except that, besides the usual atoms, one can use another form of atoms, namely $\mathbf{K}\phi$, where ϕ is again a formula. An *epistemic logic theory* is a set of axioms that are formulas in the language of epistemic logic.

The semantics of an epistemic logic theory is based on the notion of epistemic interpretation. We remind the reader that we are referring to a unique interpretation domain Γ. An *epistemic interpretation* $\mathcal{E}$ is a pair $(\mathcal{I}, \mathcal{W})$, where $\mathcal{W}$ is a set of FOL interpretations, and $\mathcal{I} \in \mathcal{W}$. The notion of satisfaction of a formula in an epistemic interpretation $\mathcal{E} = (\mathcal{I}, \mathcal{W})$ is analogous to the one in FOL, with the provision that the interpretation for the atoms is as follows:

- a FOL formula constituted by an atom $a(\mathbf{x})$ (where $\mathbf{x}$ are the free variables in the formula) is satisfied in $(\mathcal{I}, \mathcal{W})$ by the tuples $\mathbf{t}$ of constants in Γ such that $a(\mathbf{t})$ is true in $\mathcal{I}$,
- an atom of the form $\mathbf{K}\phi(\mathbf{x})$ is satisfied in $(\mathcal{I}, \mathcal{W})$ by the tuples $\mathbf{t}$ of constants in Γ such that $\phi(\mathbf{t})$ is satisfied in all epistemic interpretations $(\mathcal{J}, \mathcal{W})$ with $\mathcal{J} \in \mathcal{W}$.

Note that our definition of epistemic interpretation is a simplified view of a Kripke structure of an S5 modal system, in which every epistemic interpretation is constituted by a set of worlds, each one connected, through the accessibility relation, to all the other ones. Indeed, in our setting each world corresponds to a FOL interpretation, and the accessibility relation is left implicit by viewing the whole structure as a set.

An *epistemic model* of an epistemic logic theory is an epistemic interpretation that satisfies every axiom of the theory. In turn, an *axiom* ϕ is satisfied by an epistemic interpretation $(\mathcal{I}, \mathcal{W})$ if, for every $\mathcal{J} \in \mathcal{W}$, the epistemic interpretation $(\mathcal{J}, \mathcal{W})$ satisfies the *formula* ϕ. Observe that in order for an epistemic interpretation $(\mathcal{I}, \mathcal{W})$ to be a model of a theory, the axioms of the theory are required to be satisfied in every $\mathcal{J} \in \mathcal{W}$. Hence, with regard to the satisfaction of axioms, only $\mathcal{W}$ counts.

Observe that, in epistemic logic, the formula $\mathbf{K}(\phi \vee \psi)$ has an entirely different meaning with respect to the formula $\mathbf{K}\phi \vee \mathbf{K}\psi$. Indeed, the former is satisfied in an interpretation $(\mathcal{J}, \mathcal{W})$ if for every $\mathcal{I} \in \mathcal{W}$, there is at least one among $\{\phi, \psi\}$, that is satisfied in $\mathcal{I}$. Conversely, the latter requires either that ϕ is satisfied in all $\mathcal{I} \in \mathcal{W}$ or that ψ is satisfied in all $\mathcal{I} \in \mathcal{W}$. Observe also that, if ϕ is a FOL formula, there is a striking difference between $\mathbf{K}\exists x\phi(x)$ and $\exists x\mathbf{K}\phi(x)$. In particular, for $\exists x\mathbf{K}\phi(x)$ to be satisfied in $(\mathcal{I}, \mathcal{W})$ there must be a constant $c \in \Gamma$ such that $\phi(c)$ is satisfied in every $\mathcal{J} \in \mathcal{W}$, while for $\mathbf{K}(\exists x\phi(x))$ to be satisfied it is only required that in each $\mathcal{J} \in \mathcal{W}$ there exists a constant $c \in \Gamma$ such that $\phi(c)$ is satisfied in $\mathcal{J}$.

[3]Technically we resort to epistemic FOL with standard names, and therefore with a fixed domain, and rigid interpretation of constants [**?**].

We formalize a P2P system $\mathcal{P}$ in terms of the epistemic logic as follows. First as before we consider the theory $T_{\mathcal{P}}$, obtained as the disjoint union of the various peer theories. To such a theory we add a set of axioms $M_{\mathcal{P}}$ to capture the mapping assertions. $M_{\mathcal{P}}$ is formed by one axiom of the form

$$\forall \mathbf{x}\ (\mathbf{K}(\exists \mathbf{y}\ (body_{cq_1}(\mathbf{x}, \mathbf{y}))) \supset \exists \mathbf{z}\ body_{cq_2}(\mathbf{x}, \mathbf{z}))$$

for each P2P mapping assertion $cq_1 \rightsquigarrow cq_2$ in the peers of $\mathcal{P}$. Note that this formalization of the P2P mapping assertions intuitively reflects the idea that only what is *known* by the peers mentioned in the tail of the assertion is transferred to the peer mentioned in the head.

Let us define the notion of *FOL model of $T_{\mathcal{P}}$ based on a source database $\mathcal{D}$ for $\mathcal{P}$* as in Section 3.2. Then, we call *epistemic model of $\mathcal{P}$ based on $\mathcal{D}$* an epistemic interpretation $(\mathcal{I}, \mathcal{W})$ such that $\mathcal{W}$ is a set of models of $T_{\mathcal{P}}$ based on $\mathcal{D}$, and $(\mathcal{I}, \mathcal{W})$ is an epistemic model of $M_{\mathcal{P}}$. This implies that, for each P2P mapping assertion $cq_1 \rightsquigarrow cq_2$ and for every tuple $\mathbf{t}$ of objects in Γ, the fact that $\exists \mathbf{y}\ body_{cq_1}(\mathbf{t}, \mathbf{y})$ is satisfied in every FOL model in $\mathcal{W}$ implies that also $\exists \mathbf{z}\ body_{cq_2}(\mathbf{t}, \mathbf{z})$ is satisfied in $\mathcal{I}$, and in fact in every FOL model in $\mathcal{W}$ (since formulas in $M_{\mathcal{P}}$ are axioms).

