Peer Data Exchange

ARIEL FUXMAN
University of Toronto
PHOKION G. KOLAITIS
IBM Almaden Research Center
RENÉE J. MILLER
University of Toronto
and
WANG-CHIEW TAN
University of California, Santa Cruz

In this article, we introduce and study a framework, called *peer data exchange*, for sharing and exchanging data between peers. This framework is a special case of a full-fledged peer data management system and a generalization of data exchange between a source schema and a target schema. The motivation behind peer data exchange is to model authority relationships between peers, where a source peer may contribute data to a target peer, specified using source-to-target constraints, and a target peer may use target-to-source constraints to restrict the data it is willing to receive, but cannot modify the data of the source peer.

A fundamental algorithmic problem in this framework is that of deciding the existence of a solution: given a source instance and a target instance for a fixed peer data exchange setting, can the target instance be augmented in such a way that the source instance and the augmented target instance satisfy all constraints of the setting? We investigate the computational complexity of the problem for peer data exchange settings in which the constraints are given by tuple generating dependencies. We show that this problem is always in NP, and that it can be NP-complete even for "acyclic" peer data exchange settings. We also show that the data complexity of the certain answers of target conjunctive queries is in coNP, and that it can be coNP-complete even for "acyclic" peer data exchange settings.

R. J. Miller and A. Fuxman were partially supported by grants from NSERC. W.-C. Tan is supported in part by NSF CAREER Award IIS-0347065 and NSF grant IIS-0430994. P. G. Kolaitis is on leave from the University of California, Santa Cruz.
A prelimnary version of this article appeared in *Proceedings of the 2005 ACM Symposium on the Principles of Database Systems* (Baltimore, MD). 160–171.
Authors' addresses: A. Fuxman and R. J. Miller, Department of Computer Science, University of Toronto, Toronto, Ont., M5S 2E4, Canada; email: {afuxman,miller}@cs.toronto.edu; P. G. Kolaitis, IBM Almaden Research Center, Dept. K53, 650 Harry Road, San Jose, CA 95120; email: kolaitis@almaden.ibm.com; W.-C. Tan, Department of Computer Science, MS: SOE3, University of California, Santa Cruz, 1156 High Street, Santa Cruz, CA 95064; email: wctan@cs.ucsc.edu.

After this, we explore the boundary between tractability and intractability for deciding the existence of a solution and for computing the certain answers of target conjunctive queries. To this effect, we identify broad syntactic conditions on the constraints between the peers under which the existence-of-solutions problem is solvable in polynomial time. We also identify syntactic conditions between peer data exchange settings and target conjunctive queries that yield polynomial-time algorithms for computing the certain answers. For both problems, these syntactic conditions turn out to be tight, in the sense that minimal relaxations of them lead to intractability. Finally, we introduce the concept of a universal basis of solutions in peer data exchange and explore its properties.

Categories and Subject Descriptors: H.2.5 [**Database Management**]: Heterogeneous Databases—*Data translation*; F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Data exchange, data integration, schema mapping, certain answers, conjunctive queries, metadata model management

1. INTRODUCTION

Several different frameworks for sharing data between independent stores have been formulated and investigated in depth. *Data exchange* is one of the conceptually simpler, yet technically challenging, such frameworks Fagin et al. [2005a]. In a data exchange setting, data from a source schema is transformed to data over a target schema according to specifications given by source-to-target constraints. This framework models a situation in which the target passively receives data from the source, as long as the source-to-target constraints are satisfied. Data exchange is closely related to data integration [Lenzerini 2002]. In particular, data exchange systems can be used as building blocks in data integration systems, where data from a set of independent sources having no interaction with each other is transformed to data in a global mediated schema. *Peer data management systems* (PDMS) constitute a much more powerful and complex framework than data exchange, as they model a situation in which a number of peers interact with each other and cooperate in sharing and exchanging data [Halevy et al. 2005; Tatarinov and Halevy 2004; Tatarinov 2004]. In a peer data management system, there is no distinction between source and target, since a peer may simultaneously act as a distributor of data (and, thus, a source peer) and a recipient of data (and, thus, a target peer). In such a system, the relationship between peers is specified using constraints that can be in either direction (from one peer to another, and vice versa), instead of constraints in a single direction, as was the case in data exchange. Furthermore, each peer can be a stand-alone database system or a separate data integration system in which the schema of the peer is a mediated global schema over a set of local sources accessible only by that peer.

1.1 The Peer Data Exchange Framework

In this article, we introduce and study a framework, called *peer data exchange*, which is a generalization of data exchange and a special case of a full-fledged peer data management system. This framework models a situation in which there is interaction between two peers that have different roles and capabilities: one of them, called the *source* peer, is an “authoritative” or “trusted” peer that can contribute new data, while the other peer, called the *target* peer, imposes restrictions on the data that it is willing to accept, but has no permission or capability to modify the data of the source peer. In a peer data exchange setting, the relationship between the two peers is specified by constraints that go in either direction, that is, some are source-to-target constraints and others are target-to-source constraints; in addition, target constraints may be present. As in data exchange, the source-to-target constraints specify what data a source peer is willing to exchange. Unlike data exchange, however, the target is no longer a passive recipient of source data that obeys the source-to-target constraints. Instead, the target peer uses target-to-source constraints to impose restrictions on the data that it is willing to receive; moreover, the target may have its own data. Thus the role of source-to-target and target constraints is quite different from the role of target-to-source constraints. Specifically, source-to-target and target constraints are used to generate new tuples in the target relations, while target-to-source constraints eliminate extensions of the target relations that violate the target-to-source constraints. Suppose that we are given a source instance and a target instance that may or may not satisfy the constraints of the setting; if the constraints are not satisfied, the goal then is to augment the target data in such a way that the *given* source instance and the *augmented* target instance satisfy all constraints between the two peers, as well as other existing target constraints. As an illustration, the source peer may be an authoritative genomic database, such as Swiss-Prot [O'Donovan et al. 2002], while the target peer may be a genomic database maintained at a university under a different schema and populated with various data. At regular intervals of time, the university database is willing to receive new data from Swiss-Prot but cannot export any data back to Swiss-Prot. The target may restrict the data it is willing to receive to only Swiss-Prot data that it views as relevant. Hence, the data received has to satisfy constraints that go in either direction.

1.2 Algorithmic Problems

The first fundamental algorithmic problem in peer data exchange is that of deciding the *existence of a solution*. More formally, a peer data exchange setting consists of a source schema **S**, a target schema **T**, a set of source-to-target constraints Σ_{st}, a set of target-to-source constraints Σ_{ts}, and a set Σ_t of target constraints. Each such setting gives rise to the following decision problem: given a source instance and a target instance, can the target instance be augmented in such a way that the given source instance and the augmented target instance satisfy all constraints of the peer exchange setting? The second fundamental algorithmic problem in peer data exchange is that of obtaining the *certain answers* of queries posed over the target schema. The concept of the certain answers has become the standard semantics of query answering in data integration [Abiteboul and Duschka 1998; Lenzerini 2002], data exchange [Fagin et al. 2005a], and peer data management [Halevy et al. 2005]; this concept is also perfectly meaningful in peer data exchange.

