Inconsistency Tolerance in P2P Data Integration: An Epistemic Logic Approach

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

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

[2] Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
Via Salaria 113, 00198 Roma, Italy
{degiacomo, lembo, lenzerini, rosati}@dis.uniroma1.it

Abstract. We study peer-to-peer data integration, where each peer models an autonomous system that exports data in terms of its own schema, and data interoperation is achieved by means of mappings among the peer schemas, rather than through a global schema. We propose a multi-modal epistemic semantics based on the idea that each peer is conceived as a rational agent that exchanges knowledge/belief with other peers, thus nicely modeling the modular structure of the system. We then address the issue of dealing with possible inconsistencies, and distinguish between two types of inconsistencies, called local and P2P, respectively. We define a nonmonotonic extension of our logic that is able to reason on the beliefs of peers under inconsistency tolerance. Tolerance to local inconsistency essentially means that the presence of inconsistency within one peer does not affect the consistency of the whole system. Tolerance to P2P inconsistency means being able to resolve inconsistencies arising from the interaction between peers. We study query answering and its data complexity in this setting, and we present an algorithm that is sound and complete with respect to the proposed semantics, and optimal with respect to worst-case complexity.

1 Introduction

In this paper we study data integration in a peer-to-peer (P2P) architecture. In a P2P data integration system (P2PDIS), each peer is an autonomous information system providing part of the overall information available from a distributed environment, and acts both as a client and as a server. Information integration in these systems does not rely on a single global view (as in traditional data integration [22]): instead, it is achieved by establishing mappings between peers, and by exploiting such mappings to collect and merge data from the various peers when answering user queries.

P2P data integration has been the subject of several investigations in the last years. Recent papers focused on providing techniques for evolving from basic P2P networks supporting only file exchanges to more complex systems like schema-based P2P networks, capable of supporting the exchange of structured contents. From papers like [19,4,18,10,16,27] the idea of peer data management emerges: every peer is characterized by a schema that represents the domain of interest from the peer perspective, and is equipped with mappings to other peers [25], each mapping providing a semantic relationship between pairs of peers. Data integration in such systems is typically virtual: data stored in one peer is not replicated in other peers, and when a query is posed to a peer, query processing is done by both looking at local data, and collecting relevant data from other peers according to the mappings. Cycles in the mappings pose challenging problems, and various proposals have been put forward to deal with them. For example, in [10], an epistemic semantics is proposed that weakens the usual first-order semantics of mappings, and allows for both a better modeling of the modular structure of the system, and decidable (even polynomially tractable w.r.t. data complexity) query answering. Some papers look at peer data management under the perspective of exchanging data between peers. Peers are again interconnected by means of mappings, but in this case, the focus is on materializing the data flowing from one peer to another [14,2].

In this paper we are interested in virtual P2P data integration, and thus we do not deal with the issue of materializing exchanged data. In particular, we aim at addressing an important problem that is still unexplored in P2P data integration, namely inconsistency tolerance, i.e., how to deal with inconsistencies in the data stored by the peers.

The problem of dealing with inconsistency has been addressed in several research projects both in the context of a single database, and in the context of traditional data integration. This problem is closely related to the studies in *belief revision and update* [1,15], which deal with the problem of integrating new information with previous knowledge. In the context of databases, the underlying theory takes the form of a database schema, and the revision process focuses on data. Thus, research in this setting often concentrates on specialized algorithmic and complexity results for this case. The general goal is to provide informative answers even when a database that does not satisfy its integrity constraints (see, for example, [3,8]). Most of these papers rely on the notion of repair as introduced in [3]: a repair of a database is a new database that satisfies the constraints in the schema, and minimally differs from the original one.

The above results are not specifically tailored to the case of different consistent sources that are mutually inconsistent, which is the case of interest in data integration. More recently, some papers (see, e.g., [9,6]) have tackled data inconsistency in a data integration setting, where the basic idea is to apply the repairs to data retrieved from the sources, again according to some minimality criteria. To the best of our knowledge, the only paper that deals with inconsistencies in P2P architectures is [5]. That approach is based on the notion of "solution" for a peer P, i.e., an instance for the peer database schema that respects both the mappings and the trust relationships that P has with other peers, and stays as close as possible to the available data in the system. This mechanism characterizes how each peer locally repairs data collected from other peers. On the converse, we provide here a formal semantics to the whole P2PDIS which does not rely on a particular repairing strategy adopted by the peers.

In this paper we follow the approach of [10] and study its extension as follows:

– We want to stress the modularity of P2P architectures, i.e., the fact that each peer is autonomous. To this end, we formalize a P2P data integration system in terms of a multi-modal epistemic logic, namely $K45_n$, where each peer is modeled as

a rational agent that exchanges knowledge/belief with other peers. This is in line with the idea of modeling a distributed information system in terms of multi-agent modal logic [13]. Our formalization nicely models the modular structure of the system, without resorting to any assumptions, such as acyclicity, on its topology.
– We want our semantics to be inconsistency tolerant in two ways. First, we want a P2PDIS to be able to "isolate" peers that are locally inconsistent, i.e., that contain inconsistent data. Second, we aim at a system that is tolerant to P2P inconsistency, i.e., is able to repair inconsistent data coming from different peers. In order to deal with both types of tolerance, we introduce a novel nonmonotonic epistemic logic, called $K45_n^A$, which extends $K45_n$ with nonmonotonic modal operators. Within this logic, we represent a P2PDIS in which each locally inconsistent peer is isolated, and each other peer on the one hand, believes its own data, and, on the other hand, maximizes information coming from other peers, but without falling into inconsistency.
– Finally, we aim at designing a distributed query answering algorithm in the line of the one proposed in [10]. Indeed, we present an algorithm that is sound and complete with respect to our $K45_n^A$-formalization of P2PDISs, thus showing that query answering is decidable. More precisely, under reasonable assumptions on the reasoning capabilities of each peer, our algorithm works in coNP data complexity (i.e., the complexity with respect to the size of the data at the peer sources). We also observe that the problem is coNP-hard already for very simple peer theories, thus showing that our technique is optimal with respect to worst-case complexity.

The paper is organized as follows. In Section 2 we introduce the P2PDIS framework. In Section 3 we model the framework in terms of the multi-modal epistemic logic $K45_n$. In Section 4 we present $K45_n^A$, and use it for handling inconsistency. In Section 5 we provide a sound and complete query answering technique, and establish computational complexity of query answering. In Section 6 we conclude the paper.

