On Reconciling Data Exchange, Data Integration, and Peer Data Management

Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma
<degiacomo,lembo,lenzerini,rosati>@dis.uniroma1.it

ABSTRACT

Data exchange and virtual data integration have been the subject of several investigations in the recent literature. At the same time, the notion of peer data management has emerged as a powerful abstraction of many forms of flexible and dynamic data-centered distributed systems. Although research on the above issues has progressed considerably in the last years, a clear understanding on how to combine data exchange and data integration in peer data management is still missing. This is the subject of the present paper. We start our investigation by first proposing a novel framework for peer data exchange, showing that it is a generalization of the classical data exchange setting. We also present algorithms for all the relevant data exchange tasks, and show that they can all be done in polynomial time with respect to data complexity. Based on the motivation that typical mappings and integrity constraints found in data integration are not captured by peer data exchange, we extend the framework to incorporate these features. One of the main difficulties is that the constraints of this new class are not amenable to materialization. We address this issue by resorting to a suitable combination of virtual and materialized data exchange, showing that the resulting framework is a generalization of both classical data exchange and classical data integration, and that the new setting incorporates the most expressive types of mapping and constraints considered in the two contexts. Finally, we present algorithms for all the relevant data management tasks also in the new setting, and show that, again, their data complexity is polynomial.

Categories and Subject Descriptors: H.2 [Database Management]: Heterogeneous Databases; F.2 [Analysis of Algorithms and Problem Complexity]: General; H.2 [Database Management]: Systems

General Terms: Theory, Algorithms.

Keywords: Data Integration, Data Exchange, Peer-to-Peer.

1. INTRODUCTION

Data exchange [25, 14] and virtual data integration [27] have been extensively studied in the last years to address the issue of data residing in independent data sources that need to be moved to, or accessed through, a new data schema (called target schema or global schema in the two contexts, respectively). The target/global schema and the sources are related via mappings. While in data exchange the focus is on materializing data in the target schema through the mapping, in data integration no actual exchange of data is generally needed, and the database conforming to the global schema is virtual.

Recently, the notion of peer data management (PDM) system has emerged as a powerful abstraction for many forms of flexible and dynamic data-centered distributed systems [6, 22]. PDM systems are characterized by a set of autonomous nodes (called, indeed, peers) that hold data and that are linked to other nodes by means of so-called peer-to-peer (P2P) mappings.

Although research in all the three above mentioned areas has progressed considerably in the last years, a clear understanding on how to combine data exchange and virtual data integration in peer data management is still missing. In this paper we address this subject by providing a new framework for P2P data exchange and data integration, which admits traditional data integration and data exchange frameworks as special cases.

We start our investigation by first concentrating on P2P data exchange, revisiting the basic data exchange definitions in this new setting, providing new algorithms for all the relevant data exchange tasks, and showing that they are polynomial in data complexity (i.e., the complexity computed only w.r.t. the size of the underlying database instance). Then, based on the motivation that typical mappings and integrity constraints found in data integration are not captured by peer data exchange, we extend the framework to incorporate these features. We present algorithms for the relevant data management tasks also in the new setting, and show that, again, their data complexity is polynomial.

Related Work. The basic notions of data exchange and its first formalization were given in [14]. In particular, a *solution* to the data exchange problem for a given source instance is a finite target instance that, together with the source instance, satisfies both *target dependencies* and *source-to-target dependencies*. Target dependencies are tuple generating dependencies (TGDs) and equality generating dependencies (EGDs), whereas source-to-target dependencies are TGDs mapping the source schema to the target schema. The semantics of the queries posed over the target schema is given in terms of the *certain answers*, i.e., those answers that are returned by evaluating the query over each solution. A *universal solution* is a special solution that is homomorphic to every possible solution. Universal solutions are particularly important in data exchange, since, as shown in [14], the certain answers to a union of conjunctive queries q can be obtained by evaluating q over any universal solution. Notably, in the case in which target TGDs belong to the class of *weakly-acyclic* TGDs, a class of TGDs which admits limited forms of cycles among different relation arguments, a *canonical universal solution* can be computed in polynomial time in data complexity. Consequently, also query answering in this setting is polynomial w.r.t. the size of the data. The importance of imposing weak-acyclicity on the target TGDs, is also investigated in [26], where it is shown that it is possible to construct an instance of the data exchange problem with a single non-weakly-acyclic TGD on the target schema for which such a problem is undecidable. With the aim of identifying the "smallest" universal solution (and thus optimizing materialization of data), [15] introduced the notion of core. The *core* of a universal solution A is the smallest subset C of A such that A has an homomorphism to C. The cores of all universal solutions are mutually isomorphic and are in turn universal solutions. The core is therefore identified as the best solution for the data exchange problem. A polynomial-time algorithm is given in [15] for core computation in a setting where target dependencies are only EGDs. A notable result in [20] shows that computing the core can be done in polynomial time in the presence of both weakly-acyclic TGDs and EGDs on the target schema. Previous results in this direction had been presented in [19]. Other works on data exchange studied, respectively, query answering (by first-order rewriting) for first-order logic (FOL) queries [3], schema mapping compositions [16], exchange of XML documents when both the source and the target schemas are XML DTDs [5], and relationship between data exchange and incomplete information [28]. We point out that some of the motivations in [28] are also at the basis of the semantics given in the present work.

Virtual data integration has similar logical foundations to those of data exchange [27]. Again, mappings are specified in terms of TGDs between the source and the global schema (although several limitations on the form of such dependencies have also been considered in the literature), and the semantics of data integration systems and query answering is as in data exchange. Among the papers addressing virtual data integration (we refer the reader to [27, 24] for a picture on the subject), particularly interesting for the present work are those facing the problem of integrating data in the presence of powerful forms of integrity constraints specified on the global schema. In particular, we recall the work done in [8] on data integration in the presence of keys and foreign keys, where an algorithm for computing the certain answers to unions of conjunctive queries is provided. A more general result was given in [10], where a practical algorithm based on query rewriting is proposed for answering unions of conjunctive queries in the presence of key dependencies and *non-key-conflicting* inclusion dependencies (IDs). Non-key-conflicting IDs generalize foreign keys, while preventing propagation of keys between different relations. As shown in [10] and [9], query answering in such a setting can be solved in time polynomial in data complexity, whereas as soon as we allow for IDs outside the class of non-key-conflicting IDs, query answering (in fact, already logical implication) becomes undecidable. Notably, as shown in [30], logical implication (and therefore query answering) for the class of non-key-conflicting IDs becomes undecidable if we admit only finite database instances, whereas it remains decidable for the (strict) subclass of non-key-conflicting IDs, called *foreign key dependencies*, which is still a generalization of foreign keys.

Peer data management has recently been investigated in several papers, and techniques have been provided for evolving from basic P2P systems supporting only file exchanges to more complex systems supporting the integration and exchange of structured contents [22, 6, 21, 12, 11, 17, 31, 4, 13]. Data integration in such systems does not require to replicate in other peers data stored in one peer, 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 P2P mappings. As for the semantics of P2P data integration, the usual approach is to adopt a first-order logic interpretation of P2P mappings (followed, e.g., by [21, 22, 6]). However, the presence of cycles in the P2P mappings poses challenging problems (in particular query answering turns out to be undecidable, as shown in [22]). Therefore, some approaches have imposed limitations on the form of P2P mappings in order to allow for decidable query answering. Other works [12, 11, 17] have argued that such limitations are unrealistic in a fully decentralized setting, and have proposed a weaker semantics for mappings, allowing for both a better modelling of the modular structure of the system, and decidable (even polynomially tractable w.r.t. data complexity) query answering.

Data exchange in the context of peer data management is still largely unexplored. Analogously to traditional data exchange, the focus in this scenario is on materializing the data flowing from one peer to another. Such a problem has been studied in [18] in a setting in which only two peers (source and target) interact, and mappings from the target peer to the source peer are interpreted as integrity constraints. Notably, checking the existence of solutions, as well as query answering, is shown to be intractable, even in the absence of integrity constraints in the target schema. In [7] data exchange is studied, but mainly under the perspective of repairing data in the presence of violations of integrity constraints.

Contributions. As we said before, our goal is both to study P2P data exchange, and to investigate the possibility of combining data exchange and virtual data integration in peer data management. The results of our investigation can be summarized as follows.

1. We extend traditional data exchange to P2P data exchange (Section 2), under a semantic characterization of P2P mappings coherent with the epistemic semantics given in [12]. We show that our formalization is a generalization of traditional data exchange, in the sense that it fully captures the framework of [14] for the case in which a P2P data exchange system collapses to a classical data exchange system.
2. We study relevant data exchange tasks (Section 3), i.e., computation of universal solutions and computation of the core, in the P2P setting, and we show that, by virtue of our semantic characterization, and of the usual restrictions on the form of integrity constraints, such problems are solvable in polynomial-time w.r.t. data complexity.
3. We combine data exchange and virtual data integration in the P2P setting (Section 4), based again on our semantic interpretation of P2P mappings. More precisely, we incorporate in our framework new forms of schema dependencies, typical of virtual data integration [9, 30], together with virtual mappings, which do not impose any actual exchange of data between peers, but are used in both the computation of the core and query answering. We discuss the relationship between our P2P framework and classical data integration for the case in which no exchange of data is required, i.e., integration of data is purely virtual.
4. We show that in our P2P data exchange and integration framework, the relevant data exchange and data integration tasks still remain polynomial in data complexity, again under suitable restrictions on the form of integrity constraints (Section 5).