In the sequel, we investigate these algorithmic problems for peer data exchange settings in which the constraints between the peers are given by a finite set of tuple-generating dependencies (tgds) [Beeri and Vardi 1984]. We also allow for target constraints in the form of target tgds or target equality-generating dependencies (target egds). By definition, a tgd from one relational schema to another is a first-order formula of the form $\forall\mathbf{x}(\varphi(\mathbf{x}) \rightarrow \exists\mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$, where $\varphi(\mathbf{x})$ is a conjunction of atomic formulas over the first schema and $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulas over the second. An equality-generating dependency on a relational schema is a formula of the form $\forall\mathbf{x}(\varphi(\mathbf{x}) \rightarrow z_1 = z_2)$, where $\varphi(\mathbf{x})$ is a conjunction of atomic formulas over the schema and z_1, z_2 are among the variables in $\mathbf{x}$. Tuple-generating dependencies have been used for specifying data exchange between relational schemas [Fagin et al. 2005a, 2005b]; moreover, they are the core of the mapping specification language of the Clio schema-mapping and data exchange system [Popa et al. 2002]. Tuple-generating dependencies generalize both the local-as-view (LAV) and the global-as-view (GAV) constraints in data integration [Lenzerini 2002], since the former are tgds in which $\varphi(\mathbf{x})$ is a single atomic formula, and the latter are tgds in which $\psi(\mathbf{x}, \mathbf{y})$ is a single atomic formula. In their full generality, tuple-generating dependencies are GLAV (*global-and-local-as-view*) constraints.

1.3 Summary of Results

Consider a fixed peer data exchange setting in which Σ_{st} is a finite set of source-to-target tgds, Σ_{ts} is a finite set of target-to-source tgds, and $\Sigma_t = \emptyset$ (no target constraints). Our first main result asserts that testing for the existence of solutions is in NP, and that the data complexity of the certain answers of unions of conjunctive queries is in coNP. These complexity bounds turn out to be tight, because we exhibit peer data exchange settings as above for which testing for the existence of solutions is NP-complete, while the data complexity of the certain answers of conjunctive queries is coNP-complete; actually, the lower bounds hold even for peer data exchange settings in which the "dependency" graph between the relations of the peers is acyclic. We also show that the same upper bounds hold even if the setting allows for a set Σ_t of target constraints that is the union of a finite set of target egds and a finite *weakly acyclic* set of target tgds.

The complexity of testing for the existence of solutions and computing the certain answers in peer data exchange settings should be compared and contrasted with the complexity of the same problems for data exchange, which can be viewed as the special case of peer data exchange in which $\Sigma_{ts} = \emptyset$ (no target-to-source tgds) and also $J = \emptyset$ (the target contains no data before the exchange). Indeed, as shown in Fagin et al. [2005a] there are polynomial-time algorithms to test for the existence of solutions and to compute the certain answers of unions of conjunctive queries in every data exchange setting in which Σ_{st} is a finite set of source-to-target tgds and Σ_t is the union of a finite set of target egds and a finite weakly acyclic set of target tgds. Moreover, if $\Sigma_t = \emptyset$ (no target constraints), then testing for the existence of solutions is trivial for data exchange, since solutions always exist. There is also a sharp contrast with full-fledged peer data management systems, where, as shown in Halevy et al. [2005], computing the certain answers of conjunctive queries can be an undecidable problem. Thus, from a computational point of view, peer data exchange is more challenging than ordinary data exchange, but less intractable than full peer data management.

After this, we explore the boundary between tractability and intractability in peer data exchange settings with no target constraints. We identify a class of peer data exchange settings, denoted by $\mathcal{C}_{tract}$, for which the existence-of-solutions problem is solvable in polynomial time. The class $\mathcal{C}_{tract}$ is defined by imposing syntactic conditions on the constraints between the peers; these conditions are extracted through a careful examination of the impact of existentially quantified variables and of their relationship to other variables occurring in the constraints. Even though the definition of $\mathcal{C}_{tract}$ is quite technical, $\mathcal{C}_{tract}$ itself is a broad class that contains several important special cases of peer data exchange, including the case in which the source-to-target tgds are *full* tgds and the case in which the target-to-source tgds are *local-as-view* (LAV) constraints. Moreover, minimal relaxations of the syntactic conditions defining $\mathcal{C}_{tract}$ are satisfied by peer data exchange settings for which the existence-of-solutions problem is NP-complete. Thus, $\mathcal{C}_{tract}$ turns out to be a maximal class of peer data exchange settings with a tractable existence-of-solutions problem. As regards the data complexity of query answering, we show that there are peer data exchange settings in $\mathcal{C}_{tract}$ and conjunctive queries such that computing the certain answers is a coNP-complete problem. Nonetheless, for every peer data exchange setting in $\mathcal{C}_{tract}$, we identify a large class of conjunctive queries whose certain answers can be computed in polynomial time. In particular, for peer data exchange settings with no target constraints and such that all source-to-target tgds are full, the certain answers of every conjunctive query can be computed in polynomial time.

Finally, to gain a deeper insight into the differences between data exchange and peer data exchange, we introduce the concept of a *universal basis* of solutions in peer data exchange. We compare this concept to the concept of a *universal* solution in data exchange introduced in [Fagin et al. 2005a], and study some key properties of universal bases in peer data exchange.

1.4 Related Work

There is an extensive literature on data integration using sound, complete and exact views [Abiteboul and Duschka 1998; Grahne and Mendelzon 1999; Lenzerini 2002]. Several different frameworks and systems for sharing data in networks of independent sources have also been formulated and studied [Bernstein et al. 2002; Calvanese et al. 2004b; Franconi et al. 2003, 2004; Li 2004]. In particular, a framework with semantics based on epistemic logic has been investigated in Calvanese et al. [2004a, 2004b, 2005]. This is in contrast to the first-order interpretation used in PDMS and in the work reported here. An advantage of the epistemic-semantics framework is that it enjoys good computational properties; in particular, the certain answers of fixed conjunctive queries posed against a peer can be computed in polynomial time. Finally, Bertossi and Bravo [2004] also used first-order interpretations, but proposed a semantics drawn from the area of consistent query answering that was based on repairs [Arenas et al. 1999]. This approach has the advantage that data can be shared between peers, even when there is no consistent solution satisfying all constraints. However, the complexity of the problem of obtaining certain answers is higher than in peer data exchange (Π_2^p-complete vs. coNP-complete), and no tractability results have been given for this semantics.

2. PEER DATA EXCHANGE SETTINGS

This section contains the precise definitions of a peer data exchange setting and the associated algorithmic problems, as well as a brief discussion of the relationship of peer data exchange settings with data exchange settings and peer data management systems.