2 Framework

In our work, we use the framework for peer-to-peer (P2P) data integration presented in [10], which is briefly described in this section.

We refer to a fixed, infinite, denumerable set Γ of constants. Such constants are shared by all peers, and denote the data items managed by the P2PDIS. 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 *P2P data integration system* $\mathcal{P} = \{P_1, \ldots, P_n\}$ is constituted by a set of n peers. Each peer $P_i \in \mathcal{P}$ (cf. [19]) is defined as a tuple $P_i = (id, G, S, L, M, \mathcal{L})$, where:

– id is a symbol that identifies the peer P_i within $\mathcal{P}$, called the identifier of P_i.
– G is the *schema* of P_i, which is a finite set of formulas of $\mathcal{L}_{A_G}$ (representing local integrity constraints), where A_G is a relational alphabet (disjoint from the other alphabets in $\mathcal{P}$) called the *alphabet* of P_i. We assume that the language $\mathcal{L}_{A_G}$ of peer P_i includes the special sentence $\bot_i$ that is false in every interpretation for

$\mathcal{L}_{A_G}$. Intuitively, the peer schema provides an intensional view of the information managed by the peer.

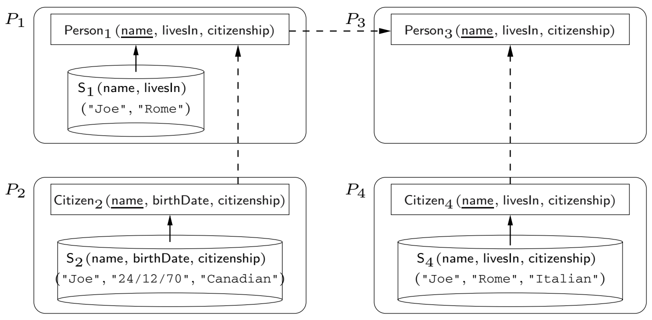
– S is the *(local) source schema* of P_i, which is simply a finite relational alphabet (again disjoint from the other alphabets in $\mathcal{P}$), called the *local alphabet* of P_i. Intuitively, the source schema describes the structure of the data sources of the peer (possibly obtained by wrapping physical sources), i.e., the sources where the real data managed by the peer are stored.
– 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, respectively over the source schema S and over the peer schema G. The local mapping assertions establish the connection between the elements of the source schema and those of the peer schema in P_i. 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. In the terminology used in data integration, the combination of peer schema, source schema, and local mapping assertions constitutes a GLAV *data integration system* [22] managing a set of sound data sources S defined in terms of a (virtual) global schema G.
– M is a set of *P2P mapping assertions*, which specify the semantic relationships that the peer P_i has with the other peers. Each assertion in M is an expression of the form $cq' \rightsquigarrow cq$, where cq, called the *head* of the assertion, is a conjunctive query over the peer (schema of) P_i, while 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 $\mathcal{P}$. A P2P mapping assertion $cq' \rightsquigarrow cq$ from peer P_j to peer P_i expresses the fact that the P_j-concept represented by cq' is mapped to the P_i-concept represented by cq. From an extensional point of view, the assertion specifies that every tuple that can be retrieved from P_j by issuing query cq' satisfies cq in P_i. Observe that no limitation is imposed on the topology of the whole set of P2P mapping assertions in the system $\mathcal{P}$, and hence, as in [10], the set of all P2P mappings may be cyclic.
– $\mathcal{L}$ is a relational query language specifying the class of queries that the peer P_i can process. We assume that $\mathcal{L}$ is any fragment of FOL that accepts at least conjunctive queries and the sentence $\bot_i$. We say that the queries in $\mathcal{L}$ are those *accepted by* P_i. Notice that this implies that, for each P2P mapping assertion $cq' \rightsquigarrow cq$ from another peer P_j to peer P_i in M, we have that cq' is accepted by P_j.

An *extension* for a P2PDIS $\mathcal{P} = \{P_1, \ldots, P_2\}$ is a set $\mathcal{D} = \{D_1, \ldots, D_n\}$, where each D_i is an extension of the predicates in the local source schema of peer P_i.

A P2PDIS, together with an extension, is intended to be queried by external users. A user enquires the whole system by accessing any peer P of $\mathcal{P}$, and by issuing a *query* q to P. The query q is processed by P if and only if q is expressed over the schema of P and is accepted by P.

Example 1. Let us consider the P2PDIS in Figure 1, in which we have 4 peers P_1, P_2, P_3, and P_4 (in the following, we assume that each peer P_i is identified by i).

The global schema of peer P_1 is formed by a relation schema $\mathsf{Person}_1(\underline{\mathsf{name}}, \mathsf{livesIn}, \mathsf{citizenship})$, where name is the key (we underline the key



P_1
Person$_1$ (name, livesIn, citizenship)
S$_1$ (name, livesIn)
("Joe", "Rome")
P_3
Person$_3$ (name, livesIn, citizenship)
P_2
Citizen$_2$ (name, birthDate, citizenship)
S$_2$ (name, birthDate, citizenship)
("Joe", "24/12/70", "Canadian")
P_4
Citizen$_4$ (name, livesIn, citizenship)
S$_4$ (name, livesIn, citizenship)
("Joe", "Rome", "Italian")

Fig. 1. A P2P system

of a relation). P_1 contains a local source $\mathsf{S}_1(\mathsf{name}, \mathsf{livesIn})$, mapped to the global view by the assertion $\{x, y \mid \mathsf{S}_1(x, y)\} \leadsto \{x, y \mid \exists z.\, \mathsf{Person}_1(x, y, z)\}$. Moreover, it has a P2P mapping assertion $\{x, z \mid \exists y.\, \mathsf{Citizen}_2(x, y, z)\} \leadsto \{x, z \mid \exists y.\, \mathsf{Person}_1(x, y, z)\}$ relating information in peer P_2 to those in peer P_1.

P_2 has $\mathsf{Citizen}_2(\underline{\mathsf{name}}, \mathsf{birthDate}, \mathsf{citizenship})$ as global schema, and a local source $\mathsf{S}_2(\mathsf{name}, \mathsf{birthDate}, \mathsf{citizenship})$ mapped to the global schema through the local mapping $\{x, y, z \mid \mathsf{S}_2(x, y, z)\} \leadsto \{x, y, z \mid \mathsf{Citizen}_2(x, y, z)\}$. P_2 has no P2P mappings.