Finally, given a query q over one of the peers P in $\mathcal{P}$ and a source database $\mathcal{D}$ for $\mathcal{P}$, we define the *certain answers* $ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D})$ to q in $\mathcal{P}$ based on $\mathcal{D}$ under the epistemic semantics, as the set of tuples $\mathbf{t}$ of constants in Γ such that for every epistemic model $(\mathcal{I}, \mathcal{W})$ of $\mathcal{P}$ based on $\mathcal{D}$, we have that $\mathbf{t} \in q^{\mathcal{I}}$.

Observe that the epistemic semantics can be considered as a well-behaved, sound approximation of the first-order semantics, since it is immediate to verify that, for each q, $\mathcal{P}$, and $\mathcal{D}$, if $\mathbf{t} \in ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D})$, then $\mathbf{t} \in ans_{fol}(q, \mathcal{P}, \mathcal{D})$.

4. INTERACTIONS BETWEEN MAPPINGS IN P2P SYSTEMS

In this section, we discuss the issue of interaction between various mappings, comparing the epistemic and FOL semantics for P2P systems presented above. The comparison is guided by three principles, namely modularity, generality, and decidability of query answering. To highlight the differences between the two semantics, we will consider the simplest setting in which interactions may occur, namely systems containing only two P2P mappings. The three types of systems we discuss in the following are depicted in Figure **??**(a,b,c), and represent respectively the case of parallel, sequential, and cyclic composition, where each circle represents a peer, and an arrow from a peer P' to a peer P represents a mapping assertion whose head is a CQ over P and whose tail is a CQ over P'. We will also discuss a P2P system that rougthly correspond to the case of data integration (cf. Figure **??**(d).

We first need to provide some definitions. Given a peer $P = (G, S, L, M)$, we denote as $\tau(P)$ the peer (G, S', L', M) such that:

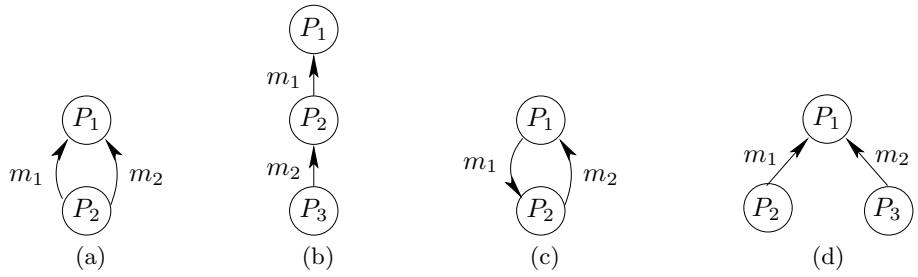

Figure 1: Interactions between two mappings

1. S' is obtained from S by adding a new source predicate symbol r, of the same arity as cq', for each P2P mapping assertion $cq' \rightsquigarrow cq$ in M between a peer P' and P. We also denote as $Q(r)$ the query cq' in the tail of the corresponding P2P mapping assertion, and denote as $P(r)$ the peer P', i.e., the peer over which the query $Q(r)$ is expressed.

2. L' is obtained from L by adding the local mapping assertion $\{\mathbf{x} \mid r(\mathbf{x})\} \rightsquigarrow cq$ for each P2P mapping assertion $cq' \rightsquigarrow cq$ in M.

Furthermore, for a P2P system $\mathcal{P}$, we denote as $\tau(\mathcal{P})$ the P2P system $\{\tau(P) \mid P \in \mathcal{P}\}$. For each peer P, we call *auxiliary alphabet* of P, denoted as $AuxAlph(P)$, the set of new source predicate symbols thus defined. Informally, in each peer the additional sources corresponding to the predicates in the auxiliary alphabet are used to "simulate" the effect of the P2P mapping assertions with respect to contributing to the data of the peer.

4.1 Parallel composition

We consider a P2P system $\mathcal{P}_{par}$ with the structure depicted in Figure **??**(a), and to highlight the interdependence between mappings, we further assume that P_1 does not contain local sources (and local mappings). Hence, $\mathcal{P}_{par}$ is constituted by two peers $P_1 = (G_1, \emptyset, \emptyset, \{m_1, m_2\})$, and $P_2 = (G_2, S_2, L_2, \emptyset)$.

Informally, in the context of parallel composition, we can consider a semantics for P2P systems as modular, if for every query q over P_1, and for every source database D_2 for P_2, the certain answers to q in $\mathcal{P}_{par}$ with respect to D_2 under the considered semantics can be computed by first populating P_1 with the data retrieved by independently applying the two mappings and then evaluating q over such data. Formally, let m_1 be $cq'_1 \rightsquigarrow cq_1$, let m_2 be $cq'_2 \rightsquigarrow cq_2$, and consider the peer $\tau(P_1) = (G_1, \{r_1, r_2\}, \{m'_1, m'_2\}, \{m_1, m_2\})$, where m'_1 is $\{\mathbf{x} \mid r_1(\mathbf{x})\} \rightsquigarrow cq_1$ and m'_2 is $\{\mathbf{x} \mid r_2(\mathbf{x})\} \rightsquigarrow cq_2$. For a local source database D_2 for P_2, let $\delta(P_1, D_2)$ be the local source database for $\tau(P_1)$ such that $r_1^{\delta(P_1,D_2)}$ coincides with the certain answers $ans(cq'_1, P_2, D_2)$ over the single peer P_2, and $r_2^{\delta(P_1,D_2)}$ coincides with the certain answers $ans(cq'_2, P_2, D_2)$ over P_2. Now, semantics X is modular if for every query q to P_1 and for every source database D_2 for P_2, we have that $ans_X(q, P_1, \mathcal{P}, \{D_2\})$ coincides with the certain answers $ans(q, \tau(P_1), \delta(P_1, D_2))$ over $\tau(P_1)$. The following theorems show that a P2P system as simple as $\mathcal{P}_{par}$ is sufficient to separate the epistemic and the FOL semantics with respect to modularity.