2.1 Preliminaries

A *schema* is a finite collection $\mathbf{R} = (R_1, \ldots, R_k)$ of relation symbols, each of a fixed arity. An *instance* I over $\mathbf{R}$ is a sequence $(R_1^I, \ldots, R_k^I)$ such that each R_i^I is a finite relation of the same arity as R_i. We shall often use R_i to denote both the relation symbol and the relation R_i^I that interprets it. Given a tuple $\mathbf{t}$, we denote by $R(\mathbf{t})$ the association between $\mathbf{t}$ and the relation R where it occurs. Let $\mathbf{S} = (S_1, \ldots, S_n)$ and $\mathbf{T} = (T_1, \ldots, T_m)$ be two disjoint schemas. We refer to $\mathbf{S}$ as the *source* schema and to $\mathbf{T}$ as the *target* schema. We write $(\mathbf{S}, \mathbf{T})$ to denote the schema $(S_1, \ldots, S_n, T_1, \ldots, T_m)$. Instances over $\mathbf{S}$ will be called *source* instances, while instances over $\mathbf{T}$ will be called *target* instances. If I is an instance over $\mathbf{S}$ and J is an instance over $\mathbf{T}$, then we write (I, J) to denote the instance K over $(\mathbf{S}, \mathbf{T})$ such that $S_i^K = S_i^I$ and $T_j^K = T_j^J$, for $1 \le i \le n$ and $1 \le j \le m$.

As usual, the *active domain* of an instance consists of all elements occurring in a tuple in one of the relations of the instance. We assume that the elements of the active domain come from two disjoint sets, the set of *constants* and the set of *labeled nulls*. Typically, constants will be denoted by $a, b, c, \ldots$, while labeled nulls will be denoted by $n_1, n_2, \ldots, p_1, p_2, \ldots$.

A *source-to-target tuple-generating dependency* (tgd) is a formula of the form $\forall\mathbf{x}(\phi_s(\mathbf{x}) \to \exists\mathbf{y}\psi_t(\mathbf{x}, \mathbf{y}))$, where $\phi_s(\mathbf{x})$ is a conjunction of atomic formulas over the source schema $\mathbf{S}$, and $\psi_t(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulas over the target schema $\mathbf{T}$. Similarly, a *target-to-source tgd* is a formula of the form $\forall\mathbf{x}(\alpha_t(\mathbf{x}) \to \exists\mathbf{y}\beta_s(\mathbf{x}, \mathbf{y}))$, where $\alpha_t(\mathbf{x})$ is a conjunction of atomic formulas over the target schema $\mathbf{T}$, and $\beta_s(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulas over the source schema $\mathbf{S}$. For example, if $\mathbf{S}$ contains a binary relation E, and $\mathbf{T}$ contains a binary relation H, then the source-to-target tgd $\forall x \forall y \forall z(E(x, z) \wedge E(z, y) \to H(x, y))$ transforms pairs of nodes connected via an E-path of length 2 to H-edges. Similarly, $\forall x \forall y(H(x, y) \to \exists z(E(x, z) \wedge E(z, y)))$ is a target-to-source tgd that transforms H-edges to pairs of nodes connected via an E-path of length 2.

A *target tgd* is a formula of the form $\forall\mathbf{x}(\phi(\mathbf{x}) \to \exists\mathbf{y}\psi(\mathbf{x}, \mathbf{y}))$, where both $\phi(\mathbf{x})$ and $\psi(\mathbf{x}, \mathbf{y})$ are conjunctions of atomic formulas over the target schema $\mathbf{T}$. A *target equality-generating dependency* (egd) is a formula of the form $\forall\mathbf{x}(\varphi(\mathbf{x}) \to z_1 = z_2)$, where $\varphi(\mathbf{x})$ is a conjunction of atomic formulas over $\mathbf{T}$ and z_1, z_2 are among the variables in $\mathbf{x}$. Clearly, functional dependencies on $\mathbf{T}$ are special cases of target egds. In what follows, we will often drop the universal quantifiers in

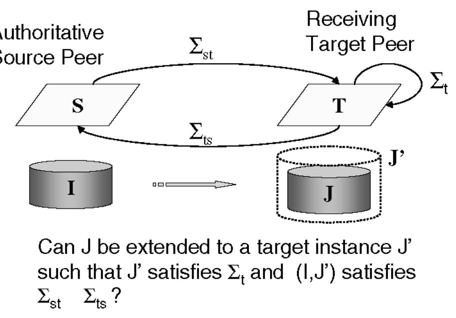
Authoritative Source Peer
Receiving Target Peer
Σ_{st}
Σ_t
Σ_{ts}
S
T
I
J
J'
Can J be extended to a target instance J' such that J' satisfies Σ_t and (I,J') satisfies Σ_{st} Σ_{ts}?

Fig. 1. Illustration of peer data exchange.

front of a dependency, and implicitly assume such quantification. However, we will write down all existential quantifiers.

2.2 Peer Data Exchange Settings and Solutions

Definition 2.1. A *peer data exchange (PDE) setting* is a quintuple $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ such that (see Figure 1)

—$\mathbf{S}$ is a source schema and $\mathbf{T}$ is a target schema;
—Σ_{st} is a finite set of source-to-target tgds;
—Σ_{ts} is a finite set of target-to-source tgds;
—Σ_t is a finite set of target tgds and target egds.

Given a source instance I and a target instance J of $\mathcal{P}$ such as all elements in the active domain of (I, J) are constants, it may be the case that (I, J) violates the constraints of $\mathcal{P}$. Thus we will be interested in finding instances (whose active domain may also contain labeled nulls), which we call *solutions*, that satisfy all constraints of $\mathcal{P}$. In peer data exchange the target peer is assumed to be willing to accept data from an authoritative, trusted source. Therefore, we will consider solutions where the instance of the target peer may be augmented with data from the source. However, the target peer does not have the authority or ability to interfere with the source's data, which therefore remain unchanged.

Definition 2.2. Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a PDE setting, I a source instance, and J a target instance such that all elements in the active domain of (I, J) are constants. We say that a target instance J' is a *solution for* (I, J) *in* $\mathcal{P}$ if

—$J \subseteq J'$;
—$(I, J') \models \Sigma_{st} \cup \Sigma_{ts}$;
—$J' \models \Sigma_t$.

Note that in the above definition, we could have allowed source constraints. This, however, would not have changed the concept of a solution, as the source instance remains unchanged. This definition generalizes the notion of solution in data exchange settings [Fagin et al. 2005a] in two ways. The first and more significant one is the presence of the target-to-source dependencies Σ_{ts}; the second is that the input has a target instance J, in addition to the source instance I. Thus, data exchange settings are a special case of PDE settings where both Σ_{ts} and J are empty.