P_3 has $\mathsf{Person}_3(\underline{\mathsf{name}}, \mathsf{livesIn}, \mathsf{citizenship})$ as global schema, contains no local sources, and has a P2P mapping $\{x, y, z \mid \mathsf{Person}_1(x, y, z)\} \leadsto \{x, y, z \mid \mathsf{Person}_3(x, y, z)\}$ with P_1, and a P2P mapping $\{x, y, z \mid \mathsf{Citizen}_4(x, y, z)\} \leadsto \{x, y, z \mid \mathsf{Person}_3(x, y, z)\}$ with P_4.

P_4 has $\mathsf{Citizen}_4(\underline{\mathsf{name}}, \mathsf{livesIn}, \mathsf{citizenship})$ as global schema, and a local source $\mathsf{S}_4(\mathsf{name}, \mathsf{livesIn}, \mathsf{citizenship})$ mapped to the global schema through the local mapping $\{x, y, z \mid \mathsf{S}_4(x, y, z)\} \leadsto \{x, y, z \mid \mathsf{Citizen}_4(x, y, z)\}$. P_4 has no P2P mappings.

Finally, Figure 1 shows also an extension of the P2P data integration system, which includes S_1("Joe", "Rome"), S_2("Joe", "24/12/70", "Canadian"), and S_4("Joe", "Rome", "Italian").

3 Multi-modal Epistemic Formalization

In this section we present a logical formalization of P2PDISs of the kind described above. Although one possible choice for formalizing such systems is classical first order logic, it was argued in [10] that using epistemic logic brings several advantages. However, while [10] resorted to epistemic logic with a single modal operator, here we use a multi-modal epistemic logic, based on the premise that each peer in the system can be seen as a rational agent. Furthermore, we move from the modal logic of knowledge/belief *S5* to the modal logic *K45* [11,23]. More precisely, the formalization we provide in this section, is based on $K45_n$, the multi-modal version of *K45*.

The language $\mathcal{L}(K45_n)$ of $K45_n$ 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 $K45_n$, 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". In fact, in $K45_n$, we do not have that what is known by an agent must hold in the real world: the agent can have inaccurate knowledge of what is true, i.e., believe something to be true although in reality it is false. Often this kind of knowledge is referred to as belief. On the other hand, $K45_n$ 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 $K45_n$ formula ϕ:

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

To define the semantics of $K45_n$, we start from first-order interpretations. In particular, we restrict our attention to first-order interpretations that share a fixed infinite domain Δ. We further assume that for each domain element $d \in \Delta$, we have a unique constant $c_d \in \Gamma$ that denotes exactly d, and, vice versa, that every constant $c_d \in \Gamma$ denotes exactly one domain element $d \in \Delta$[1].

Formulas of $K45_n$ are interpreted over $K45_n$-structures. A *$K45_n$-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:

if $(w_1, w_2) \in R_i$ and $(w_2, w_3) \in R_i$ then $(w_1, w_3) \in R_i$, i.e., R_i is transitive
if $(w_1, w_2) \in R_i$ and $(w_1, w_3) \in R_i$ then $(w_2, w_3) \in R_i$, i.e., R_i is euclidean

A *$K45_n$-interpretation* is a pair (E, w), where $E = (W, \{R_1, \dots R_n\}, V)$ is a $K45_n$-structure, and w is a world in W. 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 $E, w \models \phi$ iff:[2]

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

We say that a sentence ϕ is *satisfiable* if there exists a *$K45_n$-model* for ϕ, i.e., a $K45_n$-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 $\Sigma \models_{K45_n} \phi$, if and only if in every $K45_n$-model E, w of Σ, we have that $E, w \models \phi$.

Notice that, since each accessibility relation of a $K45_n$-structure is transitive and Euclidean, all instances of axiom schemas 4 and 5 are satisfied in every $K45_n$-interpretation, whereas no instance of the axiom schema $(\mathbf{K_i}\phi \supset \phi)$ is so.

[1] In other words, the constants in Γ act as *standard names* [23].

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

Due to the characteristics mentioned above, $K45_n$ is well-suited to formalize P2PDISs of the kind presented in Section 2. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a P2PDIS in which each peer P_i has identifier i. For each peer $P_i = (i, G, S, L, M, \mathcal{L})$ we define the theory $\mathcal{T}_K(P_i)$ in $K45_n$ as the union of the following sentences:

- Global schema G of P_i: for each sentence ϕ in G, 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 alphabets of all the other peers in $\mathcal{P}$.
- Local mapping assertions L between G and the local source schema S: for each mapping assertion $\{\mathbf{x} \mid \exists \mathbf{y}.\, \mathit{body}_{cq_S}(\mathbf{x}, \mathbf{y})\} \rightsquigarrow \{\mathbf{x} \mid \exists \mathbf{z}.\, \mathit{body}_{cq_G}(\mathbf{x}, \mathbf{z})\}$ in L, we have
$$\mathbf{K_i}(\forall \mathbf{x}.\, \exists \mathbf{y}.\, \mathit{body}_{cq_S}(\mathbf{x}, \mathbf{y}) \supset \exists \mathbf{z}.\, \mathit{body}_{cq_G}(\mathbf{x}, \mathbf{z}))$$
- P2P mapping assertions M: for each P2P mapping assertion $\{\mathbf{x} \mid \exists \mathbf{y}.\, \mathit{body}_{cq_j}(\mathbf{x}, \mathbf{y})\} \rightsquigarrow \{\mathbf{x} \mid \exists \mathbf{z}.\, \mathit{body}_{cq_i}(\mathbf{x}, \mathbf{z})\}$ between the peer j and the peer i in M, we have
$$\forall \mathbf{x}.\, \mathbf{K_j}(\exists \mathbf{y}.\, \mathit{body}_{cq_j}(\mathbf{x}, \mathbf{y})) \supset \mathbf{K_i}(\exists \mathbf{z}.\, \mathit{body}_{cq_i}(\mathbf{x}, \mathbf{z})) \qquad (1)$$
In words, this sentence specifies the following rule: for each tuple of values $\mathbf{t}$, if peer j knows the sentence $\exists \mathbf{y}.\, \mathit{body}_{cq_j}(\mathbf{t}, \mathbf{y})$, then peer i knows the sentence $\exists \mathbf{z}.\, \mathit{body}_{cq_i}(\mathbf{t}, \mathbf{z})$ holds.