2. P2P DATA EXCHANGE

We first analyze the case of pure P2P data exchange systems. We define syntax, semantics, and the main notions of admissible state, universal solution, core, and certain answers.

2.1 Syntax

We start with a preliminary definition. Given a relational signature S:

- a *tuple-generating dependency (TGD)* is an assertion of the form $q_i \rightarrow q_j$, where q_i and q_j are conjunctive queries (CQs) of the same arity over the signature S;

- an *equality-generating dependency (EGD)* is an assertion of the form $q(x_1, x_2) \rightarrow x_1 = x_2$, where $q(x_1, x_2)$ is a conjunctive query over the signature S, and x_1 and x_2 are the distinguished variables of q.

Intensional level. A *P2P data exchange system* (*PDE*-system) is a triple $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E \rangle$ where:

- $\mathcal{P}$ is a set of peers, each with a relational signature. We assume that the signatures of the peers are pairwise disjoint;
- $\mathcal{C}_E$ is a set of *local constraints* (or simply *constraints*), i.e., TGDs and EGDs over the signature of a single peer;
- $\mathcal{M}_E$ is a set of *P2P mappings*, i.e., a set of TGDs of the form $q_i \rightarrow q_j$, where q_i is a CQ over the signature of a peer P_i and q_j is a CQ of the same arity of q_i over the signature of a peer P_j.

Extensional level. We define a countably infinite set of constant symbols $\mathcal{C}$ (representing ordinary values) and a countably infinite set of constant symbols $\mathcal{N}$ (representing labeled null values). We assume that $\mathcal{C} \cap \mathcal{N} = \emptyset$ and define $\Gamma = \mathcal{C} \cup \mathcal{N}$. We assume that Γ is totally ordered.

An *indefinite instance* (or simply *instance*) for a signature S is a (not necessarily finite) set of facts built upon the relation symbols in S and the constant symbols in Γ. Given an instance B, we denote by $nulls(B)$ the set of constants from $\mathcal{N}$ occurring in B. An instance B is *definite* if $nulls(B) = \emptyset$. In the following, we use the symbol B to denote a generic instance and the symbol D to denote a definite instance.

We denote by $S(\mathcal{P})$ the signature obtained as the union of the signatures of the peers in $\mathcal{P}$.

A *state* B for the *PDE*-system $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E \rangle$ is an instance for $S(\mathcal{P})$. Such a state is called *definite* if it is a definite instance for $S(\mathcal{P})$. Given a state B, we denote by Γ_B the set of values from Γ occurring in B.

Queries. A *query* over a *PDE*-system $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E \rangle$ is a *union of conjunctive queries (UCQ)* over the signature of a peer $P \in \mathcal{P}$.

2.2 Semantics

Given a definite instance D for a signature S and a query q over S, we denote by $Eval(q, D)$ the standard (i.e., under CWA) evaluation of q in D.

Given two states B_1, B_2 for $\mathcal{S}$, a *homomorphism* from B_1 to B_2 is a function $h : \Gamma \rightarrow \Gamma$ such that

- $h(c) = c$, for every constant $c \in \mathcal{C}$;
- $r(h(t_1), \ldots, h(t_n)) \in B_2$, for every fact $r(t_1, \ldots, t_n) \in B_1$.

We say that two states B_1, B_2 for $\mathcal{S}$ are *homomorphically equivalent* if there exists a homomorphism from B_1 to B_2 and vice-versa.

Let B a state for $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E \rangle$, a definite instance D for $\mathcal{S}$ is called an *instantiation* of B if there exists an injective function $f : \Gamma_B \rightarrow \mathcal{C}$ such that $f(c) = c$, for every constant $c \in \Gamma_B \cap \mathcal{C}$, and $D = \{r(f(t_1), \ldots, f(t_n)) \mid r(t_1, \ldots, t_n) \in B\}$. In other words, the function f maps: (i) every constant value appearing in B to itself; (ii) every null value appearing in B to a distinct constant value not appearing in B. Also, Then we say that:

- a TGD $q_i \rightarrow q_j$ is *satisfied* in D if $Eval(q_i, D) \subseteq Eval(q_j, D)$;
- an EGD $q(x_1, x_2) \rightarrow x_1 = x_2$ is *satisfied* in D if, for each $\langle t_1, t_2 \rangle \in Eval(q, D)$, $t_1 = t_2$.

Observe that P2P mappings are TGDs, hence we could interpret them as above. Since P2P mappings are TGDs, we could adopt the same notion of satisfaction as the one specified above. However, since we are placing no constraints on the topology of the P2P mappings, and therefore we accept arbitrary cyclic configurations of the TGDs in $\mathcal{M}_E$, this would induce to undecidability. In order to regain decidability, here we take up the idea in [12] of weakening the semantics of P2P mappings. In particular, we interpret P2P mappings as containments between certain answers. Notice that such an interpretation is coherent with the idea that peers export certain answers only (cf. [12]).

Let $\mathcal{D}$ be a set of definite instances and let $q_i \rightarrow q_j$ be a TGD. We say that $q_i \rightarrow q_j$ is *CERT-satisfied* in $\mathcal{D}$ if

$$\bigcap_{D \in \mathcal{D}} Eval(q_i, D) \subseteq \bigcap_{D \in \mathcal{D}} Eval(q_j, D)$$

DEFINITION 1. *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E \rangle$ be a PDE-system, B a state for $\mathcal{S}$, and $\mathcal{D}$ a set of definite instances for $S(\mathcal{P})$. We say that $\mathcal{D}$* satisfies $\mathcal{S}$ and B *if:*

1. *for each $D \in \mathcal{D}$ there is a homomorphism from B to D;*
2. *for each $D \in \mathcal{D}$ and for each TGD and EGD ϕ in $\mathcal{C}_E$, ϕ is satisfied in D;*
3. *for each P2P mapping ϕ in $\mathcal{M}_E$, ϕ is CERT-satisfied in $\mathcal{D}$.*

Notice that our notion of *CERT*-satisfaction of P2P mappings applies to a set of definite instances as a whole. Therefore, in our approach, "what satisfies" $\mathcal{S}$ and B is not a single instance, but a set of definite instances. It can be immediately verified that, for every pair of sets $\mathcal{D}$ and $\mathcal{D}'$ that satisfy $\mathcal{S}$ and B, the set $\mathcal{D} \cup \mathcal{D}'$ satisfies $\mathcal{S}$ and B. Consequently, for every *PDE*-system $\mathcal{S}$ and state B, there is a unique maximal set of definite instances that satisfies $\mathcal{S}$ and B. We denote by $Sem(\mathcal{S}, B)$ such a maximal set.

DEFINITION 2. **(Consistent state)** *Let $\mathcal{S}$ be a PDE-system and B a state for $\mathcal{S}$. We say that a state B is $\mathcal{S}$-*consistent *if $Sem(\mathcal{S}, B) \neq \emptyset$.*

DEFINITION 3. **(Admissible state)** *Let $\mathcal{S}$ be a PDE-system and B a state for $\mathcal{S}$. We say that B is $\mathcal{S}$-*admissible *if (i) B is $\mathcal{S}$-consistent, (ii) every instantiation of B belongs to $Sem(\mathcal{S}, B)$.*

Intuitively, an $\mathcal{S}$-admissible state is both $\mathcal{S}$-consistent, and such that all data exchange specified by $\mathcal{S}$ has taken place.

DEFINITION 4. **(Universal solution)** *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E \rangle$ be a PDE-system and B an $\mathcal{S}$-consistent state. We say that a state B' for $\mathcal{S}$ is a* universal $\mathcal{S}$-solution of B *if: (i) B' is $\mathcal{S}$-admissible; (ii) $Sem(\mathcal{S}, B') = Sem(\mathcal{S}, B)$.*

It is immediate to verify that, if B' is a universal $\mathcal{S}$-solution of B, then for each $D \in Sem(\mathcal{S}, B)$ there exists a homomorphism from B' to D.

Informally, a universal $\mathcal{S}$-solution for B is a "correct representative" of all the definite instances in $Sem(\mathcal{S}, B)$, in the sense that a universal $\mathcal{S}$-solution represents the "positive information" that is common to all the definite instances in $Sem(\mathcal{S}, B)$. Observe that, if a state B is $\mathcal{S}$-admissible, then it is trivially a universal $\mathcal{S}$-solution of itself.

The notion of $\mathcal{S}$-core below formalizes a property of "minimality" for a universal $\mathcal{S}$-solution.

DEFINITION 5. **(Core)** *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E \rangle$ be a PDE-system and B an $\mathcal{S}$-consistent state. We say that a state B' for $\mathcal{S}$ is an $\mathcal{S}$-*core of B *if B' is a universal $\mathcal{S}$-solution of B and there exists no proper subset B'' of B' such that B'' is a universal $\mathcal{S}$-solution of B.*

The following propositions are consequences of the above definitions.

PROPOSITION 1. *Let $\mathcal{S}$ be a PDE-system and B an $\mathcal{S}$-consistent state. Then, (i) there exists at least a universal $\mathcal{S}$-solution of B, and (ii) all universal $\mathcal{S}$-solutions of B are homomorphically equivalent.*

PROPOSITION 2. *Let $\mathcal{S}$ be a PDE-system and B an $\mathcal{S}$-consistent state. Then, there exists a unique $\mathcal{S}$-core of B up to isomorphism.*

Based on the above property, in the following we will indicate any $\mathcal{S}$-core of B as *the* $\mathcal{S}$-core of B.