As noted earlier, tuple-generating dependencies are GLAV constraints that generalize both LAV and GAV constraints in data integration systems. Our PDE framework is able to capture GLAV with exact views in data integration systems [Lenzerini 2002]. Indeed, consider an exact GLAV constraint $\forall \mathbf{x}(Q_1(\mathbf{x}) \leftrightarrow Q_2(\mathbf{x}))$, where $Q_1(\mathbf{x})$ is a conjunctive query $\exists \mathbf{y}\phi_s(\mathbf{x}, \mathbf{y})$ over the source, and $Q_2(\mathbf{x})$ is a conjunctive query $\exists \mathbf{z}\psi_t(\mathbf{x}, \mathbf{z})$ over the target. Then the above GLAV constraint is equivalent to the source-to-target dependency $\forall \mathbf{x}\forall \mathbf{y}(\phi_s(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi_t(\mathbf{x}, \mathbf{z}))$ and the target-to-source dependency $\forall \mathbf{x}\forall \mathbf{z}(\psi_t(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\phi_s(\mathbf{x}, \mathbf{y}))$.

Although the definition of PDE setting involves two peers, it can be easily extended to a family of source peers exchanging data with the same target peer. Assume that $\mathbf{S}_1, \ldots, \mathbf{S}_n, \mathbf{T}$ are pairwise disjoint schemas. A *multi-PDE setting* is a family $\mathcal{P}_1 = (\mathbf{S}_1, \mathbf{T}, \Sigma_{s_1t}, \Sigma_{ts_1}, \Sigma_{t_1}), \ldots, \mathcal{P}_n = (\mathbf{S}_n, \mathbf{T}, \Sigma_{s_nt}, \Sigma_{ts_n}, \Sigma_{t_n})$ of PDE settings. Given instances $I_1, \ldots, I_n$ of the source peers, and an instance J of the target peer, a *solution J' for $((I_1, \ldots, I_n), J)$ in $\mathcal{P}_1, \ldots, \mathcal{P}_n$* is a target instance J' containing J such that J' is a solution for (I_m, J) in $\mathcal{P}_m$, for every $m \leq n$. Note that, in defining multi-PDE settings, we could have allowed constraints on the sources $\mathbf{S}_1, \ldots, \mathbf{S}_n$, as well as constraints between these sources. This, however, would have no impact on which target instances are solutions, as the source instances have to remain unchanged.

It is clear that J' is a solution for $((I_1, \ldots, I_n), J)$ in $\mathcal{P}_1, \ldots, \mathcal{P}_n$ if and only if J' is a solution for $(I_1 \cup \cdots \cup I_n, J)$ in the PDE setting $(\mathbf{S}_1 \cup \cdots \cup \mathbf{S}_n, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$, where $\Sigma_{st} = \cup_{m=1}^{n} \Sigma_{s_mt}$, $\Sigma_{ts} = \cup_{m=1}^{n} \Sigma_{ts_m}$, and $\Sigma_t = \cup_{m=1}^{n} \Sigma_{t_m}$. Thus, every multi-PDE setting can be simulated by a single PDE that has the same space of solutions as the original multi-PDE.

2.3 Algorithmic Problems in PDE Settings

Given a source instance I and a target instance J of a PDE setting $\mathcal{P}$, a solution for (I, J) may or may not exist; furthermore, if a solution exists, it need not be unique up to isomorphism.

Example 1. Let $\mathcal{P}$ be a PDE setting in which the source schema and the target schema consist of the binary relation symbols E and H respectively, and the constraints are as follows:

$$\begin{aligned} \Sigma_{st} &: E(x,z) \wedge E(z,y) \rightarrow H(x,y) \\ \Sigma_{ts} &: H(x,y) \rightarrow E(x,y) \\ \Sigma_{t} &: \emptyset \quad \text{(no target constraints)} \end{aligned}$$

If $I = \{E(a,b), E(b,c)\}$, where a, b, and c are distinct constants, and $J = \emptyset$, then no solution for (I, J) exists. If $I = \{E(a,a)\}$ and $J = \emptyset$, then $J' = \{H(a,a)\}$ is the only solution for (I, J). If $I = \{E(a,b), E(b,c), E(a,c)\}$ and $J = \emptyset$, then both $\{H(a,c)\}$ and $\{H(a,b), H(b,c), H(a,c)\}$ are solutions for (I, J).

This example illustrates a basic difference between data exchange settings and peer data exchange settings. Specifically, if a data exchange setting has no target constraints ($\Sigma_t = \emptyset$), then, for every source instance I, a solution always exists. As seen above, however, this need not be true for peer data exchange settings with $\Sigma_t = \emptyset$ and $J = \emptyset$. We will study in depth the problem of deciding the existence of a solution in a peer data exchange setting, and we will unveil deeper differences between data exchange and peer data exchange.

Definition 2.3. Assume that $\mathcal{P}$ is a PDE setting. The *existence-of-solutions problem for* $\mathcal{P}$, denoted by $\texttt{SOL}(\mathcal{P})$, is the following decision problem: given a source instance I and a target instance J such that all elements of the active domain of (I, J) are constants, is there a solution J' for (I, J) in $\mathcal{P}$?

The other basic algorithmic problem that we will study is that of obtaining the *certain answers* of target queries in PDE settings. The definition of certain answers we use is an adaptation of the standard concept used in incomplete databases [Grahne 1991; van der Meyden 1998] and information integration [Abiteboul and Duschka 1998; Lenzerini 2002]; in our context, this means that the set of "possible" worlds is the set of all solutions for a given source instance and a given target instance in a PDE setting.

Definition 2.4. Let $\mathcal{P}$ be a PDE setting and q a query over the target schema of $\mathcal{P}$. Let also I be a source instance and J a target instance such that all elements of the active domain of (I, J) are constants.

A tuple $\mathbf{t}$ is a *certain answer of* q *on* (I, J), denoted $\mathbf{t} \in \mathbf{certain}(q, (I, J))$, if $J' \models q[\mathbf{t}]$, for every solution J' for (I, J) in $\mathcal{P}$. We write $\mathbf{certain}(q, (I, J))$ to denote the set of all certain answers of q on (I, J). If q is a Boolean query, then $\mathbf{certain}(q, (I, J)) = \texttt{true}$ if $J' \models q$, for every solution J' for (I, J) in $\mathcal{P}$; otherwise, $\mathbf{certain}(q, (I, J)) = \texttt{false}$. Note that if q is a Boolean query, then computing the certain answers of q in the PDE setting $\mathcal{P}$ is a decision problem.

Let q be the Boolean query $\exists x \exists y \exists z (H(x, y) \wedge H(y, z))$, and consider the PDE setting in Example 1. Then $\mathbf{certain}(q, (\{E(a, a)\}, \emptyset)) = \texttt{true}$; on the other hand, $\mathbf{certain}(q, (\{E(a, b), E(b, c), E(a, c)\}, \emptyset)) = \texttt{false}$.

2.4 Relationship to PDMS

Peer data management systems (PDMS), formalized and studied by Halevy et al. [2005], constitute a decentralized, extensible architecture in which peers interact with each other in sharing and exchanging data. As mentioned in the Introduction, every PDE setting is a special case of a PDMS. In this section, we describe the relationship between peer data exchange settings and peer data management systems in precise terms.