We denote by $\mathcal{T}_K(\mathcal{P})$ the theory corresponding to the P2PDIS $\mathcal{P}$, i.e., $\mathcal{T}_K(\mathcal{P}) = \bigcup_{i=1,\ldots,n} \mathcal{T}_K(P_i)$.

Example 2. We provide now the formalization of the P2PDIS of Example 1. The theory $\mathcal{T}_K(P_1)$ modeling peer P_1 is the conjunction of:

$$\begin{array}{l}
\mathbf{K_1}(\forall x, y, y', z, z'.\, \mathsf{Person}_1(x, y, z) \wedge \mathsf{Person}_1(x, y', z') \supset y = y' \wedge z = z') \\
\mathbf{K_1}(\forall x, y.\, \mathsf{S}_1(x, y) \supset \exists z.\, \mathsf{Person}_1(x, y, z)) \\
\forall x, z.\, \mathbf{K_2}(\exists y.\, \mathsf{Citizen}_2(x, y, z)) \supset \mathbf{K_1}(\exists y.\, \mathsf{Person}_1(x, y, z))
\end{array}$$

The theory $\mathcal{T}_K(P_2)$ modeling peer P_2 is the conjunction of:

$$\begin{array}{l}
\mathbf{K_2}(\forall x, y, y', z, z'.\, \mathsf{Citizen}_2(x, y, z) \wedge \mathsf{Citizen}_2(x, y', z') \supset y = y' \wedge z = z') \\
\mathbf{K_2}(\forall x, y, z.\, \mathsf{S}_2(x, y, z) \supset \mathsf{Citizen}_2(x, y, z))
\end{array}$$

The theory $\mathcal{T}_K(P_3)$ modeling peer P_3 is the conjunction of:

$$\begin{array}{l}
\mathbf{K_3}(\forall x, y, y', z, z'.\, \mathsf{Person}_3(x, y, z) \wedge \mathsf{Person}_3(x, y', z') \supset y = y' \wedge z = z') \\
\forall x, y.\, \mathbf{K_1}(\exists z.\, \mathsf{Person}_1(x, z, y)) \supset \mathbf{K_3}\exists z.\, \mathsf{Person}_3(x, z, y) \\
\forall x, y, z.\, \mathbf{K_4}(\mathsf{Citizen}_4(x, y, z)) \supset \mathbf{K_3}\mathsf{Person}_3(x, y, z)
\end{array}$$

The theory $\mathcal{T}_K(P_4)$ modeling peer P_4 is the conjunction of:

$$\begin{array}{l}
\mathbf{K_4}(\forall x, y, y', z, z'.\, \mathsf{Citizen}_4(x, y, z) \wedge \mathsf{Citizen}_4(x, y', z') \supset y = y' \wedge z = z') \\
\mathbf{K_4}(\forall x, y, z.\, \mathsf{S}_4(x, y, z) \supset \mathsf{Citizen}_4(x, y, z))
\end{array}$$

The extension $\mathcal{D} = \{D_1, \ldots, D_n\}$ of a P2PDIS $\mathcal{P}$ is modeled as a sentence constituted by the conjunction of all facts corresponding to the tuples stored in the sources, i.e., $DB(\mathcal{D}) = \bigwedge_{i=1}^{n} DB(D_i)$ where $DB(D_i) = \mathbf{K_i}(\bigwedge_{t \in r^{D_i}} r(t))$.

A client of the P2PDIS interacts with one of the peers, say peer P_i, posing a *query* to it. A query q is an open formula $q(\mathbf{x})$ with free variables $\mathbf{x}$ expressed in the language accepted by the peer P_i (we recall that such a language is a subset of first-order logic). The semantics of a query $q \in \mathcal{L}$ posed to a peer $P_i = (i, G, S, L, M, \mathcal{L})$ of $\mathcal{P}$ with respect to an extension $\mathcal{D}$ is defined as the set of tuples $ANS_{K45_n}(q, i, \mathcal{P}, \mathcal{D}) = \{\mathbf{t} \mid \mathcal{T}_K(\mathcal{P}) \cup DB(\mathcal{D}) \models_{K45_n} \mathbf{K_i} q(\mathbf{t})\}$, where $q(\mathbf{t})$ denotes the sentence obtained from the open formula $q(\mathbf{x})$ by replacing all occurrences of the free variables in $\mathbf{x}$ with the corresponding constants in $\mathbf{t}$.

Interestingly, our current formalization extends the one in [10] in two ways. First, we have moved to multi-modal epistemic logic, so as to model each peer as an autonomous agent. Second, we have moved from *S5* to *K45*, hence dropping the assumption that what is believed by an agent is actually true. These modifications set the stage for the treatment of inconsistencies to be presented next.

4 Inconsistency Tolerance

We now modify our basic framework so as to be able to handle inconsistency. In particular, we want the P2PDIS to be inconsistency-tolerant in the following sense:

1. When a peer is *locally inconsistent*, i.e., data at the sources in P_i contradict, via the local mapping, the peer schema, making the whole peer inconsistent, the P2PDIS should be equivalent to the one obtained by eliminating the peer P_i from the system. In other words, an inconsistent peer should be "isolated" from the other peers: in this way, a local inconsistency does not affect the overall consistency (and meaning) of the system. The choice of isolating locally inconsistent peers is motivated by the modularity of P2PDISs pursued by our approach, in which each peer is considered as a black box. Of course, the study of inconsistency might be also interesting in an alternative setting not focused on modularity. However, this is outside the scope of the present paper.
2. In the presence of *P2P inconsistency*, i.e., when in a peer P_i the data coming from another peer P_j (through a P2P mapping) contradict the local data of P_i (or the data coming to P_i from another peer P_k), the peer P_i should not reach an inconsistent state: rather, it should discard a *minimal* amount of the data retrieved from the other peers in order to preserve consistency.