Finally, we define the semantics of queries in $\mathcal{S}$-admissible states.

DEFINITION 6. **(Certain answers)** *Let $\mathcal{S}$ be a PDE-system, B an $\mathcal{S}$-admissible state, and q a query over $\mathcal{S}$. Then, the set of* certain answers *to q in $\mathcal{S}$ and B is defined as follows:*

$$Ans(q, \mathcal{S}, B) = \bigcap_{D \in Sem(\mathcal{S}, B)} Eval(q, D).$$

We now show that answering UCQs in an $\mathcal{S}$-admissible state B reduces to evaluating the query over the state B. To this aim, we introduce a preliminary notion. Given an indefinite instance B for a signature S and a query q over S, we define $Eval_{Null}(q, B)$ as the evaluation of q in B when interpreting all the null values occurring in B as ordinary constant values (i.e., $Eval_{Null}(q, B)$ corresponds to $Eval(q, B)$ when B is considered as a definite instance in which null values are considered as ordinary constant values). Moreover, assuming that q is of arity k, we define $Eval_{Null\downarrow}(q, B) = Eval_{Null}(q, B) \cap \mathcal{C}^k$, i.e., $Eval_{Null\downarrow}(q, B)$ is the set of tuples from $Eval_{Null}(q, B)$ in which null values do not occur. For instance, if $B = \{r(a, n)\}$, with $a \in \mathcal{C}$, $n \in \mathcal{N}$, and $q = \{x, y \mid r(x, y)\}$, then $Eval_{Null}(q, B) = \{\langle a, n\rangle\}$ while $Eval_{Null\downarrow}(q, B) = \emptyset$. With the above notion in place, we can provide the following theorem.

THEOREM 1. *Let $\mathcal{S}$ be a PDE-system and B an $\mathcal{S}$-admissible state. Then, $Ans(q, \mathcal{S}, B) = Eval_{Null\downarrow}(q, B)$.*

EXAMPLE 1. *Consider the PDE-system $\mathcal{S} = \langle\{P_1, P_2\}, \mathcal{C}_E, \mathcal{M}_E\rangle$, where the signature of P_1 consists only of one binary relation symbol R_1, the signature of P_2 consists of the two binary relation symbols R_2 and R_3, $\mathcal{C}_E$ contains the following set of dependencies:*

$$\begin{array}{rcl} \{x \mid R_2(y, x)\} & \rightarrow & \{x \mid R_3(x, z)\} \\ \{y, z \mid R_3(x, y), R_3(x, z)\} & \rightarrow & y = z, \end{array}$$

and $\mathcal{M}_E$ consists of the following set of dependencies:

$$\begin{array}{rcl} \{x \mid R_1(x, y)\} & \rightarrow & \{x \mid R_2(x, z)\} \\ \{x \mid R_3(y, x)\} & \rightarrow & \{x \mid R_1(z, x)\}. \end{array}$$

Assume that $B = \{R_1(a, b), R_2(a, d), R_3(f, b), R_3(h, d)\}$ is a state for $\mathcal{S}$. Then, a universal $\mathcal{S}$-solution of B is $B' = \{R_1(a, b), R_1(x_1, b), R_1(x_2, d), R_2(a, d), R_2(a, x_3), R_3(d, x_4), R_3(f, b), R_3(h, d)\}$, where each x_i denotes a different value from $\mathcal{N}$, whereas the $\mathcal{S}$-core is $C = \{R_1(a, b), R_1(x_2, d), R_2(a, d), R_3(d, x_4), R_3(f, b), R_3(h, d)\}$. Notice that, according to minimality of the core, C does not contain facts $R_1(x_1, b)$ and $R_2(a, x_3)$, since it already contains the ground atoms $R_1(a, b)$ and $R_2(a, d)$. Now, consider the query $q(x) \leftarrow R_1(y, x)$ posed over peer P_1. It is easy to see that $Ans(q, \mathcal{S}, B) = \{b, d\}$. □

2.3 Relationship with classical data exchange

We now show that the setting presented above is in fact a generalization of the "classical" data exchange setting as defined in [14, 15]. We remind the reader that the classical data exchange setting $\mathcal{M}$ is a tuple $(\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ where $\mathbf{S}$ and $\mathbf{T}$ are relational signatures called respectively source schema and target schema, Σ_{st} is a set of source-to-target dependencies (i.e., TGDs of the form $q_s \rightarrow q_t$, where q_s is a CQ over $\mathbf{S}$ and q_t is a CQ over $\mathbf{T}$), and Σ_t is a set of target dependencies (i.e., weakly-acyclic TGDs and EGDs specified over $\mathbf{T}$). In the above setting, given a definite (i.e., without occurrences of null values) and finite instance I for $\mathbf{S}$, the following data exchange problems are studied: (*i*) existence of a solution, i.e., find a finite instance J for $\mathbf{T}$ such that $\langle I, J\rangle$ satisfies (in the sense of [14, 15]) Σ_{st} and Σ_t; (*ii*) find a universal solution for I in $\mathcal{M}$, i.e., a solution that is homomorphic to every possible solution for I in $\mathcal{M}$; (*iii*) find the core of the universal solutions for I in $\mathcal{M}$. The core C of a universal solution J is the smallest subset of J such that J has homomorphism to C; as shown in [15], all cores are identical up to isomorphism, therefore they are referred to as the core of the universal solutions for I in $\mathcal{M}$.

Given one such data exchange setting $\mathcal{M}$, we define the corresponding *PDE*-system $\mathcal{S}_{\mathcal{M}}$ as the triple $\langle\mathcal{P}, \mathcal{C}_E, \mathcal{M}_E\rangle$, where (*i*) $\mathcal{P}$ is composed of only two peers: P_s with signature $\mathbf{S}$, and P_t with signature $\mathbf{T}$; (*ii*) $\mathcal{C}_E = \Sigma_t$; (*iii*) $\mathcal{M}_E = \Sigma_{st}$. Moreover, a definite and finite instance I for $\mathbf{S}$ simply corresponds to a definite and finite instance for the signature of P_s. We point out that $\mathcal{S}_{\mathcal{M}}$ is a *PDE*-system of special kind: $\mathcal{P}$ contains only P_s and P_t, the TGDs in $\mathcal{C}_E$ are weakly-acyclic, no local constraints on the signature of P_s can be defined, and $\mathcal{M}_E$ contains only P2P mappings from P_s to P_t (i.e., P2P mappings from P_t to P_s are not allowed). Furthermore, the initial instance I corresponds to a state for $\mathcal{S}_{\mathcal{M}}$ where neither occurrences of null values nor facts concerning the signature of P_t are allowed.

We now show the correspondence between the problems of existence of a solution, computing a universal solution, and computing the core in the classical data exchange setting with the problems of checking $\mathcal{S}$-consistency of a state, computing a universal $\mathcal{S}$-solution, and computing an $\mathcal{S}$-core, respectively.

THEOREM 2. *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ be a classical data exchange setting, as defined in [14, 15], I a finite definite instance for $\mathbf{S}$, and $\mathcal{S}_{\mathcal{M}}$ the PDE-system corresponding to $\mathcal{M}$. Then:*

1. *I is an $\mathcal{S}$-consistent state for $\mathcal{S}_{\mathcal{M}}$ iff there exists a solution (in the sense of [14]) for I in $\mathcal{M}$;*
2. *if I' is a finite universal $\mathcal{S}$-solution of I, then $(I' - I)$ is a universal solution (in the sense of [14]) for I in $\mathcal{M}$, and vice-versa, if J is a universal solution for I in $\mathcal{M}$, then $I \cup J$ is a (finite) universal $\mathcal{S}$-solution of I;*
3. *if I' is the $\mathcal{S}$-core of I, then $I' - I$ is the core (in the sense of [15]) of the universal solutions for I in $\mathcal{M}$, and vice-versa, if J is the core of the universal solutions for I in $\mathcal{M}$, then $I \cup J$ is the $\mathcal{S}$-core of I.*

The above properties show that the notions of universal $\mathcal{S}$-solution and of $\mathcal{S}$-core of a state provide a generalization to the P2P setting of the notions of universal solution and core of classical data exchange systems. Note also that, in this light, Proposition 1 and 2 above are generalizations of well-known properties of universal solutions and core in the classical data exchange setting [14, 15].

Such a correspondence may look surprising at first, considering the different interpretation of the P2P mappings (i.e., source-to-target TGDs) in the two settings. However, since the source instance B is a definite instance, then answers and certain answers to CQs in B coincide, and therefore the difference between the two semantics does not show up.

3. REASONING IN P2P DATA EXCHANGE

In this section we consider reasoning in *PDE*-systems. In particular, we first define the E-CHASE for a given finite state B for a *PDE*-system $\mathcal{S}$, and prove some important semantic properties of the E-CHASE. Then, we turn our attention to *weakly-acyclic PDE*-systems, a class of *PDE*-systems where the constraint language is suitably restricted, and characterize the complexity of state consistency, state admissibility, computing universal solutions, query answering, and computing the core for these *PDE*-systems.