According to Halevy et al. [2005], a PDMS $\mathcal{N}$ with peers $P_1, \ldots, P_n$ has the following characteristics:

—Each peer P_i has its own *peer schema*, which is disjoint from those of the other peers, but visible to all other peers.

—Each peer schema is a mediated global schema over a set of local sources that are accessible only by the peer (thus each peer can be viewed a data integration system). We will call the schema of the local sources a *local schema*. The relationship between the peer schema and the local schema is specified using *storage descriptions* that are *containment descriptions* $R \subseteq Q$ or *equality* *descriptions* $R = Q$, where R is one of the relations in the peer schema, and Q is a query over the local schema of the peer.

—The relationship between peers is specified using three types of *peer mappings*: *inclusion mappings*, *equality mappings*, and *definitional mappings*, where
  (1) each inclusion mapping is a containment $Q_1(\mathcal{A}_1) \subseteq Q_2(\mathcal{A}_2)$ between conjunctive queries $Q_1(\mathcal{A}_1)$ and $Q_2(\mathcal{A}_2)$, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are subsets of the set of all relations in the peer schemas;
  (2) each equality mapping is an equality $Q_1(\mathcal{A}_1) = Q_2(\mathcal{A}_2)$ between conjunctive queries $Q_1(\mathcal{A}_1)$ and $Q_2(\mathcal{A}_2)$ as above;
  (3) each definitional mapping is a Datalog program with rules having single relations from the peer schemas in both the head and the body of each rule.

Fix a PDMS $\mathcal{N}$. A *stored instance of* $\mathcal{N}$ is a set of relations conforming with the local schemas of the peers of $\mathcal{N}$. A *peer instance of* $\mathcal{N}$ is a set of relations conforming with the peer schemas of $\mathcal{N}$ (in the terminology of Halevy et al. [2005], a peer instance is called a *data instance*). A peer instance G is *consistent* with $\mathcal{N}$ and a stored instance D if G together with D satisfy all the specifications given by the storage descriptions and the peer mappings of $\mathcal{N}$ (see [Halevy et al. 2005] for the precise definition). This concept captures what it means for a peer instance to be a *solution* for a given set of stored relations in the PDMS $\mathcal{N}$.

We now have all the necessary background to spell out the relationship between peer data exchange settings and peer data management systems. Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a PDE setting. We claim that there is a PDMS $\mathcal{N}(\mathcal{P})$ with two peers $\mathbf{S}$ and $\mathbf{T}$ such that the solutions for a given instance in $\mathcal{P}$ essentially coincide with the consistent peer instances for a corresponding stored instance in $\mathcal{N}(\mathcal{P})$. The specification of the PDMS $\mathcal{N}(\mathcal{P})$ is as follows:

—The peer mappings of $\mathcal{N}(\mathcal{P})$ are given by the dependencies in $\Sigma_{st} \cup \Sigma_{ts} \cup \Sigma_t$. In particular, $\mathcal{N}(\mathcal{P})$ has no definitional mappings.

—For every relation symbol S_i in the schema of $\mathbf{S}$, there is a relation symbol S_i^* in the local schema of $\mathbf{S}$ of the same arity as S_i, and an equality storage description $S_i^* = S_i$.

—For every relation symbol T_j in the schema of $\mathbf{T}$, there is a relation symbol T_j^* in the local schema of $\mathbf{T}$ of the same arity as T_j, and a containment storage description $T_j^* \subseteq T_j$.

Note that the local and peer schemas of $\mathbf{S}$ and $\mathbf{T}$ in $\mathcal{N}(\mathcal{P})$ are replicas of the schemas of $\mathbf{S}$ and $\mathbf{T}$ in $\mathcal{P}$. Intuitively, the equality storage descriptions for $\mathbf{S}$ capture the fact that in peer data exchange the data of the source peer remain unchanged, whereas the containment storage descriptions for $\mathbf{T}$ capture the fact that in peer data exchange the data of the target peer may be augmented with new data. Let I be a source instance and let J be a target instance of $\mathcal{P}$. It is now easy to verify that K is a solution for (I, J) in $\mathcal{P}$ if and only if (I, K) is a consistent peer instance for the stored instance (I^*, J^*) of

$\mathcal{N}(\mathcal{P})$, where I^* and J^* are copies of I and J over the local sources of $\mathbf{S}$ and $\mathbf{T}$, respectively.

In conclusion, every PDE setting can be viewed as a PDMS with equality storage descriptions $S_i^* = S_i$ for the source peer, containment storage descriptions $T_j^* \subseteq T_j$ for the target peer, and peer mappings given by the constraints of the PDE.

There are peer data management systems for which testing for the existence of solutions and computing the certain answers of conjunctive queries are undecidable problems [Halevy et al. 2005]. We will show that the state of affairs is quite different for peer data exchange settings.

3. COMPLEXITY

Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a fixed peer data exchange setting. In this section, we show that the existence-of-solutions problem for $\mathcal{P}$ is in NP, where Σ_{st} and Σ_{ts} are arbitrary finite sets of source-to-target tgds and target-to-source tgds, and Σ_t is assumed to be the union of a finite set of target egds with a *weakly acyclic* finite set of target tgds. For such settings, the data complexity of the certain answers of unions of conjunctive queries is in coNP. We also show that there are PDE settings with $\Sigma_t = \emptyset$ for which the existence-of-solutions problem is NP-complete, and the data complexity of the certain answers of conjunctive queries is coNP-complete.

These results about peer data exchange settings contrast sharply both with results about peer data management systems and with results about data exchange settings. As mentioned earlier, there are PDMS for which these problems are undecidable [Halevy et al. 2005]. For data exchange settings in which Σ_{st} is an arbitrary finite set of source-to-target tgds and Σ_t is the union of a finite set of target egds with a weakly acyclic finite set of target tgds (recall that in data exchange settings there are no target-to-source tgds), these problems are solvable in polynomial time [Fagin et al. 2005a]. In fact, if $\Sigma_t = \emptyset$, then the existence-of-solutions problem is trivial, as solutions always exists.

3.1 Upper Bound

The concept of a *weakly acyclic* set of tgds was formulated by A. Deutsch and L. Popa in 2001, and independently used in Fagin et al. [2003] and Deutsch and Tannen [2003] (in the latter article, under the term *constraints with stratified witness*). The *chase procedure* terminates in polynomial time on such sets of tgds. Intuitively, weak acyclicity is a syntactic condition placed on sets of tgds to ensure that a chase step does not use labeled nulls from an attribute to create new labeled nulls in the same attribute. This ensures that the chase sequence is finite and, actually, that the chase procedure terminates in polynomial time [Fagin et al. 2005a]. For schema mappings where Σ_t is the union of a weakly acyclic set of tgds with a set of egds, the existence-of-solutions problem can be checked in polynomial time using the chase procedure. Without the weak acyclicity condition on the set of target tgds, it has been shown that the existence-of-solutions problem may be undecidable [Kolaitis et al. 2006].