We now formally state the above notions of local inconsistency and P2P inconsistency. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a P2PDIS and $\mathcal{D} = \{D_1, \ldots, D_n\}$ be an extension $\mathcal{D}$ for $\mathcal{P}$. We say that:

– A peer $P_i \in \mathcal{P}$ is *locally inconsistent wrt* D_i if $\mathcal{T}_K^-(P_i) \cup DB(D_i) \models_{K45_n} \mathbf{K_i} \bot_i$, where $\mathcal{T}_K^-(P_i)$ is obtained from $\mathcal{T}_K(P_i)$ by dropping the sentences formalizing the P2P mappings (otherwise we say that P_i is locally consistent wrt D_i).
– A peer $P_i \in \mathcal{P}$ is *P2P inconsistent wrt* $\mathcal{D}$ if P_i is locally consistent wrt D_i and $\mathcal{T}_K(\mathcal{P}) \cup DB(\mathcal{D}) \models_{K45_n} \mathbf{K_i} \bot_i$.

As we said before, we aim at a formalization that makes our system inconsistency-tolerant. It is immediate to see that no monotonic logic, e.g., $K45_n$, is suited for this purpose. Therefore, we now introduce a nonmonotonic variant of our logic, called $K45_n^A$.

The Nonmonotonic Modal Logic $K45_n^A$. The language $\mathcal{L}(K45_n^A)$ is an extension of $\mathcal{L}(K45_n)$, obtained by adding to the first-order modal language a new set of modal operators, $\mathbf{A_1}, \ldots, \mathbf{A_n}$. The semantics of $\mathcal{L}(K45_n^A)$ sentences is *nonmonotonic*, and is formally defined as follows. A $K45_n^A$-structure E is a tuple $(W, \{R_1, \ldots, R_n, R_1^a, \ldots, R_n^a\}, V)$, where W is a set of worlds, each R_i and each R_i^a are transitive and Euclidean binary relations over W, and V is a function mapping worlds to first-order interpretations. Therefore, with respect to $K45_n$-structures, $K45_n^A$-structures have n additional accessibility relations $R_1^a, \ldots, R_n^a$. Such relations account for the additional modal operators $\mathbf{A_1}, \ldots, \mathbf{A_n}$.

The notion of truth of a $K45_n^A$ sentence in a world of a $K45_n^A$-structure is analogous to the notion given in Section 3 for $K45_n$, with the addition of:

$$E, w \models \mathbf{A_i}\phi \quad \text{iff} \quad E, w' \models \phi \text{ for each } w' \text{ such that } (w, w') \in R_i^a$$

So far, the logic $K45_n^A$ does not appear as a significant extension of $K45_n$: indeed, according to the above notion of truth, the new modal operators $\mathbf{A_i}$ are treated just like any $\mathbf{K_i}$ operator in $K45_n$, so there is no apparent reason to distinguish the $\mathbf{A_i}$'s operators from the $\mathbf{K_i}$'s. Actually, the different meaning of the two sets of modal operators in the logic $K45_n^A$, as well as its nonmonotonicity, is due to the following notion of $K45_n^A$-model for a sentence ϕ, which makes use of a preference order over $K45_n^A$-structures.

Let $E = (W, \{R_1, \ldots, R_n, R_1^a, \ldots, R_n^a\}, V)$ and $E' = (W', \{R_1', \ldots, R_n', R_1^a, \ldots, R_n^a\}, V')$ be $K45_n^A$-structures. We say that E' *is preferred to* E if the following conditions hold:

1. $W' \supseteq W$ and $V'(w) = V(w)$ for every $w \in W$,
2. $R_i' \supseteq R_i$, for all $i \in \{1, \ldots, n\}$,
3. there exist $w_1 \in W$, $w_2 \in W'$, $i \in \{1, \ldots, n\}$ such that $(w_1, w_2) \in R_i' - R_i$ and there exists no $w' \in W$ such that $(w_1, w') \in R_i$ and $V(w') = V'(w_2)$.

Intuitively, E' is preferred to E if E' is a structure "larger" than E (conditions 1 and 2) and there exists a world w_1 which is connected in E' (through the relation R_i') to a larger set of possible worlds than in E (condition 3), which means that w_1 in E has "less objective knowledge" than in E' with respect to the modality K_i. For instance, it can be immediately verified that, if E' is preferred to E, then, for each first-order sentence ϕ and for each $w \in W$, if $E', w \models \mathbf{K_i}\phi$ then $E, w \models \mathbf{K_i}\phi$, but not vice-versa.

Let $\phi \in \mathcal{L}(K45_n^A)$, let $E = (W, R_1, \ldots, R_n, R_1^a, \ldots, R_n^a, V)$ be a $K45_n^A$-structure, and let $w \in W$. (E, w) is a *$K45_n^A$-model for ϕ* if the following conditions hold:

1. $E, w \models \phi$;
2. $R_i = R_i^a$ for each $i \in \{1, \ldots, n\}$;
3. there exists no $K45_n^A$-structure $E' = (W', \{R_1', \ldots, R_n', R_1^a, \ldots, R_n^a\}, V')$ such that E' is preferred to E, and $E', w \models \phi$.

The notions of model of a set of sentences and of logical implication are defined in the same way as in the case of $K45_n$.

The above semantics formalizes the idea of selecting $K45_n^A$-structures that satisfy two intuitive principles: (*i*) *knowledge is minimal*, which is realized through the notion of preference between structures; and (*ii*) *assumptions are justified by knowledge*, which is realized by the fact that, for each i, the meaning of the operators $\mathbf{A_i}$ and $\mathbf{K_i}$ is the same, since $R_i = R_i^a$. Such semantic principles of minimal knowledge and justified assumptions are well-known in nonmonotonic reasoning [24,26]. In particular, the logic $K45_n^A$ can be seen as a first-order, multimodal generalization of [24].

Handling Local Inconsistency. To capture tolerance wrt local inconsistency, we need to refine the epistemic formalization of P2P mapping assertions presented in Section 3 as follows: for each P2P mapping assertion of peer i in M, we replace in $\mathcal{T}_K(P_i)$ the sentence (1) with

$$\forall \mathbf{x}.\, \neg \mathbf{A_j} \bot_j \wedge \mathbf{K_j}(\exists \mathbf{y}.\, body_{cq_j}(\mathbf{x},\mathbf{y})) \supset \mathbf{K_i}(\exists \mathbf{z}.\, body_{cq_i}(\mathbf{x},\mathbf{z}))$$

It is easy to see that, for a P2PDIS $\mathcal{P}$ without locally inconsistent peers, the new formalization of $\mathcal{P}$ coincides with the formalization in the logic $K45_n$.