3.1 The E-CHASE

We now introduce the notion of E-CHASE, which extends the classical chase procedure for TGDs and EGDs to the treatment of P2P mappings. Remember that P2P mappings are TGDs with a special interpretation based on the notion of *CERT*-satisfaction. In the following definition, with a little abuse of notation, we write $q(\vec{x}, \vec{y})$ to denote a conjunctive query whose distinguished variables are those of $\vec{x}$, and whose existential variables are those of $\vec{y}$. Moreover, $Ch_{\mathcal{S},B}(i)[t_1 \leftarrow t_2]$ denotes the state obtained from $Ch_{\mathcal{S},B}(i)$ by replacing each occurrence of t_1 with t_2.

DEFINITION 7. (E-CHASE) *Let $\mathcal{S}$ be a PDE-system and B a finite state for $\mathcal{S}$. Let $Ch_{\mathcal{S},B}(0) = B$. For every positive integer i, let $Ch_{\mathcal{S},B}(i+1)$ be the state obtained from $Ch_{\mathcal{S},B}(i)$ by applying the following chase rules (fixing an arbitrary well-founded order of application on the rules and of the null values in $\mathcal{N}$):*

```
1. TGD-rule:
   if $q_1(\vec{x},\vec{y}) \rightarrow q_2(\vec{x},\vec{z}) \in \mathcal{C}_E$
      and $\vec{t} \in Eval_{Null}(q_1, Ch_{\mathcal{S},B}(i))$
      and $\vec{t} \notin Eval_{Null}(q_2, Ch_{\mathcal{S},B}(i))$
   then $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i) \cup \{q_2(\vec{t},\vec{n})\}$, where $\vec{n}$ is
   the $k$-tuple of the first $k$ null values that do not appear in
   $Ch_{\mathcal{S},B}(i)$;
2. EGD-rule:
   if $q(x_1,x_2) \rightarrow x_1 = x_2 \in \mathcal{C}_E$
      and $\langle t_1,t_2\rangle \in Eval_{Null}(q, Ch_{\mathcal{S},B}(i))$
      and $t_1 \neq t_2$
   then if $t_1 \in \mathcal{N}$
         then $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i)[t_1 \leftarrow t_2]$
         else if $t_2 \in \mathcal{C}$
              then $Ch_{\mathcal{S},B}(i+1) :=$ FAIL
              else $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i)[t_2 \leftarrow t_1]$;
3. P2P-mapping-rule:
   if $q_1(\vec{x},\vec{y}) \rightarrow q_2(\vec{x},\vec{z}) \in \mathcal{M}_E$
      and $\vec{t} \in Eval_{Null\downarrow}(q_1, Ch_{\mathcal{S},B}(i))$
      and $\vec{t} \notin Eval_{Null\downarrow}(q_2, Ch_{\mathcal{S},B}(i))$
   then $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i) \cup \{q_2(\vec{t},\vec{n})\}$, where $\vec{n}$ is
   the $k$-tuple of the first $k$ null values that do not appear in
   $Ch_{\mathcal{S},B}(i)$.
```

Finally, we define E-CHASE$(\mathcal{S}, B)$ = FAIL *if there exists i such that $Ch_{\mathcal{S},B}(i)$ = FAIL, otherwise* E-CHASE$(\mathcal{S}, B) = \bigcup_{i\in\mathbb{N}} Ch_{\mathcal{S},B}(i)$.

We now show how to exploit E-CHASE for reasoning over *PDE*-systems.

THEOREM 3. *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E\rangle$ be a PDE-system and B a finite state for $\mathcal{S}$. Then,*

- *B is $\mathcal{S}$-consistent iff* E-CHASE$(\mathcal{S}, B) \neq$ FAIL*;*
- *if* E-CHASE$(\mathcal{S}, B) \neq$ FAIL*, then* E-CHASE$(\mathcal{S}, B)$ *is a universal $\mathcal{S}$-solution of B;*
- *B is $\mathcal{S}$-admissible iff* E-CHASE$(\mathcal{S}, B) = B$.

3.2 Reasoning in weakly-acyclic *PDE*-systems

So far, we have not imposed any limitation on the form of the constraints occurring in *PDE*-systems. In particular, general *PDE*-systems allow arbitrary TGDs and EGDs. Notice, however, that in the presence of arbitrary TGDs, a finite universal solution in general does not exist [14, 26]. On the other hand, we are particularly interested in those *PDE*-systems $\mathcal{S}$ that, given a finite state B for $\mathcal{S}$, admit a finite universal $\mathcal{S}$-solution of B. To this aim, in the line of [14, 15], we restrict the class of TGDs allowed in $\mathcal{S}$ to the class of *weakly-acyclic* TGDs.

We call *weakly-acyclic PDE-system* a *PDE*-system $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E\rangle$ such that the TGDs in $\mathcal{C}_E$ are weakly-acyclic.

Let us consider a weakly-acyclic *PDE*-system $\mathcal{S}$ and a finite state B for $\mathcal{S}$. Based on the well-known complexity results for the chase of weakly-acyclic TGDs and EGDs, and considering that every P2P-mapping rule fires only for tuples of ordinary constants already appearing in B (and hence each such rule can fire at most a polynomial number of times w.r.t. data complexity), the tractability result of the chase of weakly-acyclic TGDs and EGDs extends to our notion of chase (E-CHASE). Consequently, the following upper bounds hold.

THEOREM 4. *Let $\mathcal{S}$ be a weakly-acyclic PDE-system and B a finite state for $\mathcal{S}$. Then,*

- *deciding whether B is $\mathcal{S}$-consistent can be done in polynomial time w.r.t. data complexity;*
- *deciding whether B is $\mathcal{S}$-admissible can be done in polynomial time w.r.t. data complexity;*
- *if B is $\mathcal{S}$-consistent, then a universal $\mathcal{S}$-solution of B can be computed in polynomial time w.r.t. data complexity.*
- *if B is $\mathcal{S}$-admissible, then query answering can be solved in logarithmic space w.r.t. data complexity.*

We remark that, even in weakly-acyclic *PDE*-systems, the set of TGDs in $\mathcal{C}_E$ and $\mathcal{M}_E$ is in general non-weakly-acyclic (see, e.g., Example 1). Thus, even under the restriction that $\mathcal{C}_E$ is weakly-acyclic, a classical first-order interpretation of the P2P mappings would lead to a situation where a finite universal $\mathcal{S}$-solution does not exist (and all the reasoning tasks studied above are undecidable). Conversely, the $CERT$-satisfaction of P2P mappings required in our framework guarantees the existence of a finite universal $\mathcal{S}$-solution.

We now consider the problem of computing the $\mathcal{S}$-core in weakly-acyclic *PDE*-systems. It is easy to see that the problem of computing the $\mathcal{S}$-core of a *PDE*-system starting from an *arbitrary* finite state is in general NP-hard: such hardness is due to the fact that a state may contain arbitrary combinations of null values (which implies that the so-called *block size* of B may be not bounded [20]). So we look for significant classes of states which admit a polynomial (w.r.t. data complexity) computation of the $\mathcal{S}$-core. To this aim, we introduce the class of ground $\mathcal{S}$-evolutions.

DEFINITION 8. **(Ground $\mathcal{S}$-evolution)** *Let $\mathcal{S}$ be a weakly-acyclic PDE-system. We say that a finite state B is a* ground $\mathcal{S}$-evolution *if there exists a sequence of finite $\mathcal{S}$-consistent states $B_1, \ldots, B_n$ and a sequence of finite definite instances $D_1, \ldots, D_n$ such that:*

1. *B_1 is a definite state;*
2. *for each i such that $1 \leq i \leq n-1$, B_{i+1} is an $\mathcal{S}$-core of $B_i \cup D_i$;*
3. *$B = B_n \cup D_n$.*

Informally, a ground $\mathcal{S}$-evolution is a state that is obtained by (iteratively) adding ground facts to an $\mathcal{S}$-core of a previous state. From the practical viewpoint, this is a very interesting class of states, since each such state is obtained from a definite state by repeatedly: (i) inserting new ground data; (ii) computing the new $\mathcal{S}$-core.

To compute an $\mathcal{S}$-core of a ground $\mathcal{S}$-evolution B, we essentially re-use the algorithm FINDCORE for computing cores in a classical data exchange setting (see [20]). This algorithm is constituted by two parts. In the first part (steps 1 and 2), the algorithm computes the chase with respect to a set of weakly-acyclic TGDs and EGDs, starting from a set of facts. Essentially, such a chase is analogous to our procedure for defining E-CHASE *without* the P2P-mapping-rule, with the following differences: (i) it is based on an encoding of EGDs through new TGDs (and hence it only considers TGDs); (ii) since adding such new TGDs makes the whole set of TGDs non-weakly-acyclic, to guarantee termination it imposes that the order of application of rules in the chase must be *nice*, i.e., must satisfy a given condition (we refer to [20] for more details).

In the following, by PDEFINDCORE we indicate a slight modification of the algorithm in [20], which: (i) starts from a state (instead of a definite instance); (ii) computes a universal solution (steps 1 and 2 of the algorithm) extending the chase procedure defined in [20] with the P2P-mapping-rule of Definition 7 (which can be executed in any order).