THEOREM 4.1. *There is a P2P system $\mathcal{P}_{par} = \{P_1, P_2\}$ of the form as above, a source database D_2 for P_2, and a query q to P_1 such that $ans_{fol}(q, P_1, \mathcal{P}, \{D_2\}) \neq ans(q, \tau(P_1), \delta(P_1, D_2))$.*

Proof (sketch). We exhibit $\mathcal{P}_{par} = \{P_1, P_2\}$, D_2, and q such that the claim holds. Let $P_1 = (\{u/1\}, \emptyset, \emptyset, \{m_1, m_2\})$ and $P_2 = (G_2, \{s/1\}, \{\ell_2\}, \emptyset)$, with $G_2 = \{\forall x\,(u_3(x) \supset u_1(x) \vee u_2(x))\}$, and

$$\begin{array}{rcl} \ell_2 & = & \{x \mid s(x)\} \rightsquigarrow \{x \mid u_3(x)\} \\ m_1 & = & \{x \mid u_1(x)\} \rightsquigarrow \{x \mid u(x)\} \\ m_2 & = & \{x \mid u_2(x)\} \rightsquigarrow \{x \mid u(x)\} \end{array}$$

Consider the source database $D_2 = \{s(a)\}$ for P_2. It is easy to see that for the query $q = \{x \mid u(x)\}$ we have that $ans_{fol}(q, P_1, \mathcal{P}, \{D_2\}) = \{a\}$, while $\delta(P_1, D_2) = \emptyset$, and hence $ans(q, \tau(P_1), \delta(P_1, D_2)) = \emptyset$. □

For the epistemic semantics, from the results in the next section, we get the following theorem.

THEOREM 4.2. *Let $\mathcal{P}_{par}$ and D_2 be as above. Then, for every query q over P_1 we have that $ans_{\mathbf{k}}(q, P_1, \mathcal{P}, \{D_2\}) = ans(q, \tau(P_1), \delta(P_1, D_2))$.*

4.2 Sequential composition

We consider a P2P system $\mathcal{P}_{seq}$ with the structure depicted in Figure **??**(b). Again, to highlight the interaction between the mappings, we assume that both P_1 and P_2 do not contain local sources. Hence, $\mathcal{P}_{seq}$ is constituted by three peers $P_1 = (G_1, \emptyset, \emptyset, \{m_1\})$, $P_2 = (G_2, \emptyset, \emptyset, \{m_2\})$, and $P_3 = (G_3, S_3, L_3, \emptyset)$.

Informally, in the context of sequential composition, we can consider a semantics for P2P systems as modular, if for every query q_1 over P_1, and for every source database D_3 for P_3, the certain answers to q in $\mathcal{P}_{seq}$ with respect to D_3 under the considered semantics can be computed by (i) populating P_2 with the data retrieved by applying the mapping m_2, (ii) using such data to populate P_1 by applying the mapping m_1, and (iii) evaluating q over P_1. Formally, let m_1 be $cq_2 \rightsquigarrow cq_1$, let m_2 be $cq_3 \rightsquigarrow cq'_2$, and consider the peers $\tau(P_1) = (G_1, \{r_1\}, \{m'_1\}, \{m_1\})$ with $m'_1 = \{\mathbf{x} \mid$

$r_1(\mathbf{x})\} \rightsquigarrow cq_1$ and $\tau(P_2) = (G_2, \{r_2\}, \{m_2'\}, \{m_2\})$ with $m_2' = \{\mathbf{x} \mid r_2(\mathbf{x})\} \rightsquigarrow cq_2'$. For a local source database D_3 for P_3, let $\delta(P_2, D_3)$ be the local source database for $\tau(P_2)$ such that $r_2^{\delta(P_2,D_3)} = ans(cq_3, P_3, D_3)$ and let $\delta(P_1, P_2, D_3)$ be the local source database for $\tau(P_1)$ such that $r_1^{\delta(P_1,P_2,D_3)} = ans(cq_2, P_2, \delta(P_2, D_3))$. Now, semantics X is modular if for every query q to P_1 and for every source database D_3 for P_3, we have that $ans_X(q, P_1, \mathcal{P}, \{D_3\}) = ans(q, \tau(P_1), \delta(P_1, P_2, D_3))$.

We show that also in the context of sequential composition, while the epistemic semantics for P2P systems is modular, the FOL semantics is not so.

THEOREM 4.3. *There is a P2P system $\mathcal{P}_{seq} = \{P_1, P_2, P_3\}$ of the form as above, a source database D_3 for P_3, and a query q over P_1 such that $ans_{fol}(q, P_1, \mathcal{P}, \{D_3\}) \neq ans(q, \tau(P_1), \delta(P_1, P_2, D_3))$.*

Proof (sketch). Exploiting a result in [**?**], we exhibit $\mathcal{P}_{seq} = \{P_1, P_2, P_3\}$, D_3, and q such that the claim holds. Let $P_1 = (\{u/2\}, \emptyset, \emptyset, \{m_1\})$, $P_2 = (\{v/2\}, \emptyset, \emptyset, \{m_2\})$, and $P_3 = (\{w/2\}, \{s/2\}, \{\ell_2\}, \emptyset)$, with

$$\begin{array}{rcl} m_1 & = & \{x, y \mid \exists z_1, z_2\,(v(x, z_1) \wedge v(z_1, z_2) \wedge v(z_2, y))\} \rightsquigarrow \\ & & \{x, y \mid u(x, y)\} \\ m_2 & = & \{x, y \mid w(x, y)\} \rightsquigarrow \{x, y \mid \exists z\,(v(x, z) \wedge v(z, y))\} \\ \ell_2 & = & \{x, y \mid s(x, y)\} \rightsquigarrow \{x, y \mid w(x, y)\} \end{array}$$

Consider the source database $D_3 = \{s(a_i, a_{i+1}) \mid 1 \leq i \leq 7\}$ for P_2. It is easy to see that for the query $q = \{x, y \mid \exists z\,(u(x, z) \wedge u(z, y))\}$ we have that $ans_{fol}(q, P_1, \mathcal{P}, \{D_3\}) = \{(a_1, a_7)\}$, while $ans(q, \tau(P_1), \delta(P_1, P_2, D_3)) = \emptyset$. □

For the epistemic semantics, from the results in the next section, we get the following theorem.

THEOREM 4.4. *Let $\mathcal{P}_{seq}$ and D_3 be as above. Then, for every query q over P_1 we have that $ans_{\mathbf{k}}(q, P_1, \mathcal{P}, \{D_3\}) = ans(q, \tau(P_1), \delta(P_1, P_2, D_3))$.*

A problem related to the one considered here for sequential P2P systems is the one of *mapping composition*, as defined in [**?**]. In that paper, the authors study a system in which peer schemas are empty, and P2P mappings are as here (i.e., GLAV mappings between CQs), but interpreted according to the FOL semantics. The authors show that in this setting the composition of two (sets of) P2P mappings is quite involved, and, if it exists [**?**], it is generally formed by an infinite number of P2P mappings between the first and the last peer.