On the other hand, the following theorem shows that, with the above change, the P2PDIS is tolerant to local inconsistency, in the sense that it isolates the peers that are locally inconsistent.

Theorem 1. *Let $\mathcal{P}$ be a P2PDIS, let $\mathcal{D}$ be an extension for $\mathcal{P}$, let $P_i \in \mathcal{P}$ be a peer locally inconsistent wrt D_i, and let $\mathcal{P}' = \mathcal{P} - \{P_i\}$. Then, for each query q posed to a peer $P_j \in \mathcal{P}$ different from P_i, we have that $ANS_{K45_n^A}(q, j, \mathcal{P}, \mathcal{D}) = ANS_{K45_n^A}(q, j, \mathcal{P}', \mathcal{D})$.*

Handling Both Local and P2P Inconsistency. We are now ready to formalize, in $K45_n^A$, P2PDISs that are inconsistency-tolerant wrt both local and P2P mappings. Again, the $K45_n^A$ theory representing the P2PDIS $\mathcal{P}$, denoted by $\mathcal{T}_A(\mathcal{P})$, is similar to the theory $\mathcal{T}_K(\mathcal{P})$ defined in Section 3, but with an important difference on how to formalize P2P mapping assertions. In particular, such a formalization is obtained by replacing each sentence of the form (1) with

$$\forall \mathbf{x}.\, \neg \mathbf{A_j} \bot_j \wedge \mathbf{K_j}(\exists \mathbf{y}.\, body_{cq_j}(\mathbf{x},\mathbf{y})) \wedge \neg \mathbf{A_i}(\neg \exists \mathbf{z}.\, body_{cq_i}(\mathbf{x},\mathbf{z})) \supset \mathbf{K_i}(\exists \mathbf{z}.\, body_{cq_i}(\mathbf{x},\mathbf{z}))$$

Informally, the above sentence specifies the following rule: for each tuple of values $\mathbf{t}$, if peer j is consistent and knows the sentence$\exists \mathbf{y}.\, body_{cq_j}(\mathbf{t},\mathbf{y})$, and the sentence $\exists \mathbf{z}.\, body_{cq_i}(\mathbf{t},\mathbf{z})$ *is consistent with what peer i knows*, then peer i knows the sentence $\exists \mathbf{z}.\, body_{cq_i}(\mathbf{t},\mathbf{z})$. In other words, information flows from peer j to peer i through a P2P mapping assertion only if adding such information to peer i does not give rise to a P2P inconsistency in peer i. More precisely, the meaning of the above sentence in $K45_n^A$ is that exactly a *maximal* amount of information (i.e., a maximal set of tuples) consistent with peer i flows from peer j to peer i through the P2P mapping assertion.

The semantics $ANS_{K45_n^A}(q, i, \mathcal{P}, \mathcal{D})$ of a query q posed to a peer P_i of a P2PDIS $\mathcal{P}$ wrt an extension $\mathcal{D}$ is defined as for $K45_n$, except that now we have to take into account the $K45_n^A$ formalization of the $\mathcal{P}$. The following theorem shows that such a formalization is a "conservative extension" of the one based on $K45_n$, in the sense that, if no peer is locally inconsistent, and the data at the sources do not give rise to P2P inconsistencies, then the semantics of queries is the same in the two logics.

Theorem 2. *Let $\mathcal{P}$ be a P2PDIS and let $\mathcal{D}$ be an extension for $\mathcal{P}$ such that each peer in $\mathcal{P}$ is neither locally inconsistent, nor P2P inconsistent wrt $\mathcal{D}$. Then, for each peer $P_i \in \mathcal{P}$ and for each query q posed to P_i, $ANS_{K45_n^A}(q, i, \mathcal{P}, \mathcal{D}) = ANS_{K45_n}(q, i, \mathcal{P}, \mathcal{D})$.*

Moreover, the following theorem shows that the new formalization enjoys one of the basic properties for being tolerant to P2P inconsistency.

Theorem 3. *Let $\mathcal{P}$ be a P2PDIS and let $\mathcal{D}$ be an extension for $\mathcal{P}$. If $P_i \in \mathcal{P}$ is locally consistent wrt D_i, then $\mathcal{T}_A(\mathcal{P}) \cup DB(\mathcal{D}) \not\models_{K45_n^A} \mathbf{K_i}\bot_i$.*

Finally, we remark that the above semantics implies that: (*i*) when inconsistency arises between local data and non-local data in a peer, i.e., when data coming from the peer sources through the local mapping contradicts the data retrieved by a peer through a P2P mapping, then the peer always prefers the local data. Formally, in this case there is one $K45_n^A$-model for the P2PDIS, which represents the situation in which non-local data is discarded; (*ii*) when inconsistency arises between two different pieces of non-local data, i.e., when a piece of data retrieved by a peer through a P2P mapping contradicts another piece of data retrieved through the P2P mappings, then no preference is made between these two pieces of information, in the sense that in this case there are two $K45_n^A$-models for the P2PDIS, each of which represents the situation in which one of the two pieces of data is discarded.

Example 3. Consider the P2PDIS of Example 1. It is easy to see that P_3 gets from P_1 that $\mathsf{Person}_3(\texttt{"Joe"}, \texttt{"Rome"}, \texttt{"Canadian"})$ and from P_4 that $\mathsf{Person}_3(\texttt{"Joe"}, \texttt{"Rome"}, \texttt{"Italian"})$, but since name is a key for Person_3 this would give rise to an inconsistency. As a result, we have two $K45_n^A$ models, one in which $\mathsf{Person}_3(\texttt{"Joe"}, \texttt{"Rome"}, \texttt{"Canadian"})$ holds, and one in which $\mathsf{Person}_3(\texttt{"Joe"}, \texttt{"Rome"}, \texttt{"Italian"})$ holds, and hence P_3 does not know anymore the citizenship of "Joe". However, P_3 still knows that "Joe" lives in "Rome". In other words, the query $\{x \mid \exists y. \mathsf{Person}_3(\texttt{"Joe"}, x, y)\}$ returns $\{\texttt{"Rome"}\}$, while the query $\{y \mid \exists x. \mathsf{Person}_3(\texttt{"Joe"}, x, y)\}$ returns the empty set.