THEOREM 5. *Let $\mathcal{S}$ be a weakly-acyclic PDE-system and B a finite state for $\mathcal{S}$. If B is a ground $\mathcal{S}$-evolution, then* PDEFINDCORE$(\mathcal{S}, B)$ *is an $\mathcal{S}$-core of B.*

Proof (sketch). The key properties for the correctness of the algorithm PDEFINDCORE are the following: (i) the fact that the state B is a ground $\mathcal{S}$-evolution implies that the *block size* of B is bounded to a value independent of the size of the data; (ii) the addition of the P2P-mapping-rule does not change the bound on the maximal *depth* [20] of null values in the chase, since by definition such a rule does *not* propagate null values. This allows to extend the correctness proof of [20] to PDEFINDCORE. □

From the complexity of E-CHASE (Theorem 4) and of the algorithm FINDCORE [20], we can derive the following complexity characterization for computing the S-core of an $\mathcal{S}$-consistent state.

THEOREM 6. *Let $\mathcal{S}$ be a weakly-acyclic PDE-system and B a finite state that is both $\mathcal{S}$-consistent and a ground $\mathcal{S}$-evolution. The $\mathcal{S}$-core of B can be computed in time polynomial w.r.t. data complexity.*

Proof (sketch). The key property is that, since the P2P-mapping-rule only propagates constant values, it follows that each P2P mapping can fire this rule at most n^k times, where n is the number of constant values occurring in B, and k is the arity of the P2P mapping. Thus, adding this rule does not affect the polynomial upper bound of the chase, which in turn (based on the complexity of FINDCORE shown in [20]) implies the polynomial upper bound of the algorithm PDEFINDCORE. □

4. ADDING VIRTUAL MAPPINGS AND VIRTUAL CONSTRAINTS

The framework proposed so far does not capture the notion of mappings as studied in virtual data integration. A virtual mapping has the same syntactic form as a data exchange mapping. However, while a mapping from peer P_1 to peer P_2 of the latter kind is satisfied if the data involved in the mapping have been exchanged from P_1 to P_2, a virtual mapping is not interpreted as a condition to be enforced on the data (which are only virtual), but rather as a correspondence that is to be used when computing certain answers to queries. This kind of mappings is typical of virtual data integration [27], and is also used in commercial data federation tools such as IBM DB2 Information Integrator or Oracle 10g Information Integration. The interest in virtual mappings stems from the fact that, in data integration, the global schema represents a virtual database, and the interpretation of the mappings between the sources and the global schema (or, between two peers) should take into account such virtual nature of data. This means that a state should be considered admissible even if its data do not explicitly satisfy a virtual mapping, contrary to the idea behind data exchange mappings. At the same time, when computing the certain answers, we should obviously consider only those databases that satisfy all mappings (including the virtual ones).

Another concept that is typical of data integration is the one of virtual constraint. Since the global database of a data integration system is virtual, if the global schema contains integrity constraints, they should be considered "virtual", i.e., they represent conditions that all databases conforming to the schema should satisfy, rather than conditions to be enforced on actual data. This means that the source data, the virtual mappings, and the global schema constitute a database with incomplete information (see [27, 23, 1]), i.e., they are an abstraction for a set of databases, and the constraints in the schema are part of the specification of such set of global databases.

The challenge we face in this section and in Section 5 is whether we can add virtual mappings and virtual constraints to peer data exchange systems. To this aim, in the rest of this section we present the notion of peer data exchange and integration system, while in the next section we present techniques for all relevant data management tasks in these systems.

4.1 Syntax

We present a new notion of peer data management system, aiming at combining data exchange and data integration. According to the discussion above, a system of this type has two types of constraints and two types of mappings. Constraints of the two classes are called e-constraints and i-constraints, respectively, where the former have the same interpretation as in Section 2, and the latter are virtual constraints. Mappings of the two classes are called e-mappings, i.e., data exchange mappings (cf. Section 2), and i-mappings, i.e., virtual (data integration) mappings, respectively.

A P2P data exchange and integration (*PDEI*) system is a 5-tuple $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ where:

- $\mathcal{P}$ is a set of peers (each with a relational signature);
- $\mathcal{C}_E$ is a set of *e-constraints*, and $\mathcal{C}_I$ is a set of *i-constraints*; each constraint of any of the two classes is either a TGD or an EGD over a single peer;
- $\mathcal{M}_E$ (*e-mappings*) and $\mathcal{M}_I$ (*i-mappings*) are two sets of P2P mappings.

Note that in the general definition above we do not pose any limitation on constraints. Suitable limitations will be introduced in Section 5 to ensure decidability of data management tasks.

4.2 Semantics

We now define the semantics of *PDEI*-systems. As in the case of *PDE*-systems, the semantic structures for these systems are sets of definite instances.

DEFINITION 9. *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ be a PDEI-system. We say that a set of definite instances $\mathcal{D}$* satisfies $\mathcal{S}$ and B *if:*

1. *for each $D \in \mathcal{D}$ there is a homomorphism from B to D;*
2. *for each $D \in \mathcal{D}$ and for each TGD and EGD ϕ in $\mathcal{C}_I \cup \mathcal{C}_E$, ϕ is satisfied in D;*
3. *for each mapping ϕ in $\mathcal{M}_I \cup \mathcal{M}_E$, ϕ is CERT-satisfied in $\mathcal{D}$.*

It can be immediately verified that, analogously to the case of *PDE*-systems, for every *PDEI*-system $\mathcal{S}$ and state B, there is a unique maximal set of definite instances that satisfies $\mathcal{S}$ and B, and we denote by $Sem(\mathcal{S}, B)$ such a maximal set. The definitions of $\mathcal{S}$*-consistent* state, *universal* $\mathcal{S}$*-solution*, $\mathcal{S}$*-core*, and *certain answers* are identical to the corresponding definitions given for *PDE*-systems, except that they are based on the new definition of $Sem(\mathcal{S}, B)$, and on the notion of $\mathcal{S}$-admissible state, which is more involved, and is discussed below.

As in the case of *PDE*-systems, a state should be considered admissible if all the data exchange required by e-constraints and e-mappings has taken place. However, for checking whether such exchange has taken place, we should consider not only the actual data at the peers, but also the facts implied by the virtual mappings and the virtual constraints. Consider, for example, the e-mapping $e = q_1 \rightarrow q_2$ from peer P_1 to peer P_2, where q_2 is constituted by one atom with predicate p, and suppose that a tuple t satisfies q_1 in peer P_1. Obviously, the e-mapping e specifies that t should also satisfy p. Should this mean that the fact $p(t)$ should be in P_2? We argue that this is not necessary. Indeed, in order to fulfill the data exchange specification represented by e, it is sufficient that $p(t)$ is logically implied by the virtual mappings to P_2 and the virtual constraints in P_2 (together with the data stored in the various peers). This means that, for a state to be admissible, all the data exchange required by e-constraints and e-mappings has taken place, modulo the implicit knowledge represented by the virtual mappings and constraints. In other words, for a state B to be admissible, the state B' obtained from B by adding all the consequence of B according to virtual mappings and constraints must satisfy all e-mappings and e-constraints.

To come up with the right definition according to the above intuition, we need to introduce some preliminary notions. If $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ is a *PDEI*-system, we denote by S^E the *PDE*-system $\langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \emptyset, \emptyset \rangle$, i.e., the system obtained from $\mathcal{S}$ by dropping both virtual constraints and virtual mappings, and we denote by S^I the system $\langle \mathcal{P}, \emptyset, \emptyset, \mathcal{C}_I, \mathcal{M}_I \rangle$, i.e., the system obtained from $\mathcal{S}$ by dropping both e-constraints and e-mappings. Moreover, we denote by $S^{I \rightarrow E}$ the *PDE*-system $\langle \mathcal{P}, \mathcal{C}_I, \mathcal{M}_I, \emptyset, \emptyset \rangle$, i.e., the system obtained from $\mathcal{S}$ by dropping both e-constraints and e-mappings, and by turning the virtual constraints $\mathcal{C}_I$ and virtual mappings $\mathcal{M}_I$ into e-constraints and e-mappings, respectively.

DEFINITION 10. **(Admissible state for *PDEI*-systems)** *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ be a PDEI-system and B a state for $\mathcal{S}$. We say that B is $\mathcal{S}$-*admissible *if (i) B is $\mathcal{S}$-consistent, and (ii) there exists a universal $\mathcal{S}^{I \rightarrow E}$-solution of B that is $\mathcal{S}^E$-admissible.*

Note that, if for a system $\mathcal{S}$, the sets $\mathcal{C}_I$ and $\mathcal{M}_I$ are empty, then trivially B itself is a universal $\mathcal{S}^{I \rightarrow E}$-solution of B and hence the above definition coincides with the definition of admissibility presented in Section 2.

The following theorem shows that, if we are in an $\mathcal{S}$-admissible state, and we have to compute the certain answers to a union of conjunctive queries, then we can concentrate on virtual mappings and virtual constraints only.