Under the epistemic semantics it is actually always possible to compose mappings, and the resulting mapping is still of (almost) the same form as the original ones.

Let us define composition of two sequential P2P mappings as follows.

DEFINITION 4.5. *Let $\mathcal{P}_{seq}$ be constituted by the three peers $P_1 = (G_1, \emptyset, \emptyset, \{m_1\})$, $P_2 = (G_2, \emptyset, \emptyset, \{m_2\})$, and $P_3 = (G_3, S_3, L_3, \emptyset)$, such that G_1 and G_2 are simple relational alphabets (empty peer schemas), and m_1 and m_2 are the following mapping assertions:*

$$\begin{array}{rcl} m_1 & = & cq_2 \rightsquigarrow cq_1 \\ m_2 & = & cq_4 \rightsquigarrow cq_3 \end{array}$$

where cq_1 is a query over P_1, cq_2 and cq_3 are queries over P_2, and cq_4 is a query over P_3. Furthermore, let $r_2 AuxAlph(P_2)$ be the auxiliary predicate symbol related in $\tau(P_2)$ to the mapping assertion m_2, i.e., such that $Q(r_2) = cq_3$ in $\tau(P_2)$. The composition *of the mapping m_1 and m_2, denoted by m_{12}, is the following mapping assertion between P_1 and P_3:*

$$m_{12} = cq \rightsquigarrow cq_1$$

where:

- *$cq = cq_2'[cq_3/r_2]$, i.e., cq is the query obtained from the query cq_2' by replacing each occurrence of r_2 with cq_3 (with the proper substitution of the distinguished variables);*
- *cq_2' is the* perfect reformulation *of the query cq_2 in $\tau(P_2)$. (The notion of perfect reformulation is given in Definition **??**). Notably, cq_2' is a union of conjunctive queries (UCQ), since it can be shown that the perfect reformulation of a conjunctive query in this setting is always a finite UCQ. Moreover, the only relation symbol that occurs in cq_2' is r_2;*

Roughly speaking, what the above definition says is that the composition of m_1 and m_2 is computed by first reformulating the tail query of m_1 (i.e., the one over the intermediate peer P_2) according to P_2, and then using the mapping m_2 to transform such a query into a query over P_3. Notably, the data retrieved by the mapping are interpreted exactly as before on the global schema of the first peer, since the head of the composed mapping is identical to the head of the first mapping.

The mapping composition above defined satisfies the notion of correctness given in [**?**], i.e., the original system and the one obtained by dropping peer P_2 and using the mapping obtained by composition are equivalent with respect to query answering.

THEOREM 4.6. *Let $\mathcal{P}_{seq}$ be constituted by the three peers P_1, P_2, P_3 as in Definition **??** and let $\mathcal{P}'_{seq}$ be constituted by $P_1' = (G_1, S_1, L_1, \{m_{12}\})$ and P_3, where m_{12} is the composition of m_1 and m_2. Let D_1 be a local source database for P_1 and P_1' and let D_3 be a local source database for P_3, and let $\mathcal{D} = D_1 \cup D_3$. Then, for every query q over G_1 we have that $ans_{\mathbf{k}}(q, P_1, \mathcal{P}_{seq}, \mathcal{D}) = ans_{\mathbf{k}}(q, P_1', \mathcal{P}'_{seq}, \mathcal{D})$.*

Proof (sketch). The proof is a consequence of Theorem **??** and of the fact that the set of answers returned by the execution of P_1.user-query-handler(q) over $\mathcal{D}$ is the same as the set of answers returned by the execution of P_1'.user-query-handler(q) over $\mathcal{D}$. □

4.3 Simple cycle between two peers

We consider a P2P system $\mathcal{P}_{cyc}$ with the structure depicted in Figure **??**(c). The presence of a cycle between two peers suffices to make query answering undecidable under the FOL semantics.

THEOREM 4.7. *There is a P2P system $\mathcal{P}_{cyc} = \{P_1, P_2\}$ of the form as above, a source database $\mathcal{D}$ for $\mathcal{P}_{cyc}$, such that computing the certain answers to queries over the single peers P_1 and P_2 is decidable, while computing the certain answers to queries in $\mathcal{P}_{cyc}$ based on $\mathcal{D}$ under the FOL semantics is undecidable.*

Proof (sketch). The theorem follows from undecidability of query answering under inclusion and functional dependencies [**?**, **?**, **?**].

Consider a relational schema $\mathcal{R}$ with inclusion and functional dependencies. We construct the peer $P_1 = (G_1, S_1, L_1, M_1)$ as follows: G_1 contains the relations of $\mathcal{R}$, plus two additional relations *inc* and *fun*, both containing one attribute $r.A$ for each attribute A in a relation r of $\mathcal{R}$. G_1 contains all inclusion assertion of $\mathcal{R}$, plus one inclusion assertion $r[\mathbf{A}, \mathbf{B}] \subseteq \mathit{inc}[r.\mathbf{A}, r.\mathbf{B}]$ and one functional dependency $\mathit{fun} : r.\mathbf{A} \rightarrow r.\mathbf{B}$, for each functional dependency $r : \mathbf{A} \rightarrow \mathbf{B}$ in $\mathcal{R}$ (we have denoted by $r.\mathbf{A}$ the tuple of attributes corresponding to $\mathbf{A}$). S_1 contains a source relation s_r for each relation r in $\mathcal{R}$, and L_1 maps such relations to the corresponding relations in G_1. M_1 contains a single P2P mapping assertion $\{\mathbf{x} \mid \mathit{inc}(\mathbf{x})\} \rightsquigarrow \{\mathbf{x} \mid \mathit{rem}(\mathbf{x})\}$.

Then we construct the peer $P_2 = (G_2, \emptyset, \emptyset, M_2)$, where G_2 contains only the relation *rem* (of the same arity as *inc* and *fun*), and M_2 contains a single P2P mapping assertion $\{\mathbf{x} \mid \mathit{rem}(\mathbf{x})\} \rightsquigarrow \{\mathbf{x} \mid \mathit{fun}(\mathbf{x})\}$.