Definition 3.1 (*Weakly Acyclic Set of tgds*). Let Σ be a set of tgds over a fixed schema. Construct a directed graph, called the *dependency graph*, as follows:

(1) There is a node for every pair (R, A) with R a relation symbol of the schema and A an attribute of R; call such a pair (R, A) a *position*.

(2) Add edges as follows: for every tgd $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ in Σ and for every x in $\mathbf{x}$ that occurs in ψ, and for every occurrence of x in ϕ in position (R, A_i)
   (a) for every occurrence of x in ψ in position (S, B_j), add an edge $(R, A_i) \rightarrow (S, B_j)$ (if it does not already exist);
   (b) in addition, for every existentially quantified variable y and for every occurrence of y in ψ in position (T, C_k), add a *special edge* $(R, A_i) \rightarrow (T, C_k)$ (if it does not already exists).

Note that there may be two edges in the same direction between two nodes but exactly one of the two edges is special. Then Σ is *weakly acyclic* if the dependency graph has no cycle going through a special edge.

For example, the tgd $E(x, y) \rightarrow \exists z E(x, z)$ is weakly acyclic; in contrast, the tgd $E(x, y) \rightarrow \exists z E(y, z)$ is not, because the dependency graph contains a self-loop. More precisely, let (E, A) and (E, B) be the first and second positions of E. The dependency graph consists of an edge from (E, B) to (E, A) and a special self-loop on (E, B). It should be noted that weakly acyclic sets of tgds include as a special case sets of full tgds, that is, tgds of the form $\forall \mathbf{x}(\varphi(\mathbf{x}) \rightarrow \psi(\mathbf{x}))$ in which no existentially quantified variables occur in the right-hand side. They also include acyclic sets of inclusion dependencies as a special case.

To obtain the complexity upper bounds, we extend the chase procedure as used in [Fagin et al. 2005a] to what we call a *solution-aware* chase procedure. Intuitively, this procedure chases an instance K_1 that does not satisfy a set of tgds with a solution K that contains K_1 and satisfies the tgds. Each *solution-aware chase step* uses values from K to make K_1 satisfy a tgd. Instead of creating labeled nulls to witness the existential variables of a tgd during a chase step, a solution-aware chase step uses values from the given solution K as witnesses. These values are guaranteed to exist since K contains K_1 and satisfies the tgds. Note that values from K are used only when a chase step is applied with a tgd that contains existential variables. Our definition of a solution-aware chase step makes use of the concept of a *homomorphism*, which is defined as follows.

Let K_1 and K_2 be two instances over the same schema. A *homomorphism* $h : K_1 \rightarrow K_2$ is a mapping from $\underline{\text{Const}} \cup \mathit{Var}$ to $\underline{\text{Const}} \cup \mathit{Var}$ such that

(1) $h(c) = c$, for every c in $\underline{\text{Const}}$;

(2) for every fact $R(t)$ of K_1, we have that $R(h(t))$ is a fact of K_2.

Here, $\underline{\text{Const}}$ denotes the set of all values that occur in the input (source and target) instances, *Var* refers to an infinite set of labeled nulls, t denotes a tuple $(a_1, \ldots, a_n)$, and $h(t)$ denotes the tuple $(h(a_1), \ldots, h(a_n))$; we assume that $\mathit{Var} \cap \underline{\text{Const}} = \emptyset$. Similarly, a *homomorphism from a conjunction* $\phi(\mathbf{x})$ *of atomic formulas to an instance* K is a mapping from the variables in $\mathbf{x}$ to $\underline{\text{Const}} \cup \mathit{Var}$

such that for every atom $R(x_1, \dots, x_n)$ in $\phi(\mathbf{x})$, we have that $R(h(x_1), \dots, h(x_n))$ is a fact in K.

We are now ready to give the definition of *solution-aware chase step* and *solution-aware chase*.

Definition 3.2 (*Solution-Aware Chase Step*). Let K_1 be an instance.

—(*tgd*). Let d be a tgd $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}))$. Let K be an instance that contains K_1 such that K satisfies d. Let h be a homomorphism from $\phi(\mathbf{x})$ to K_1 such that there is no extension of h to a homomorphism h' from $\phi(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y})$ to K_1. We say that *d can be applied to K_1 with homomorphism h and solution K*, or simply, *d can be applied to K_1 with homomorphism h* if K is understood from context.

Let h' be an extension of h such that every variable in $\mathbf{y}$ is assigned a value in K and $h' : \psi(\mathbf{x}, \mathbf{y}) \rightarrow K$ and let $K_2 = K_1 \cup h'(\psi(\mathbf{x}, \mathbf{y}))$, where $h'(\psi(\mathbf{x}, \mathbf{y})) = \{R(h(z_1), \dots, h(z_n)) : R(z_1, \dots, z_n)$ is an atom of $\psi(\mathbf{x}, \mathbf{y})\}$. We say that *the result of applying d to K_1 with h and solution K is K_2*, and write $K_1 \xrightarrow{d,h,K} K_2$. We drop K and write $K_1 \xrightarrow{d,h} K_2$ if K is understood from the context.

—(*egd*). Let d be an egd $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow (x_1 = x_2))$. Let h be a homomorphism from $\phi(\mathbf{x})$ to K_1 such that $h(x_1) \neq h(x_2)$. We say that *d can be applied to K_1 with homomorphism h*. We distinguish two cases:

—If both $h(x_1)$ and $h(x_2)$ are in $\underline{\text{Const}}$ then we say that *the result of applying d to K_1 with h is "failure,"* and write $K_1 \xrightarrow{d,h} \perp$.

—Otherwise, let K_2 be K_1 where we identify $h(x_1)$ and $h(x_2)$ as follows: if one is a constant, then the labeled null is replaced everywhere by the constant; if both are labeled nulls, then one is replaced everywhere by the other. We say that *the result of applying d to K_1 with h is K_2*, and write $K_1 \xrightarrow{d,h} K_2$.

Definition 3.3 (*Solution-Aware Chase*). Let Σ be a set of tgds and egds. Let K be an instance and K' be an instance that contains K and satisfies the set of tgds in Σ.

—A *solution-aware chase sequence of K with Σ and K'* is a sequence (finite or infinite) of solution-aware chase steps $K_i \xrightarrow{d_i,h_i} K_{i+1}$, with $i = 0, 1, \dots$, with $K = K_0$ and d_i a dependency in Σ.

—A *finite solution-aware chase of K with Σ and K'* is a finite solution-aware chase sequence $K_i \xrightarrow{d_i,h_i} K_{i+1}$, $0 \leq i \leq m$, with the requirement that either (a) $K_m = \perp$ or (b) there is no dependency d_i of Σ and there is no homomorphism h_i such that d_i can be applied to K_m with h_i. We say that K_m is the result of the finite solution-aware chase. We refer to case (a) as the case of a *failing finite solution-aware chase* and we refer to case (b) as the case of a *successful finite solution-aware chase*.