THEOREM 7. *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{C}_I, \mathcal{M}_E, \mathcal{M}_I \rangle$ be a PDEI-system, B an $\mathcal{S}$-admissible state, and q a query over $\mathcal{S}$. Then, $Ans(q, \mathcal{S}, B) = Ans(q, \mathcal{S}^I, B)$.*

4.3 Relationship with classical data integration

We briefly comment on the relationship between *PDEI*-systems and traditional virtual data integration. We remind the reader that a data integration system $\mathcal{J}$ is characterized by a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is the global schema (possibly with integrity constraints), $\mathcal{S}$ is the source schema, and $\mathcal{M}$ is the set of mappings, which specify the relationship between the global and the source schema [27]. The most general form of mappings considered in data integration is called GLAV, and directly corresponds to the notion of TGD. Given a database D for $\mathcal{S}$, the set of databases for $\mathcal{G}$ which satisfy the constraints in $\mathcal{G}$ and are coherent (in the sense of [27]) with D and $\mathcal{M}$, denoted $Mod(\mathcal{J}, D)$, represents the semantics of $\mathcal{J}$. Indeed, the notion of certain answer to a query is based on $Mod(\mathcal{J}, D)$: a tuple of constants is a certain answer to a query q posed to $\mathcal{J}$ (i.e., a query over $\mathcal{G}$) if it is an answer to q in every database in $Mod(\mathcal{J}, D)$. Observe that the semantics of mappings is the usual one in FOL, whereas in the *PDEI*-systems defined here, virtual mappings are interpreted based on the notion of CERT-satisfaction. Thus, it might seem that a data integration system cannot be rendered as a *PDEI*-system. However, as observed for the case of classical data exchange in Section 2.3, the source database D is a database, i.e., a finite definite instance, in our terminology, and hence answers and certain answers to CQs in D coincide. Based on this observation, we show below that the notion of *PDEI*-system is a generalization of virtual data integration.

Given one such data integration system $\mathcal{J}$, we define a corresponding *PDEI*-system $\mathcal{S}_{\mathcal{J}}$ with one peer P_G having the same schema as $\mathcal{G}$ and the constraints of $\mathcal{G}$ as i-constraints, one peer P_i for each source S_i in $\mathcal{S}$, and having $\mathcal{M}$ as the set of i-mappings.

THEOREM 8. *Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a virtual data integration system, and let $\mathcal{S}_{\mathcal{J}}$ be the corresponding PDEI-system. Then, for each database D for $\mathcal{S}$, $Mod(\mathcal{J}, D) = Sem(\mathcal{S}_{\mathcal{J}}, D)$.*

We end this section by comparing *PDEI*-systems with another family of virtual data integration systems, namely peer data management (PDM) systems [29, 22, 12].

The main observation in considering [29, 22] is that, there, mappings between peers are given the classical FOL semantics. Since, as discussed earlier, such a semantics leads to undecidability, in order to recover decidability of query answering, a suitable assumption on the form of mapping (basically, acyclicity) is adopted. On the other hand, here, we do not resort to any assumption on the presence of cycles, but we use a weaker, non-classical interpretation of virtual mappings. It follows that *PDEI*-systems and the PDM systems of the kind proposed in [29, 22] are incomparable.

On the contrary, it is not hard to to see that the semantics proposed here for virtual mappings directly corresponds to the semantics adopted in [12], despite the fact that such a semantics was presented in [12] by resorting to a particular form of epistemic logic. Therefore, the *PDEI*-systems proposed in this paper can be seen as a generalization of the notion of P2P virtual data integration defined in [12]. Obviously, the novel aspect of PDEI-systems is the combination of virtual data integration and P2P data exchange.

5. REASONING IN P2P DATA EXCHANGE AND INTEGRATION

In this section we consider reasoning in *PDEI*-systems. In particular, we first define the EI-CHASE for a given finite state B for a *PDEI*-system $\mathcal{S}$, and prove some important semantic properties of the EI-CHASE. Then, we turn our attention to *stratified PDEI*-systems, a class of *PDEI*-systems where the language for both e-constraints and i-constraints is restricted, and characterize the complexity of state consistency, state admissibility, computing universal solutions, and query answering in stratified *PDEI*-systems.

5.1 The EI-CHASE

To check consistency and admissibility, and to compute universal solutions in *PDEI*-systems, we now extend the chase procedure defined in Section 3. Such an extension is not trivial, since the new chase must carefully take into account the presence of the constraints in $\mathcal{C}_I$ and the P2P mappings in $\mathcal{M}_I$. Our idea is to keep the constraints in $\mathcal{C}_I$ and $\mathcal{M}_I$ as "virtual" in the chase, and use them during the chase procedure only to decide whether a constraint in $\mathcal{C}_E$ and $\mathcal{M}_E$ fires.

In other words, our idea is that the chase has to "materialize" tuples (i.e., add tuples to the chase) based on the constraints in $\mathcal{C}_E$ and $\mathcal{M}_E$, but the firing of such constraints depends not only on the tuples materialized in the chase, but also on the virtual constraints in $\mathcal{C}_I$ and $\mathcal{M}_I$. Definition 11 below formalizes the above idea.

As a preliminary notion, we define $Ans_{Null}(q, \mathcal{S}', B)$, for a CQ q of arity k, a *PDEI*-system $\mathcal{S}'$ of the form $\langle \mathcal{P}, \emptyset, \emptyset, \mathcal{C}_I, \mathcal{M}_I \rangle$, and a finite state B, as follows. If B is not $\mathcal{S}'$-consistent, then $Ans_{Null}(q, \mathcal{S}', B) = \Gamma^k$. For the case where B is $\mathcal{S}'$-consistent, we first define σ as a substitution of the null values in B, i.e., a function $nulls(B) \to \mathcal{C} \cup nulls(B)$, and let $\sigma(B)$ denote the state obtained from B by substituting each null value n with $\sigma(n)$. Then, we define $\sigma_{\mathcal{S}'}$ as any most general substitution of null values in B such that $\sigma_{\mathcal{S}'}(B)$ is $\mathcal{S}'$-consistent when considering the null values as ordinary constants. Finally, $Ans_{Null}(q, \mathcal{S}', B)$ is defined as the set of certain answers to q in $\mathcal{S}'$ and $\sigma_{\mathcal{S}'}(B)$ computed under the assumption that the null values in $\sigma_{\mathcal{S}'}(B)$ are ordinary constants. Roughly speaking, in the above definition the most general substitution $\sigma_{\mathcal{S}'}$ represents the equalities (between pairs of null values in B or a null value and a constant) that are enforced by the key constraints in $\mathcal{S}$.

DEFINITION 11. (EI-CHASE) *Let $\mathcal{S}$ be a PDEI-system, and let B a finite state for $\mathcal{S}$. Let $Ch_{\mathcal{S},B}(0) = B$. For every positive integer i, let $Ch_{\mathcal{S},B}(i+1)$ be the state obtained from $Ch_{\mathcal{S},B}(i)$ by applying the following chase rules (fixing an arbitrary well-founded order of application on the rules and of the null values in $\mathcal{N}$):*

- **TGD-rule:**
  if $q_1(\vec{x}, \vec{y}) \to q_2(\vec{x}, \vec{z}) \in \mathcal{C}_E$
  and $\vec{t} \in Ans_{Null}(q_1, \mathcal{S}^I, Ch_{\mathcal{S},B}(i))$
  and $\vec{t} \notin Ans_{Null}(q_2, \mathcal{S}^I, Ch_{\mathcal{S},B}(i))$
  then $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i) \cup \{q_2(\vec{t}, \vec{n})\}$*, where $\vec{n}$ is the k-tuple of the first k null values that do not appear in* $Ch_{\mathcal{S},B}(i)$*;*
- **EGD-rule:**
  ```
  if $q(x_1, x_2) \to x_1 = x_2 \in \mathcal{C}_E$
      and $\langle t_1, t_2 \rangle \in Ans_{Null}(q, \mathcal{S}^I, Ch_{\mathcal{S},B}(i))$
      and $t_1 \neq t_2$
  then if $t_1 \in \mathcal{N}$
          then $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i)[t_1 \leftarrow t_2]$
          else if $t_2 \in \mathcal{C}$
                  then $Ch_{\mathcal{S},B}(i+1) = \mathsf{FAIL}$
                  else $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i)[t_2 \leftarrow t_1]$;
  ```
- **P2P-mapping-rule:**
  if $q_1(\vec{x}, \vec{y}) \to q_2(\vec{x}, \vec{z}) \in \mathcal{M}_E$
  and $\vec{t} \in Ans(q_1, \mathcal{S}^I, Ch_{\mathcal{S},B}(i))$
  and $\vec{t} \notin Ans(q_2, \mathcal{S}^I, Ch_{\mathcal{S},B}(i))$
  then $Ch_{\mathcal{S},B}(i+1) := Ch_{\mathcal{S},B}(i) \cup \{q_2(\vec{t}, \vec{n})\}$*, where $\vec{n}$ is the k-tuple of the first k null values that do not appear in* $Ch_{\mathcal{S},B}(i)$.

Finally, let $CH = \bigcup_{i \in \mathbb{N}} Ch_{\mathcal{S},B}(i)$. We define EI-CHASE$(\mathcal{S}, B)$ *as follows:* EI-CHASE$(\mathcal{S}, B) = \mathsf{FAIL}$ *if either CH is not $\mathcal{S}^I$-consistent or there exists i such that $Ch_{\mathcal{S},B}(i) = \mathsf{FAIL}$, otherwise* EI-CHASE$(\mathcal{S}, B) = CH$.

It turns out that EI-CHASE is the right tool for characterizing the $\mathcal{S}$-consistency and the $\mathcal{S}$-admissibility of a finite state. Formally:

THEOREM 9. *Let $\mathcal{S}$ be a PDEI-system and B a finite state for $\mathcal{S}$. Then,*

- *B is $\mathcal{S}$-consistent iff* EI-CHASE$(\mathcal{S}, B) \neq \mathsf{FAIL}$*;*
- *if* EI-CHASE$(\mathcal{S}, B) \neq \mathsf{FAIL}$*, then* EI-CHASE$(\mathcal{S}, B)$ *is a universal $\mathcal{S}$-solution of B;*
- *B is $\mathcal{S}$-admissible iff* EI-CHASE$(\mathcal{S}, B) = B$.