Notice that query answering in P_1 is decidable, since all functional dependencies are on the relation *fun*, which is not related through inclusion dependencies to the other relations in G_1, and the implication problems and query answering problems for inclusion and functional dependencies separately are decidable [**?**, **?**]. Also, P_2 is trivially decidable. On the other hand, under the FOL semantics, the P2P mappings propagate the functional dependencies on *fun* to *inc*, and hence in turn to the relations in G_1. Therefore, the whole set of dependencies in $\mathcal{R}$ are reflected in G_1, thus making query answering in the P2P system as a whole undecidable. □

Notice that, since P_1 and P_2 are in general designed independently of each other, even if care is taken to retain decidability of query answering for each of them separately, when interconnected in a P2P system, under the FOL semantics there is no way to ensure decidability of query answering in the whole system, since no single actor has the control on all the P2P mappings. This is a further indication of the lack of modularity in systems based on the FOL semantics. Observe also that the only way to retain decidability would be to trade it with generality, by restricting the topology of the P2P mappings [**?**, **?**, **?**]. In practice this may even be unfeasible, again since no actor is in control of all P2P mappings.

Under the epistemic semantics we can retain both generality and decidability for P2P systems with arbitrary structure, as shown in the next section.

4.4 Data integration

Consider a P2P system constituted by a single node, with no P2P mapping. In other words, the only mappings in the system are the local mappings between the peer schema and the sources. The question is to compare the usual FOL interpretation of the local mapping assertions (as specified in Section 3.2), with the epistemic interpretation of such mapping assertions. Interestingly, it is possible to show that, in this case, the two semantics coincide.

Another way to compare P2P systems and traditional data integration systems is to consider a P2P system $\mathcal{P}_{di}$ with the structure depicted in Figure **??**(d), and to assume that P_1 has no local sources, and that each of the peers P_2 and P_3 consists of a single data source, i.e., G consists of a single relation and L maps such a relation to the source. This case corresponds to the traditional data integration setting in the following sense: P_1 acts as the global schema, P_2 and P_3 as sources, and the P2P mappings of P_1 as GLAV mappings between the global schema and the sources. Again, it is possible to show that in this case the two semantics coincide.

The above observations indicate that the data integration setting does not contain sufficient structure to get into the subtleties that arise in P2P systems. And this justifies why, in data integration, it has not been necessary to go beyond FOL, for example introducing semantics based on epistemic notions.

5. QUERY ANSWERING IN P2P SYSTEMS

In this section we address query answering in P2P systems. We first establish the equivalence between a system $\mathcal{P}$ and its transformation $\tau(\mathcal{P})$, then we present a distributed algorithm that is able to compute the certain answers to a query under the epistemic semantics, and finally we address termination, correctness, and complexity of the algorithm.

5.1 Equivalence between $\mathcal{P}$ and $\tau(\mathcal{P})$

We start by characterizing the knowledge of a P2P system in terms of a first-order theory, called the first-order extension of the system. Given a P2P system $\mathcal{P}$ and a source database $\mathcal{D}$ for $\mathcal{P}$, the *first-order extension* of $\mathcal{P}$ and $\mathcal{D}$, denoted as $\mathit{FOE}(\mathcal{P}, \mathcal{D})$, is the minimal first-order theory that contains a ground fact for each tuple in $\mathcal{D}$, the peer theory T_P for each peer in $\mathcal{P}$, and such that, for each peer $P = (G, S, L, M)$ in $\mathcal{P}$, for each P2P mapping assertion $q_1 \rightsquigarrow q_2 \in M$, and for each tuple $\mathbf{t}$ of elements from Γ, if $\mathit{FOE}(\mathcal{P}, \mathcal{D}) \models \exists \mathbf{y}\, \mathit{body}_{q_1}(\mathbf{t}, \mathbf{y})$ then the sentence $\exists \mathbf{z}\, \mathit{body}_{q_2}(\mathbf{t}, \mathbf{z})$ belongs to $\mathit{FOE}(\mathcal{P}, \mathcal{D})$.

The next theorem is an immediate consequence of the above definition and of the definition of epistemic model of a P2P system provided in Section 3. In the following, we call an epistemic model $(\mathcal{I}, \mathcal{W})$ for $\mathcal{P}$ based on $\mathcal{D}$ *maximal* if there

exists no epistemic model $(\mathcal{J}, \mathcal{W}')$ for $\mathcal{P}$ based on $\mathcal{D}$ such that $\mathcal{W} \subset \mathcal{W}'$. Moreover, it is immediate to verify that in a maximal epistemic model $(\mathcal{I}, \mathcal{W})$ for $\mathcal{P}$ based on $\mathcal{D}$, we have that $\mathcal{I} \in \mathcal{W}$, and for each $\mathcal{J} \in \mathcal{W}$, $(\mathcal{J}, \mathcal{W})$ is an epistemic model for $\mathcal{P}$ based on $\mathcal{D}$. In other words, a maximal epistemic model is completely characterized by the set of interpretations $\mathcal{W}$. Therefore, without loss of generality, from now on we will assume that a maximal epistemic model is a set of interpretations.

THEOREM 5.1. *Let $\mathcal{P}$ be a P2P system and let $\mathcal{D}$ be a source database for $\mathcal{P}$. The set of interpretations $\{\mathcal{I} \mid \mathcal{I} \models FOE(\mathcal{P}, \mathcal{D})\}$ is the unique maximal epistemic model $\mathcal{W}$ for $\mathcal{P}$ based on $\mathcal{D}$.*

PROOF. Suppose there exists $\mathcal{W}'$ such that $\mathcal{W} \subset \mathcal{W}'$ and $\mathcal{W}'$ is an epistemic model for $\mathcal{P}$ based on $\mathcal{D}$. Let $\mathcal{I} \in \mathcal{W}' - \mathcal{W}$. Then, $\mathcal{I} \not\models FOE(\mathcal{P}, \mathcal{D})$. Since $\mathcal{W}$ is an epistemic model for $\mathcal{P}$ based on $\mathcal{D}$, and since $\mathcal{I}$ satisfies the peer theory T_P of all peers $P \in \mathcal{P}$, it follows that there exists a P2P mapping assertion $q_1 \rightsquigarrow q_2 \in M$ for some peer $P = (G, S, L, M)$ in $\mathcal{P}$, and a tuple $\mathbf{t}$ of elements from Γ, such that $\mathcal{I} \models \exists \mathbf{y}\, body_{q_1}(\mathbf{t}, \mathbf{y})$ and $\mathcal{I} \not\models \exists \mathbf{z}\, body_{q_2}(\mathbf{t}, \mathbf{z})$. But this contradicts the fact that, by definition of epistemic model, the sentence $\forall \mathbf{x}\, ((\mathbf{K}\, \exists \mathbf{y}\, body_{q_1}(\mathbf{x}, \mathbf{y})) \supset \exists \mathbf{z}\, body_{q_2}(\mathbf{x}, \mathbf{z}))$ must hold in $\mathcal{W}'$. Consequently, such a $\mathcal{W}'$ does not exists. □

As a consequence, we have that the certain answers to a query q are the answers over the maximal epistemic model $\mathcal{W}$ of $\mathcal{P}$ based on $\mathcal{D}$, i.e., $ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D}) = \{\mathbf{t} \mid \mathbf{t} \in q^{\mathcal{I}}$ for each $\mathcal{I} \in \mathcal{W}\}$.