It was shown in Fagin et al. [2005a] that the length of every chase sequence of an instance with the union of a set of egds and a set of weakly acyclic tgds is bounded by a polynomial in the size of the instance. The next lemma shows that the same property holds even when each chase step is *solution-aware*.

LEMMA 3.4. *Let Σ be the union of a finite set of egds with a weakly acyclic finite set of tgds on some schema. Then there exists a polynomial $p(x)$ having the following property: if K and K' are instances such that K' contains K, and such that K' satisfies Σ, then the length of every solution-aware chase sequence of K with Σ and K' is bounded by $p(|K|)$, where $|K|$ is the size of K.*

PROOF. We first show that every intermediate instance in a solution-aware chase sequence is contained in K'. Furthermore, each solution-aware chase step in the sequence must be the result of applying a tgd, and not an egd. After this, we show that the length of every solution-aware chase sequence of K with the tgds in Σ and K' is bounded by $p(|K|)$, where $|K|$ is the size of K.

Let Σ be a union of a weakly acyclic set of tgds with a set of egds and let a solution-aware chase sequence of K with Σ and K' be $K_i \xrightarrow{d_i,h_i} K_{i+1}$, with $i = 0, 1, \ldots$ and $K_0 = K$. For every K_i, we have K' contains K_i.

—*Base case:* For $i = 0$, we have $K_0 = K$. Obviously, K' contains K_0.

—*Inductive case:* For the induction hypothesis, we assume the claim is true for $i < m$. So in particular K_{m-1} is contained in K'. We show next that K_m is also contained in K'. Suppose d_{m-1} is a tgd of the form: $\forall\mathbf{x}(\phi(\mathbf{x}) \to \exists\mathbf{y}\psi(\mathbf{x},\mathbf{y}))$. By the definition of a solution-aware chase step, we have that K_m is $K_{m-1} \cup h'_{m-1}(\psi(\mathbf{x},\mathbf{y}))$ where h'_{m-1} is an extension of h_{m-1} such that each variable in $\mathbf{y}$ is assigned a value in K' and the facts of $h'_{m-1}(\psi(\mathbf{x},\mathbf{y}))$ are facts in K'. (Note that since K' is a solution that contains K_{m-1}, the extension of h_{m-1} to h'_{m-1} according to the definition of a solution-aware chase step is guaranteed to exist.) Since K_m is obtained by adding a set of facts in K' to K_{m-1} and K_{m-1} is contained in K', it follows that K_m is contained in K'. Suppose d_{m-1} is an egd of the form $\forall\mathbf{x}(\phi(\mathbf{x}) \to (x_1 = x_2))$. This means that h_{m-1} can be applied to K_{m-1} with $h_{m-1}(x_1) \neq h_{m-1}(x_2)$. This implies that K_{m-1} does not satisfy d_{m-1}. Since K' contains K_{m-1} and K' is a solution, we know that K_{m-1} satisfies d_{m-1}, which is a contradiction. This completes the proof of our claim.

From the proof of the claim above, it is easy to see that every solution-aware chase step in the chase sequence applies a tgd. Next we show that the length of every solution-aware chase sequence (with tgds) is bounded by $p(|K|)$, where $|K|$ is the size of K. The proof is similar to that of Theorem 3.9 of [Fagin et al. 2005a]. For every node (R, A) in the dependency graph of Σ, define an *incoming path* to be any (finite or infinite) path ending in (R, A). Define the *rank* of (R, A), denoted by $\underline{\text{rank}}(R, A)$, as the maximum number of special edges on any such incoming path. Since Σ is weakly acyclic, there are no cycles going through special edges. Thus $\underline{\text{rank}}(R, A)$ is finite. Let r be the maximum, over all positions (R, A), of $\underline{\text{rank}}(R, A)$, and let p be the total number of positions (R, A) in the schema (equal to the number of nodes in the graph). The latter number is a constant, since the schema is fixed. Moreover, r is at most p. Thus r is not only finite but bounded by a constant. The next observation is that we can partition the nodes in the dependency graph, according to their rank, into subsets $N_0, N_1, \ldots, N_r$, where N_i is the set of all nodes with rank i. Let n be the total number of distinct values (constants or labeled nulls) that occur in the instance K. Let K'' be any instance obtained from K after some arbitrary solution-aware chase sequence. We prove by induction on i the following claim:

For every i there exists a polynomial Q_i such that the number of distinct values that occur in all positions (R, A) of N_i, in K'', is at most $Q_i(n)$.

—*Base case:* If (R, A) is a position in N_0, then there are no incoming paths with special edges. Thus no new values are ever created at position (R, A) during the solution-aware chase. Hence, the values occurring at position (R, A) in K'' are among the n values of the original instance K. Since this is true for all the positions in N_0, we can then take $Q_0(n) = n$.

—*Inductive case:* The first kind of values that may occur at a position of N_i, in K'', are those that already occur at the same position in K. The number of such values is at most n. In addition, a value may occur at a position of N_i, in K'', for two reasons: by being *copied* from some position in N_j with $j \neq i$, during a solution-aware chase step, or by being *extracted* from a value in K', also during a solution-aware chase step. We count first how many values can be extracted from K'. Let (R, A) be some position of N_i. A value can be extracted from K' into (R, A) during a solution-aware chase step only due to special edges. But any special edge that may enter (R, A) must start at a node in $N_0 \cup \cdots \cup N_{i-1}$. Applying the inductive hypothesis, the number of distinct values that can exist in all the nodes in $N_0 \cup \cdots \cup N_{i-1}$ is bounded by $P(n) = Q_0(n) + \cdots + Q_{i-1}(n)$. Let d be the maximum number of special edges that enter a position, over all positions in the schema. Then for any distinct d-tuple of values in $N_0 \cup \cdots \cup N_{i-1}$ and for any dependency in Σ there is at most *one new distinct value* that can be extracted from K' into position (R, A). (This is a consequence of the solution-aware chase step definition and of how the special edges have been defined. Observe that we are guaranteed to find a tuple in K' to extract a value from since K' contains K and satisfies the tgds in Σ.) Thus the total number of distinct values that can be extracted from K' into (R, A) is at most $(P(n))^d \times D$, where D is the number of dependencies in Σ. Since the schema and Σ are fixed, this is still a polynomial in n. If we consider *all* positions (R, A) in N_i, the total number of values that can be generated is at most $p_i \times (P(n))^d \times D$ where p_i is the number of positions in N_i. Let $G(n) = p_i \times (P(n))^d \times D$. Obviously, G is a polynomial.