5.2 Reasoning in stratified *PDEI*-systems

So far, we have not imposed any limitation on the form of the i-constraints and e-constraints occurring in *PDEI*-systems, and we do not have looked in depth at the interaction between these two kinds of constraints. Analogously to the case of *PDE*-systems, we are again interested in enforcing on *PDEI*-systems the property that, given a finite state B, there exists a finite universal $\mathcal{S}$-solution for B. A direct consequence of this requirement is that we must limit the expressive power of the language used to specify both e-constraints and i-constraints. As for e-constraints we again resort to the assumption of weak-acyclicity. Actually the same assumption could be adopted for i-constraints too, and the technical development would be a relative simple extension of the one presented in Section 3. However, in this paper we take another approach, which turns out to be much more challenging. Indeed, we consider a specific class of i-constraints that roughly correspond to key and foreign key constraints, and constitute a maximally expressive class of constraints considered in virtual data integration [30]. Thus, our choice of e-constraints and i-constraints is justified by the goal of studying *PDEI*-systems with the most expressive class of constraints considered in the two areas.

The i-constraints considered in this paper are of two types:

- A *key constraint* is a special form of EGD, stating that a set of components forms a key for a given relation. Formally, for a relation r of arity n, a key constraint has the form $key(r) = \{i_1, \ldots, i_k\}$, where for each i_h, $1 \leq i_h \leq n$. The set $\{i_1, \ldots, i_k\}$ is called the key of r.
- A *foreign key dependency* is a special form of inclusion dependency (and, therefore, of TGD), stating that a certain combination of components of a relation s (of arity m) references a *subset* of the key of a relation r (of arity n). Formally, a foreign key dependency has the form $s[j_1, \ldots, j_p] \subseteq r[i_1, \ldots, i_p]$, where for each j_h, $1 \leq j_h \leq m$, for each i_h, $1 \leq i_h \leq n$, and $\{i_1, \ldots, i_p\}$ is a (not necessarily proper) subset of the key of r. In what follows, r will be called the *head* of the foreign key dependency.

A set of key constraints and foreign key dependencies is called *legal* if it contains at most one key constraint for each relation symbol. Note that no acyclicity assumption of any type is imposed on foreign key dependencies, and therefore they are not subsumed by weakly-acyclic TDGs. Nevertheless, as we said before, key constraints and foreign key dependencies form one of the maximally expressive classes of constraints considered in the virtual data integration literature, and for which computing certain answers is decidable [9, 30]. Interestingly, several information systems that might be modeled as peers in our approach export their information in terms of formalisms, such as Entity-Relationship model, or UML-style languages, and such formalisms can be expressed in terms of key constraints and foreign key dependencies, whereas they are not captured by weakly-acyclic TGDs. An example is shown in Figure 1. Since every instance of E is linked to an instance of F by means of R, and every instance of F has a Q-link to at least one instance of E, it is immediate to see that the TGDs corresponding to the schema in Figure 1 are not weakly-acyclic. On the other hand, it is not hard to see that the semantics of the above schema is captured by a set of key constraints and foreign key dependencies.

Unfortunately, simply combining weakly-acyclic e-constraints with legal key constraints and foreign key dependencies is not yet

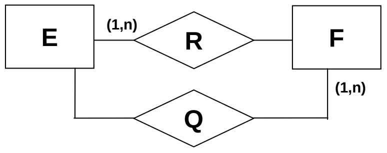

Figure 1: Cyclic inclusion dependencies

sufficient to ensure nice properties of *PDEI*-systems, as shown in the following theorem.

THEOREM 10. *There exists a PDEI-system $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$, where $\mathcal{C}_E$ is a set of weakly-acyclic TGDs, and $\mathcal{C}_I$ is a set of legal key constraints and foreign key dependencies, and a finite $\mathcal{S}$-consistent state B such that every universal $\mathcal{S}$-solution of B is infinite.*

Proof (sketch). Consider a system $\mathcal{S}$ constituted by a single peer without mappings and with the following constraints: $\mathcal{C}_E = \{\{x \mid E(x)\} \rightarrow \{x \mid R(x,y)\}\}$, and $\mathcal{C}_I = \{R[2] \subseteq F[1], F[1] \subseteq Q[2], Q[1] \subseteq E[1]\}$ (cf. Figure 1). Then, it is easy to verify that the $\mathcal{S}$-consistent state $B = \{E(a)\}$ admits only infinite universal $\mathcal{S}$-solutions. □

The above theorem tells us that, for our purpose, we need to impose a suitable limitation on the interaction between e-constraints and i-constraints in *PDEI*-systems. To meet this requirement, we introduce the notion of stratified *PDEI*-systems.

DEFINITION 12. *A PDEI-system $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ is said to be* stratified *if $\mathcal{C}_E$ is a set of weakly-acyclic TGDs, $\mathcal{C}_I$ is a set of legal key constraints and foreign key dependencies, and no head of a foreign key dependency in $\mathcal{C}_I$ appears in the left-hand side of any TGD in $\mathcal{C}_E$.*

EXAMPLE 2. *Consider now the PDEI-system $\mathcal{S} = \langle \{P_1, P_2, P_3\}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ where $\mathcal{C}_E$, and the signatures of P_1 and P_2 are as in Example 1, the signature of P_3 comprises the binary relation symbols R_4, R_5, and R_6, $\mathcal{M}_E$ comprises the same dependencies of Example 1, plus the dependency $\{x \mid R_3(x,y)\} \rightarrow \{x \mid R_5(z,x)\}$, $\mathcal{C}_I$ comprises the following set of dependencies*

$$\begin{array}{ll} key(R_4) = \{1\} & R_6[1] \subseteq R_5[1] \\ key(R_5) = \{1\} & R_5[2] \subseteq R_4[1] \\ key(R_6) = \{2\} & R_4[2] \subseteq R_6[2] \end{array}$$

and $\mathcal{M}_I$ consists of the only dependency

$$\{x \mid R_4(x,y)\} \quad \rightarrow \quad \{x \mid R_1(z,x)\}.$$

It is easy to see the the system above is stratified. Assume now that the state B for $\mathcal{S}$ is as in Example 1. Then, both $B' = \{R_1(a,b),\ R_1(x_1,d),\ R_2(a,d),\ R_3(d,x_2),\ R_3(f,b),\ R_3(h,d),\ R_5(x_3,b),\ R_5(x_4,d)\}$ and $B'' = B' - \{R_1(x_1,d)\}$ are universal $\mathcal{S}$-solutions. Indeed, it is easy to see that (i) for each $D \in Sem(\mathcal{S}, B)$ there exists a homomorphism from B'' to D, (ii) B'' is $\mathcal{S}$-consistent, and (iii) B' is a universal $\mathcal{S}^{I \rightarrow E}$-solution of B'' that is $\mathcal{S}^E$-admissible (cf. Definition 10). Both universal solutions can be obtained by applying our procedure for computing the EI-CHASE *starting to the initial state B by using a different order in the application of the rules.* □

We now show that, under the above restriction, every $\mathcal{S}$-consistent state admits a finite universal $\mathcal{S}$-solution.

THEOREM 11. *Let $\mathcal{S}$ be a stratified PDEI-system and B a finite state $\mathcal{S}$-consistent state. Then, there exists a finite universal $\mathcal{S}$-solution of B.*

We point out that the existence of a finite universal $\mathcal{S}$-solution of B does not imply that the chase of all (i.e., both exchange and virtual) constraints and mappings of $\mathcal{S}$ terminates. Conversely, it can be shown that there exists no finite state B' such that computing the certain answers to a union of conjunctive queries w.r.t. $\mathcal{S}$ and B reduces to evaluating the query over B'.

Nevertheless, in the rest of the section, we will show that stratification is sufficient to allow for developing techniques for data management tasks in *PDEI*-systems.

We start with the next theorem that characterizes the complexity of various reasoning tasks for a stratified *PDEI*-system $\mathcal{S}$. Its proof is based on the following facts: (*i*) the procedure EI-CHASE is analogous to E-CHASE, with the only difference that the conditions for firing rules are based on computing $Ans_{Null}(q, \mathcal{S}^I, Ch_{\mathcal{S},B}(i))$, i.e., computing certain answers on a system with virtual mappings and constraints, instead of evaluating queries over indefinite instances through $Eval_{Null}(q, Ch_{\mathcal{S},B}(i))$; (*ii*) $Ans_{Null}(q, \mathcal{S}^I, Ch_{\mathcal{S},B}(i))$ can be computed in polynomial time w.r.t. data complexity; (*iii*) checking $\mathcal{S}^I$ consistency of a finite state can be done in polynomial time w.r.t. data complexity.

THEOREM 12. *Let $\mathcal{S}$ be a stratified PDEI-system and B a finite state for $\mathcal{S}$. Then,*

- *deciding whether B is $\mathcal{S}$-consistent can be done in polynomial time w.r.t. data complexity;*
- *deciding whether B is $\mathcal{S}$-admissible can be done in polynomial time w.r.t. data complexity;*
- *if B is $\mathcal{S}$-consistent, then a universal $\mathcal{S}$-solution of B can be computed in polynomial time w.r.t. data complexity.*

Query answering. Unfortunately, we cannot directly use EI-CHASE to do query answering: although the state $B' =$ EI-CHASE$(\mathcal{S}, B)$ is a universal $\mathcal{S}$-solution of the initial state B, we already observed that this property is not sufficient to reduce query answering to query evaluation over B'. Indeed, Theorem 7 implies that, although we can ignore e-mappings and e-constraints to compute the certain answers to a query in B', we still have to take into account both i-mappings and i-constraints.