5 Query Processing

In this section we study query answering in a P2P setting. We present a distributed algorithm for answering queries in a P2P system, we prove its termination, soundness and completeness, and then we use it to provide the complexity characterization of the query answering problem. The algorithm extends the one presented in [7] with the capability in handling inconsistency in accordance to the P2P system formalization in the multimodal logic $K45_n^A$.

The Algorithm. The algorithm is based on two main functionalities, called *user query handler* and *peer query handler*, that are described in Figure 2. Each peer must provide such functionalities in order to answer a user query posed to any peer in the P2P system $\mathcal{P}$. Such functionalities are executed over an extension $\mathcal{D}$ of $\mathcal{P}$.

```
**Algorithm** $P$.user-query-handler, with $P = (id, G, S, L, M, \mathcal{L})$
**Input:** user query $q \in \mathcal{L}$
**Output:** $ANS_{K45_n^A}(q, id, \mathcal{P}, \mathcal{D})$
**begin**
  generate a new transaction id $T$;
  $(DP, r_q) :=$ $P$.peer-query-handler$(q, T)$;
  **return** $Eval(r_q, DP)$;
**end**
```

```
**Algorithm** $P$.peer-query-handler
**Input:** query $q \in \mathcal{L}$, transaction id $T$
**Output:** Datalog$^\neg$ program $DP = (DP_I, DP_E)$, query predicate $r_q$ in $DP_I$
**begin**
  $(r_q, DP_I) :=$ computePerfectRef$(q, \sigma(P))$;
  $(\bot_{id}, DP'_I) :=$ computePerfectRef$(\bot_{id}, \sigma(P))$;
  $DP_I = DP_I \cup DP'_I$;        $DP_E := \emptyset$;
  **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** $DP_E := DP_E \cup Ext(r, \mathcal{D})$;
      **else if** isConsistent$(\pi(r))$
      **then begin**
        $(DP', r') := \pi(r)$.peer-query-handler$(Q(r), T)$;
        $\rho := r(\mathbf{x}) \leftarrow r'(\mathbf{x}), not\ \bot_{id}$;
        $DP_I := DP_I \cup \rho \cup DP'_I$;
        $DP_E := DP_E \cup DP'_E$;
      **end**
    **end**
  **return** $(DP, r_q)$;
**end**
```

Fig. 2. Algorithms user-query-handler and peer-query-handler, executed over an extension $\mathcal{D}$

Each user query q to the peer P is the input of the user query handler of P. Such a module computes the set $ANS_{K45_n^A}(q, \mathcal{P}, \mathcal{D})$ by evaluating a suitable Datalog$^\neg$ program, i.e., a Datalog program enriched with unstratified negation, which is returned by the peer query handler of P. Roughly speaking, the module peer query handler reformulates the query q in terms of a Datalog$^\neg$ program over the data sources of P, and combines rules and facts thus obtained with the programs provided by consistent peers connected to P that the module queries by calling their own peer query handler. A suitable rule (using negation in its body) is also added to the program to make data coming from other peers contribute to answer computation only if they do not generate inconsistency within P (i.e., they do not contradict local data of the peer P or data coming from another peer). Obviously, each queried peer can in turn propagate the computation by invoking the peer query handlers of its neighborhoods. The association of the identifier of a transaction (started by the user query handler) to each peer query handler call ensures termination of the process (even in the presence of cycles among peers).

In the algorithms of Figure 2, DP denotes a Datalog$^\neg$ program constituted by a set of rules DP_I, and a set of facts DP_E. The pair (r_q, DP_I), denotes a Datalog$^\neg$ query, whereas $Eval(r_q, DP)$ indicates the evaluation of the predicate r_q over the stable models of the program DP [28]. Also, starting from $P = (id, G, S, L, M, \mathcal{L})$ we define a simplified peer $\sigma(P) = (id, G, S \cup AuxAlph(P), L \cup L_{AuxAlph(P)}, \emptyset, \mathcal{L})$, where we drop the P2P mapping assertions, and "simulate" their effects by adding the new source symbols $AuxAlph(P)$ (one for each assertion) and the new local mappings $L_{AuxAlph(P)}$ involving them. In particular for a mapping $cq' \rightsquigarrow cq$ from peer P' to P, we introduce a new source relation r with a local mapping $\{\boldsymbol{x} \mid r(\boldsymbol{x}\} \rightsquigarrow cq$, and use the notation $Q(r)$ to denote cq' and $\pi(r)$ to denote P' (see also [10] for further details).

The peer query handler makes use of a function computePerfectRef which, taken as input a query q and $\sigma(P)$, returns the *perfect reformulation of* q *in* $\sigma(P)$, that is a query q_r such that, for each extension D of the source predicates $S \cup AuxAlph(P)$, $q_r^D = \{\mathbf{t} \mid \mathcal{T}(\sigma(P)) \cup \mathcal{T}_D \models q(\mathbf{t})\}$, where $\mathcal{T}(\sigma(P))$ is the first-order theory obtained from $\mathcal{T}_K(\sigma(P))$ by dropping the modal operator in front of the assertions constructed from G and $L \cup L_{AuxAlph(P)}$ (see Section 3) and $\mathcal{T}_D$ is used denote the set of facts corresponding to D. In the terminology used in data integration, computePerfectRef computes the query that returns the certain answers to the query q posed to the single peer $\sigma(P)$ wrt a source database D [22].

We assume that each peer P is able to compute the perfect reformulation in $\sigma(P)$ of any query q accepted by P. We also assume that such reformulation can be expressed in Datalog$^\neg$, and call *reformulation capable* each peer that satisfies the above assumptions. Notable cases in which the above assumption holds can be found in the extensive literature on data integration and data exchange (see, e.g., [20,12]).

The use of the functions getTransaction and setTransaction guarantees that a peer query handler never processes the same mapping query twice in the same transaction, whereas isConsistent$(\pi(r))$ is used to check if the peer $\pi(r)$ is locally consistent. This function is implemented by asking the query $\bot_j$ to $\pi(r)$, where j is the identifier of $\pi(r)$. Furthermore, $Ext(r, \mathcal{D})$ denotes the set $\{r(\mathbf{t}) \mid \mathbf{t} \in r^{\mathcal{D}}\}$, i.e., the extension of r in $\mathcal{D}$. Finally, the rule $\rho := r(\mathbf{x}) \leftarrow r'(\mathbf{x}), not\ \bot_{id}$ specifies that data coming from the peer $\pi(r)$ contribute to the answer to the query q only if they do not generate any inconsistency in the peer P. Such a mechanism does the job since we include in the program DP_I the rules that define the predicate $\bot_{id}$ by means of the function call computePerfectRef$(\bot_{id}, \sigma(P))$.