We show that each P2P system $\mathcal{P}$ is equivalent to the system $\tau(\mathcal{P})$ obtained by adding an additional source for each P2P mapping assertion, as specified in Section **??**. More precisely, the following theorem holds.

THEOREM 5.2. *Let $\mathcal{P}$ be a P2P system, let $\mathcal{D}$ be a source database for $\mathcal{P}$, and let $\mathcal{W}$ be the maximal epistemic model of $\tau(\mathcal{P})$ based on $\mathcal{D}$. Then, the maximal epistemic model of $\mathcal{P}$ based on $\mathcal{D}$ is the set of interpretations*

$$\{\mathcal{I}' \mid \mathcal{I} \in \mathcal{W}\}$$

where $\mathcal{I}'$ is the restriction of the interpretation $\mathcal{I}$ to the predicates in $\mathcal{A} - \bigcup_{P \in \mathcal{P}} AuxAlph(P)$, and $\mathcal{A}$ denotes the union of all predicates occurring in the schemas and in the source schemas of all the peers in $\tau(\mathcal{P})$.

PROOF. The proof is immediate from the definition of first-order extension, Theorem **??**, and the definition of $\tau(\mathcal{P})$. □

From the above theorem, it follows that, since the interpretation of each predicate in $\mathcal{A} - \bigcup_{P \in \mathcal{P}} AuxAlph(P)$ is the same, answers to queries in $\mathcal{P}$ based on $\mathcal{D}$ and in $\tau(\mathcal{P})$ based on $\mathcal{D}$ are the same. Notice that such an equivalence holds only when considering the source database $\mathcal{D}$ for $\mathcal{P}$ as a source database for $\tau(\mathcal{P})$, i.e, when assuming that each additional source (that is, the extension of each predicate in $AuxAlph(P)$ in each peer) is empty.

In the following, we assume to deal with a P2P system transformed according to $\tau()$, i.e., in which such additional sources are defined in each peer and the local mapping assertions involving the additional sources are defined.

5.2 Datalog queries and perfect reformulations

We briefly recall the notion of Datalog program, which will be used in the rest of the section.

A *Datalog rule* is an expression of the form $h(\mathbf{x}) \leftarrow conj(\mathbf{x}, \mathbf{y})$, where $h(\mathbf{x})$ is an atom, $conj(\mathbf{x}, \mathbf{y})$ is a set of atoms, $\mathbf{x} = x_1, \ldots, x_n$ and $\mathbf{y} = y_1, \ldots, y_m$, where x_i and y_j are either variables or constants of Γ. We call $h(\mathbf{x})$ the *head* of the rule, and $conj(\mathbf{x}, \mathbf{y})$ the *body*.

A *Datalog program* DP is a pair (DP_I, DP_E) where DP_I, called the IDB (Intensional Database), is a set of Datalog rules, and DP_E, the EDB (Extensional Database), is a set of facts. The predicates appearing in the heads of the rules in DP_I are called the *IDB predicates*, all the other predicates are called *EDB predicates* and are the only predicates that may appear in DP_E. A *Datalog query* q is a pair (r_q, DP_I) such that $DP = (DP_I, \emptyset)$ is a Datalog program with empty EDB and having the predicate r_q among its IDB predicates.

The semantics of a Datalog program DP is given through the well-known notion of least fixpoint model $LFP(DP)$ of the program [**?**]. The evaluation of a Datalog query $q = (r_q, DP_I)$ over a FOL interpretation $\mathcal{I}$ of the EDB predicates of DP_I is the extension of the predicate r_q in the least fixpoint model of the Datalog program $(DP_I, EDB(\mathcal{I}))$, i.e.,

$$q^{\mathcal{I}} = \{r_q(\mathbf{t}) \mid r_q(\mathbf{t}) \in LFP((DP_I, EDB(\mathcal{I})))\},$$

where $EDB(\mathcal{I})$ is the set of facts that hold in the interpretation $\mathcal{I}$, i.e., $EDB(\mathcal{I}) = \{r(\mathbf{t}) \mid \mathbf{t} \in r^{\mathcal{I}}\}$.

We next introduce the notion of perfect reformulation of a query.

DEFINITION 5.3. *Given a query language $\mathcal{L}_U$, a peer $P = (G, S, L, M)$ and a query $q \in \mathcal{L}_U$ over G, a Datalog query q_1 over the alphabet $S \cup AuxAlph(P)$ is a* perfect reformulation *of q in P, if, for each local source database D_1 for $\tau(P)$, we have that $q_1^{D_1} = ans(q, \tau(P), D_1)$.*

In the following, we consider a query language $\mathcal{L}_U$ accepted by all peers, and such that, for each peer $P \in \mathcal{P}$ and for each $q \in \mathcal{L}_U$, there exists a Datalog query q' that is a perfect reformulation of q according to the definition given above (cf. [**?**, **?**, **?**]). We assume that the language $\mathcal{L}_U$ is able to express at least conjunctive queries. Moreover, we assume that each peer P has an internal functionality $\mathsf{computePerfectRef}(q, r, P)$ that, given a query q, expressed in the language $\mathcal{L}_U$, issued over the schema of P, computes in a finite amount of time a Datalog query $q' = (r, DP_I)$ such that q' is a perfect reformulation of q.

5.3 The algorithm

We define a distributed algorithm for answering queries in $\mathcal{L}_U$. More specifically, we define the two main functionalities that each peer must provide in order to answer a user query to any peer in the system. Such functionalities are executed over a given source database $\mathcal{D}$, which represents the state of the local sources of the peers when the query is issued by the user.