The way in which we take into account i-mappings and i-constraints is based on the notion *perfect reformulation* of a UCQ under inclusion dependencies [10].

Rephrased in the present setting, the notion of perfect reformulation of a query can be expressed as follows: given a set of local constraints $\mathcal{C}_I$ over a peer P, and a query q, let $\mathcal{S} = \langle \{P\}, \emptyset, \emptyset, \mathcal{C}_I, \emptyset \rangle$. Then the perfect reformulation of q w.r.t. $\mathcal{C}_I$ is a query q' such that, for each $\mathcal{S}$-admissible state B, $Ans(q, \mathcal{S}, B) = Eval_{Null\downarrow}(q', B)$.

In [10] it has been shown that, if $\mathcal{C}_I$ is a set of arbitrary inclusion dependencies, then the perfect reformulation of a UCQ q is a UCQ. Moreover, an algorithm has been defined for computing the perfect reformulation. We call COMPUTEPERFECTREF such algorithm, which takes as input a UCQ q and a set of inclusion dependencies $\mathcal{I}$ and returns a UCQ (a perfect reformulation of q) as output. Moreover, a consequence of [10] is that the above perfect reformulation is still correct for consistent states when $\mathcal{C}_I$ is a set of legal key constraints and foreign key dependencies. More precisely, if $\mathcal{C}_I$ is a set of legal key constraints and foreign key dependencies, and $\mathcal{I}$ is the set of foreign key dependencies in $\mathcal{C}_I$, then the query q' returned by COMPUTEPERFECTREF$(q, \mathcal{I})$ is such that for each $\mathcal{S}$-admissible state B, $Ans(q, \mathcal{S}, B) = Eval_{Null\downarrow}(q', B)$.

Let us now turn our attention to *PDEI*-systems. To solve the query answering problem, we resort to an encoding of stratified

PDEI-systems into *PDE*-systems which actually add tuples according to the P2P mappings in $\mathcal{M}_I$, and also considers a "composition" of the inclusion dependencies in $\mathcal{C}_I$ with the P2P mappings in $\mathcal{M}_I$. Informally, through this encoding we materialize a new state B' which is a finite portion of a universal $\mathcal{S}$-solution of B. The new state B' has the crucial property of being an "almost-correct" representative of $Sem(\mathcal{S}, B)$ with respect to UCQs, in the sense that *the certain answers to a UCQ q in B correspond to the evaluation in B' of the perfect reformulation of q w.r.t. the inclusion dependencies in $\mathcal{C}_I$*.

Formally: let $\mathcal{I}$ be a set of inclusion dependencies and let ϕ be the TGD $q_1 \rightarrow q_2$. We define

$$Expand(\phi, \mathcal{I}) = \{q' \rightarrow q_2 \mid q' \in \textsc{ComputePerfectRef}(q_1, \mathcal{I})\}$$

Moreover, given a set $\mathcal{T}$ of TGDs, we define

$$Expand(\mathcal{T}, \mathcal{I}) = \bigcup_{\phi \in \mathcal{T}} Expand(\phi, \mathcal{I})$$

Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ be a *PDEI*-system and let $\mathcal{I}$ be the set of inclusion dependencies in $\mathcal{C}_I$. We define $\tau(\mathcal{S})$ as the *PDE*-system $\tau(\mathcal{S}) = \langle \mathcal{P}, \emptyset, Expand(\mathcal{M}_I, \mathcal{I}), \emptyset, \emptyset \rangle$.

THEOREM 13. *Let $\mathcal{S} = \langle \mathcal{P}, \mathcal{C}_E, \mathcal{M}_E, \mathcal{C}_I, \mathcal{M}_I \rangle$ be a stratified PDEI-system, $\mathcal{I}$ the set of foreign key dependencies in $\mathcal{C}_I$, B a finite $\mathcal{S}$-admissible state, and q a query over $\mathcal{S}$. Then, $Ans(q, \mathcal{S}, B) = Eval_{Null\downarrow}(\textsc{ComputePerfectRef}(q, \mathcal{I}), \text{E-}\textsc{Chase}(\tau(\mathcal{S}), B))$.*

Theorems 12 and 13, and the known complexity results for ComputePerfectRef imply the following property.

THEOREM 14. *Let $\mathcal{S}$ be a stratified PDEI-system and B a finite $\mathcal{S}$-admissible state. Then, query answering can be solved in polynomial time w.r.t. data complexity.*

On $\mathcal{S}$-cores. We now turn our attention to the notion of $\mathcal{S}$-core in *PDEI*-systems, and show that, differently from the case of *PDE*-systems (see Proposition 2), a state of a *PDEI*-system $\mathcal{S}$ may admit several $\mathcal{S}$-cores.

EXAMPLE 3. *Consider the following PDEI-system $\mathcal{S}$ formed by three peers P_1, P_2 and P_3, without e-constraints and i-constraints, and with the following P2P mappings:*

$$\begin{aligned} \mathcal{M}_e &= \{\{x \mid R_1(x)\} \rightarrow \{x \mid R_2(x)\}\} \\ \mathcal{M}_i &= \{\{x \mid R_2(x)\} \rightarrow \{x \mid R_3(x)\}, \\ &\qquad \{x \mid R_3(x)\} \rightarrow \{x \mid R_2(x)\}\} \end{aligned}$$

Then the $\mathcal{S}$-consistent state $B = \{R_1(a)\}$ admits two $\mathcal{S}$-cores: $B_c = \{R_1(a), R_2(a)\}$ and $B'_c = \{R_1(a), R_3(a)\}$.

The example shows that the crucial property of the notion of core, i.e., uniqueness, is lost in arbitrary *PDEI*-systems.

6. CONCLUSIONS

Several issues are left out by our investigation. We mention three of them. First, it is open whether there are notable cases of *PDEI*-systems in which an $\mathcal{S}$-core can be computed in polynomial time. Second, we have not addressed the issue of mapping composition, which becomes particularly intriguing when we mix data exchange and data integration mappings. Finally, it would be interesting to look for more expressive combinations of data exchange and data integration mappings and constraints that still retain nice computational properties.

Acknowledgments. This research has been partially supported by FET project TONES (Thinking ONtologiES), funded by the EU under contract number FP6-7603, by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant, and by MIUR FIRB 2005 project "Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet" (TOCAI.IT).

7. REFERENCES

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS'98*, pages 254–265, 1998.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.

[3] 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.

[4] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The Hyperion project: from data integration to data coordination. *SIGMOD Record*, 32(3):53–58, 2003.

[5] M. Arenas and L. Libkin. XML data exchange: consistency and query answering. In *Proc. of PODS 2005*, pages 13–24, 2005.

[6] 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.

[7] L. Bravo and L. Bertossi. Logic programming for consistently querying data integration systems. In *Proc. of IJCAI 2003*, pages 10–15, 2003.

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

[9] 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.

[10] 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.

[11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Inconsistency tolerance in P2P data integration: an epistemic logic approach. In *Proc. of DBPL 2005*, pages 90–105, 2005.

[12] 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.

[13] P. Chatalic, G.-H. Nguyen, and M.-C. Rousset. Reasoning with inconsistencies in propositional Peer-to-Peer inference systems. In *Proc. of ECAI 2006*, pages 352–357, 2006.

[14] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theor. Comp. Sci.*, 336(1):89–124, 2005.

[15] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *ACM Trans. on Database Systems*, 30(1):174–210, 2005.

[16] R. Fagin, P. G. Kolaitis, L. Popa, and W.-C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. on Database Systems*, 30(4):994–1055, 2005.

[17] 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 (DBISP2P 2003)*, 2003.

[18] A. Fuxman, P. G. Kolaitis, R. Miller, and W. C. Tan. Peer data exchange. In *Proc. of PODS 2005*, pages 160–171, 2005.

[19] G. Gottlob. Computing cores for data exchange: New algorithms and practical solutions. In *Proc. of PODS 2005*, pages 148–159, 2005.

[20] G. Gottlob and A. Nash. Data exchange: computing cores in polynomial time. In *Proc. of PODS 2006*, pages 40–49, 2006.

[21] 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.

[22] 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.

[23] A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[24] A. Y. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proc. of VLDB 2006*, pages 9–16, 2006.

[25] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. of PODS 2005*, pages 61–75, 2005.

[26] P. G. Kolaitis, J. Panttaja, and W. C. Tan. The complexity of data exchange. In *Proc. of PODS 2006*, pages 30–39, 2006.

[27] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.

[28] L. Libkin. Data exchange and incomplete information. In *Proc. of PODS 2006*, pages 60–69, 2006.

[29] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *Proc. of VLDB 2003*, pages 572–583, 2003.

[30] R. Rosati. On the decidability and finite controllability of query processing in databases with incomplete information. In *Proc. of PODS 2006*, pages 356–365, 2006.

[31] I. Tatarinov and A. Halevy. Efficient query reformulation in peer data management. In *Proc. of ACM SIGMOD*, 2004.