Termination and Correctness. 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. Moreover, the algorithm is sound and complete with respect to the $K45_n^A$ formalization of the P2P system.

Theorem 4. *Let $\mathcal{P}$ be a P2P system, $P = (id, G, S, L, M, \mathcal{L})$ a peer in $\mathcal{P}$, $q \in \mathcal{L}$ a query of arity n over P, and $\mathcal{D}$ an extension for $\mathcal{P}$. Then, the execution of P.user-query-handler(q) over $\mathcal{D}$ terminates, and a n-tuple $\mathbf{t}$ of constants in Γ is in the set of returned answers if and only if $\mathbf{t} \in ANS_{K45_n^A}(q, id, \mathcal{P}, \mathcal{D})$.*

Complexity. Finally, we characterize the computational complexity of the problem of query answering in our P2P data integration setting, with respect to the size of data stored in the peers of $\mathcal{P}$, i.e., the size of the extension $\mathcal{D}$ for $\mathcal{P}$ (*data complexity*). Notice that computing perfect reformulations through the algorithm computePerfectRef does not depend on the data at the sources, therefore it does not affect data complexity.

Theorem 5. *Let $\mathcal{P}$ be a P2P system where each peer is reformulation capable. Let $P = (id, G, S, L, M, \mathcal{L})$ be a peer in $\mathcal{P}$, $\mathcal{D}$ an extension for $\mathcal{P}$, $q \in \mathcal{L}$ a query of arity n over P, and $\mathbf{t}$ a n-tuple of constants in Γ. The problem of establishing whether $\mathbf{t} \in ANS_{K45_n^A}(q, id, \mathcal{P}, \mathcal{D})$ is in coNP in the size of $\mathcal{D}$ (i.e., in data complexity). Moreover, it is coNP-hard in data complexity even in a setting where only key constraints are allowed in peer schemas.*

Proof (sketch). Membership in coNP follows from Theorem 4 and from the fact that checking whether $\mathbf{t} \in Eval(r_q, DP)$, where DP is a Datalog$^\neg$ program, is coNP-complete in data complexity [17]. The hardness part can be proved by a reduction of the three-colorability problem to our problem in the setting where only key constraints are allowed in peer schemas. The proof follows the line used for establishing coNP-hardness of query answering in the setting of a single inconsistent database in [8]. □

According to the above theorem, query answering in a P2PDIS under the $K45_n^A$ semantics is decidable and our algorithm turns out to be optimal with respect to worst-case data complexity. Notice that assuming that each peer is reformulation capable strips off cases in which query answering is undecidable and guarantees its membership in coNP. Obviously, with respect to [10], a computational complexity blow up in query answering arises, which is the price that we have to pay to deal with inconsistent data.

6 Conclusions

In this paper we have proposed a multi-modal nonmonotonic formalization for P2PDISs which allowed us to properly model the modularity of a P2P system, localize local inconsistency, and handle peers that may provide mutually inconsistent data. We have also provided an algorithm for query processing in our setting that is sound and complete with respect to the multi-modal semantics of the system, and have characterized the computational complexity of the query answering problem. The results reported here can be extended in several directions. First, we can remove the assumption that all peers share a common alphabet of constants by making use of mapping tables [21]. Also, we believe that preferences between peers can be smoothly integrated in our framework, in the line of [5]. We aim also at extending the framework to the case in which each peer in the system has its own strategy for resolving data inconsistency.

Acknowledgements. This research has been partially supported by the projects INFOMIX (IST-2001-33570), and TONES (IST-007603) funded by the EU, and by the project HYPER, funded by IBM through a Shared University Research Award Grant. The authors wish to thank Luciano Serafini for precious comments about the paper.

References

1. C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic*, 50:510–530, 1985.
2. M. Arenas, P. Barcelo, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *Proc. of PODS 2004*, pages 229–240, 2004.
3. M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS'99*, pages 68–79, 1999.
4. 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 WebDB 2002*, 2002.
5. L. E. Bertossi and L. Bravo. Query answering in peer-to-peer data exchange systems. In *In Proc. of the EDBT Workshop on Peer-to-Peer Computing and Databases (P2P&DB 2004)*, pages 476–485, 2004.
6. L. Bravo and L. Bertossi. Logic programming for consistently querying data integration systems. In *Proc. of IJCAI 2003*, pages 10–15, 2003.
7. A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Information Systems*, 29:147–163, 2004.
8. A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS 2003*, pages 260–271, 2003.
9. A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *Proc. of IJCAI 2003*, pages 16–21, 2003.
10. D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of PODS 2004*, pages 241–251, 2004.
11. B. F. Chellas. *Modal Logic: An introduction*. Cambridge University Press, 1980.
12. O. M. Duschka, M. R. Genesereth, and A. Y. Levy. Recursive query plans for data integration. *J. of Logic Programming*, 43(1):49–73, 2000.
13. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.
14. R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. In *Proc. of PODS 2003*, pages 90–101, 2003.
15. R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *Proc. of PODS'83*, pages 352–365, 1983.
16. E. Franconi, G. Kuper, A. Lopatenko, and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *Proc. of the VLDB International Workshop On Databases, Information Systems and Peer-to-Peer Computing*, 2003.
17. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of the 5th Logic Programming Symposium*, pages 1070–1080. The MIT Press, 1988.
18. S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *Proc. of WebDB 2001*, 2001.
19. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE 2003*, pages 505–516, 2003.
20. A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
21. A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *Proc. of ACM SIGMOD*, pages 325–336, 2003.
22. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.

23. H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.
24. V. Lifschitz. Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.
25. J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy. Representing and reasoning about mappings between domain models. In *Proc. of AAAI 2002*, pages 80–86, 2002.
26. R. Rosati. Reasoning about minimal belief and negation as failure. *J. of Artificial Intelligence Research*, 11:277–300, 1999.
27. I. Tatarinov and A. Halevy. Efficient query reformulation in peer data management. In *Proc. of ACM SIGMOD*, 2004.
28. J. D. Ullman. *Principles of Database and Knowledge Base Systems*, volume 1. Computer Science Press, 1988.