Each user query q to the peer P is the input of the *user query handler* of P. This module first initiates a transaction, that is identified in the system by a unique transaction id, then passes the query q to its own *peer query handler*. Such a functionality returns a Datalog program DP that makes use of a special query predicate r_q: the evaluation of such a predicate over the least fixpoint model of the program DP constitutes the answer set of the query q.

The peer query handler computes the Datalog program corresponding to the query q as follows: first, the perfect reformulation module of the peer computes the first part of the IDB component of the Datalog program. Then, for each source predicate of the peer P occurring in such a reformulation, the set of facts that constitute the extension of such a predicate in the source database D of the peer is added to the extensional part of the Datalog program. Moreover, for each predicate r in $AuxAlph(P)$ that occurs in the reformulation, corresponding to a P2P mapping assertion involving P, the query $Q(r)$ corresponding to the tail of the corresponding mapping assertion is asked to the peer $P(r)$ over which such a query is expressed, i.e., the peer query handler of peer $P(r)$ is called with the mapping query $Q(r)$ as input. For each such call, the Datalog program obtained as the result is added to the overall program that constitutes the answer to q.

We remark that, in order to guarantee that a peer query handler never processes the same mapping query twice in the same transaction, suitable checks are implemented through the procedures setTransaction and getTransaction. More precisely, two different states are associated to each predicate symbol r in $AuxAlph(P)$ with respect to the transaction T. If the state of r with respect to transaction T is *notProcessed*, then the mapping query $Q(r)$ still has to be processed in the transaction T, therefore the peer query handler has to compute the answer to such a query. If the state of r with respect to transaction T is *processed*, then the mapping query $Q(r)$ has already been processed in the transaction T, so the peer query handler does not process it again. Of course, when a new transaction is started by the user query handler, all predicates are initially in the *notProcessed* state for such a transaction.

The two algorithms are reported in Figure **??**, where we denote as $Eval(r, DP)$ the extension of the predicate r in the least fixpoint model of the Datalog program DP. Obviously, $Eval(r, DP) = q^{\mathcal{I}}$ where $q = (r, DP_I)$ and $\mathcal{I}$ is the interpretation of the EDB predicates of DP such that $r^{\mathcal{I}} = \{\mathbf{t} \mid r(\mathbf{t}) \in DP_E\}$ for each EDB predicate r of DP. Moreover, we denote as $Extension(r, D)$ the set of facts with predicate r in the source database D of peer P, i.e., $Extension(r, D) = \{r(\mathbf{t}) \mid r(\mathbf{t}) \in D\}$.

Algorithm P.user-query-handler
Input: user query $q \in \mathcal{L}_U$
Output: $ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D})$
begin
  generate a new transaction id T;
  $DP := P$.peer-query-handler(q, r_q, T);
  return $Eval(r_q, DP)$
end

Algorithm P.peer-query-handler
Input: query $q \in \mathcal{L}_U$, query predicate r_q, transaction id T
Output: Datalog program $DP = (DP_I, DP_E)$
begin
  $DP_I :=$ computePerfectRef(q, r_q, P);
  $DP_E := \emptyset$;
  for each predicate $r \in S \cup AuxAlph(P)$ occurring in DP_I **do**
    if getTransaction$(r, T) = notProcessed$
    then
    begin
      setTransaction$(r, T, processed)$;
      if $r \in S$
      then $DP_E := DP_E \cup Extension(r, D)$
      else begin /* $r \in AuxAlph(P)$ */
        $DP' := P(r)$.peer-query-handler$(Q(r), r_{Q(r)}, T)$;
        $DP_I := DP_I \cup DP'_I$;
        $DP_E := DP_E \cup DP'_E$
      end
    end;
  return DP
end

Figure 2: Algorithms user-query-handler and peer-query-handler, executed over a source database $\mathcal{D}$

Observe that the Datalog program DP returned to the user query handler by P.peer-query-handler(q, r_q, T) is such that the set of EDB predicates of DP is a set of source predicates of the peers, while the IDB predicates of DP are the query predicate r_q and a set of predicates from the set $\bigcup_{P \in \mathcal{P}} AuxAlph(P)$, i.e., the union of the auxiliary alphabets of the peers.

5.4 Termination, complexity, soundness and completeness

Termination of the algorithm follows immediately from the fact that, through the use of the transaction states of the procedures getTransaction and setTransaction in P.peer-query-handler, each mapping query associated with a predicate in $AuxAlph(P)$ is processed at most once for each user query. Therefore, the following property holds.

THEOREM 5.4. *Let $\mathcal{P}$ be a P2P system, let $\mathcal{D}$ be a source database for $\mathcal{P}$, and let $q \in \mathcal{L}_U$ be a query over the alphabet of a single peer in $\mathcal{P}$. Then, the execution of P.user-query-handler(q) over $\mathcal{D}$ terminates.*

The next theorem gives us soundness and completeness of the technique presented here, with respect to the epistemic semantics.

THEOREM 5.5. *Let $\mathcal{P}$ be a P2P system, let $q \in \mathcal{L}_U$ be a query of arity n over the alphabet of a single peer P in $\mathcal{P}$, and let $\mathcal{D}$ be a source database for $\mathcal{P}$. Then, for each n-tuple*

$\mathbf{t}$ *of constants in* Γ, $\mathbf{t}$ *is in the set of answers returned by the execution of* P.user-query-handler(q) *over* $\mathcal{D}$ *if and only if* $\mathbf{t} \in ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D})$.

PROOF. The proof of this property is based on the following lemma:

LEMMA 5.6. *Let* DP *be the program returned to the user query handler by the execution of* P.peer-query-handler$(q, r_q, T))$ *over* $\mathcal{D}$. *For each peer* $P = (G, S, L, M)$ *in* $\mathcal{P}$, *for each predicate* $r \in AuxAlph(P)$ *and for each tuple* $\mathbf{t}$ *of elements from* Γ, $r(\mathbf{t}) \in LFP(DP)$ *if and only if* $\exists \mathbf{y}\, body_{Q(r)}(\mathbf{t}, \mathbf{y}) \in FOE(\mathcal{P}, \mathcal{D})$.

PROOF. First of all, observe that, by the definition of first-order extension, $FOE(\mathcal{P}, \mathcal{D}) = \bigcup_{i=0}^{\infty} FOE_i$, where

$$\begin{array}{rcl} FOE_0 & = & \mathcal{D} \cup \bigcup_{P \in \mathcal{P}} T_P \\ FOE_{i+1} & = & FOE_i \cup \{\exists \mathbf{z}\, body_{q_2}(\mathbf{t}, \mathbf{z}) \mid FOE_i \models \exists \mathbf{y}\, body_{q_1}(\mathbf{t}, \mathbf{y}) \\ & & \quad \text{and } q_1 \rightsquigarrow q_2 \in M \text{ for some peer } P\} \end{array}$$

Moreover, from the well-known definition of least fixpoint model (see e.g. [**?**]), $LFP(DP) = \bigcup_{i=0}^{\infty} LFP_i$, where

$$\begin{array}{rcl} LFP_0 & = & \mathcal{D} \\ LFP_{i+1} & = & LFP_i \cup \{r(\mathbf{t}) \mid r(\mathbf{x}) \leftarrow conj(\mathbf{x}, \mathbf{y}) \in DP \\ & & \quad \text{and } \exists \text{ tuple } \mathbf{z} \text{ s.t. } conj(\mathbf{t}, \mathbf{z}) \subseteq LFP_i\} \end{array}$$

Now, we prove the thesis by induction on the construction of $LFP(DP)$ and $FOE(\mathcal{P}, \mathcal{D})$. For the base case, the thesis is immediately verified by the definition of FOE_0 and LFP_0. As for the inductive case, if $FOE_i \models \exists \mathbf{y}\, body_{Q(r)}(\mathbf{t}, \mathbf{y})$ then, since by definition of the algorithm P.peer-query-handler DP contains a Datalog query (r, DP') that is a perfect reformulation of $Q(r)$, it follows that $r(\mathbf{t}) \in LFP_{i+1}$. Conversely, if $r(\mathbf{t}) \in LFP_{i+1}$, since r is defined in DP in terms of a perfect reformulation of $Q(r)$, it follows that $\mathcal{D} \cup \bigcup_{P \in \mathcal{P}} T_P \models \exists \mathbf{y}\, body_{Q(r)}(\mathbf{t}, \mathbf{y})$. □

Finally, given a query q over P, by definition of the algorithm P.peer-query-handler, the predicate r_q is defined in the Datalog program DP used by the algorithm P.user-query-handler in terms of a perfect reformulation over the source predicates of P. Therefore, by Lemma **??**, it follows that $r_q(\mathbf{t}) \in LFP(DP)$ if and only if $\mathbf{t} \in ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D})$. □

Finally, we analyze the complexity of our technique for query processing with respect to the size of data stored in the peers of $\mathcal{P}$, i.e., the size of the source database $\mathcal{D}$ for $\mathcal{P}$ (data complexity). To this aim, we point out that in each peer the algorithm computePerfectRef is data independent, therefore data complexity of query answering does not depend on the time needed to compute the perfect reformulations through computePerfectRef(q, r_q, P).

The next theorem gives us a polynomial time bound in data complexity.

THEOREM 5.7. *Let* $\mathcal{P}$ *be a P2P system,* $\mathcal{D}$ *be a source database for* $\mathcal{P}$, $q \in \mathcal{L}_U$ *a query of arity* n *over the alphabet of a single peer in* $\mathcal{P}$, *and* $\mathbf{t}$ *a tuple of arity* n *of constants in* Γ. *The problem of establishing whether* $\mathbf{t} \in ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D})$ *is PTIME-complete in data complexity.*

Proof (sketch). Membership in PTIME follows from Theorem **??** and from the fact that checking whether $\mathbf{t} \in Eval(r_q, DP)$, where DP is the Datalog program returned by the execution of P.peer-query-handler$(q, r_q, T))$ over $\mathcal{D}$, is polynomial in data complexity [**?**]. For the hardness part, it is easy to verify that, for each Datalog program DP, there exists a P2P system $\mathcal{P}$, a source database $\mathcal{D}$ and a query q' over a peer $P \in \mathcal{P}$ such that DP is the program returned by the execution of P.peer-query-handler$(q', r_{q'}, T')$ over $\mathcal{D}$. □

Summarizing, the above theorems state that, given a query language $\mathcal{L}_U$, under the hypothesis that the perfect reformulation problem of queries from $\mathcal{L}_U$ over a single peer is decidable, the computation of the certain answers under the epistemic semantics is decidable and can be done in polynomial time in data complexity.

6. CONCLUSIONS

In this paper we have proposed a new semantics for P2P systems, and have argued that it is more suitable than the commonly adopted semantics based on FOL. We have also presented a sound, complete, and terminating procedure that, under general conditions, can generate a Datalog program that returns the certain answers to a query posed to the P2P system. Notice that our query answering technique can immediately be applied in all contexts in which each peer is able to compute the perfect reformulation of queries: for instance, when the peer schema is simply a relational alphabet with no integrity constraints [**?**, **?**]; when the peer schema is a relational schema with integrity constraints such as keys and foreign keys [**?**, **?**]; or a schema written in a variant of the entity-relationship model [**?**, **?**].

7. 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.

8. REFERENCES

[1] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in P2P systems. *IEEE Internet Computing*, 2002.

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

[3] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of the 10th Ital. Conf. on Database Systems (SEBD 2002)*, pages 161–168, 2002.

[4] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. On the expressive power of data integration systems. In *Proc. of the 21th Int. Conf. on Conceptual Modeling (ER 2002)*, 2002.

[5] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. 29:147–163, 2004.

[6] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 260–271, 2003.

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

[8] M. A. Casanova, R. Fagin, and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *J. of Computer and System Sciences*, 28(1):29–59, 1984.

[9] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[10] A. K. Chandra and M. Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. on Computing*, 14(3):671–677, 1985.

[11] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. of the 9th Int. Conf. on Database Theory (ICDT 2003)*, pages 207–224, 2003.

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

[13] M. Fitting. Basic modal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 365–448. Oxford Science Publications, 1993.

[14] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press, 1999.

[15] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *Proc. of the 4th Int. Workshop on the Web and Databases (WebDB 2001)*, 2001.

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

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

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

[19] C. Koch. Query rewriting with symmetric constraints. In *Proc. of the 2nd Int. Symp. on Foundations of Information and Knowledge Systems (FoIKS 2002)*, volume 2284 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2002.

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

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

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

[23] J. C. Mitchell. The implication problem for functional and inclusion dependencies. *Information and Control*, 56:154–173, 1983.

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

[25] J. D. Ullman. *Principles of Database and Knowledge Base Systems*, volume 1. Computer Science Press, Potomac, Maryland, 1988.