We count next the number of distinct values that can be copied to positions of N_i from positions of N_j with $j \neq i$. Such copying can happen only if there are nonspecial edges from positions in N_j with $j \neq i$ to positions in N_i. We observe first that such nonspecial edges can originate only at nodes in $N_0 \cup \cdots \cup N_{i-1}$, that is, they cannot originate at nodes in N_j with $j > i$. Otherwise, assume that there exists $j > i$ and there exists a nonspecial edge from some position of N_j to a position (R, A) of N_i. Then the rank of (R, A) would have to be larger than i, which is a contradiction. Hence, the number of distinct values that can be copied in positions of N_i is bounded by the total number of values in $N_0 \cup \cdots \cup N_{i-1}$, which is $P(n)$ from our previous consideration. Putting it all together, we can take $Q_i(n) = n + G(n) + P(n)$. Since Q_i is a polynomial, the claim is proven.

In the above claim, i is bounded by the maximum rank r, which is a constant. Hence, there exists a fixed polynomial Q such that the number of distinct values that can exist, over all positions in K'' is bounded by $Q(n)$. In particular, the number of distinct values that can exist, at a single position in K'' is also bounded by $Q(n)$. Then the total number of tuples that can exist in K'' is at most $(Q(n))^p$ (recall that p is the total number of positions in the schema). This is also a polynomial since p is constant. Since every solution-aware chase step with a tgd adds at least some tuple to K'', it follows that the length of any chase sequence is at most $(Q(n))^p$. □

Using Lemma 3.4, we can show that whenever a solution for (I, J) exists in a PDE in which Σ_t is the union of a finite set of egds with a weakly acyclic finite set of tgds, then a *small* solution must exist, where *small* means that its size is polynomially bounded by the size of (I, J).

LEMMA 3.5. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a PDE setting in which Σ_t is the union of a finite set of egds with a weakly acyclic finite set of tgds. Let I be a source instance and J be a target instance. If there exists a solution J' for (I, J), then there exists a solution J^* for (I, J) that is contained in J' and has size bounded by a polynomial in the size of (I, J).*

PROOF. The instance J^* can be constructed as follows: let Σ consists of $\Sigma_{st} \cup \Sigma_t$. Note that the union of a weakly acyclic set of tgds over the target schema $\mathbf{T}$ with a set of source-to-target tgds is still a weakly acyclic set of tgds over the schema $\mathbf{S} \cup \mathbf{T}$. Hence, Σ is the union of a weakly acyclic set of tgds with a set of egds, over the schema $\mathbf{S} \cup \mathbf{T}$. We perform a solution-aware chase of (I, J) with Σ and (I, J'). Let J^* denote the set of tuples over $\mathbf{T}$ in the result of the solution-aware chase. It is easy to see that J^* is contained in J' and J^* is a solution for (I, J). Since (I, J') contains (I, J) and (I, J') satisfies Σ, we know from Lemma 3.4 that J^* is polynomial in the size of (I, J). □

Using Lemmas 3.4 and 3.5, we can easily derive the following result.

THEOREM 3.6. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a PDE setting in which Σ_t is the union of a finite set of egds with a weakly acyclic finite set of tgds. The existence-of-solutions problem $\mathit{SOL}(\mathcal{P})$ for $\mathcal{P}$ is in* NP.

PROOF. From Lemma 3.5, if there is a solution for (I, J), then there is a solution (I, J^*) that is polynomial in the size of (I, J). Checking that $(I, J^*) \models \Sigma_{st}$, $(J^*, I) \models \Sigma_{ts}$ and $J^* \models \Sigma_t$ can be done in polynomial time in the size of (I, J) since the peer data exchange is fixed. □

By definition, a query q is *monotone* if it is preserved under the addition of tuples, that is, if $\mathbf{t} \in q(K)$ and $K \subseteq K'$, then $\mathbf{t} \in q(K')$. Clearly, unions of conjunctive queries are monotone queries.

THEOREM 3.7. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a PDE setting in which Σ_t is the union of a finite set of egds with a weakly acyclic finite set of tgds. If q is a monotone query over $\mathbf{T}$, then computing the certain answers of q is in* coNP.

PROOF. Let q be a monotone k-ary query over $\mathbf{T}$. To show that the complement of the certain answers of q is in NP, it suffices to show that there is a polynomial $p(n)$ such that, for every source instance I, every target instance J, and every tuple $\mathbf{t}$ of length k, we have that $\mathbf{t} \notin \text{certain}(q, (I, J))$ if and only if there is a solution J^* for (I, J) of size bounded by $p(|(I, J)|)$ such that $\mathbf{t} \notin q(J^*)$, where $|(I, J)|$ is the size of (I, J). Clearly, if there is a solution J^* for (I, J) of any size such that $\mathbf{t} \notin q(J^*)$, then $\mathbf{t} \notin \text{certain}(q, (I, J))$. For the other direction, recall that, by Lemma 3.5, there is a polynomial $p(n)$ such that if J' is a solution for (I, J), then there is a solution J^* for (I, J) that is contained in J' and has size bounded by $p(|(I, J)|)$. If $\mathbf{t} \notin \text{certain}(q, (I, J))$, then there is a solution J' for (I, J) such that $\mathbf{t} \notin q(J')$. Consequently, there is a solution J^* for (I, J) that is contained in J' and has size bounded by $p(|(I, J)|)$. Since q is a monotone query, it follows that $\mathbf{t} \notin q(J^*)$. □

3.2 Lower Bound

We show next that there are PDE settings with no target constraints for which the existence-of-solutions problem is NP-hard, and computing the certain answers of target conjunctive queries is coNP-hard. Although the latter result could be derived from Abiteboul and Duschka [1998, Theorem 5.1], and Grahne and Mendelzon [1999], Theorem 8, we give a self-contained proof using a particularly simple reduction from the 3-COLORABILITY problem whose features we will analyze later on.

THEOREM 3.8. *There exists a peer data exchange setting $\mathcal{P}$ with $\Sigma_t = \emptyset$ for which the existence-of-solutions problem is* NP-*complete. Moreover, there is a Boolean conjunctive query q such that computing the certain answers of q in $\mathcal{P}$ is a* coNP-*complete problem.*

PROOF. From Theorem 3.6, we know that, for every fixed peer data exchange setting without target constraints, the existence of solutions problem is in NP. The lower bound is established by reducing graph 3-COLORABILITY to the existence-of-solutions problem for some particular peer data exchange setting. As usual, a *graph* is a structure $G = (V, E)$, where V is a set of nodes and $E \subseteq V^2$ is a binary relation which is symmetric and irreflexive (no self-loops).

Let $\mathcal{P}$ be the following peer data exchange setting. The source schema $\mathbf{S}$ consists of two binary relations E, F, and a unary relation H, while the target schema $\mathbf{T}$ consists of a binary relation C. The constraints between $\mathbf{S}$ and $\mathbf{T}$ are as follows:

$$\begin{aligned} \Sigma_{st} &: E(x, y) \rightarrow \exists u C(x, u) \\ \Sigma_{ts} &: C(x, u) \rightarrow H(u) \\ &\quad C(x, u) \wedge C(y, u) \rightarrow F(x, y) \end{aligned}$$

We now show that 3-COLORABILITY has a polynomial-time reduction to SOL($\mathcal{P}$).

Given a graph $G = (V, E)$ with no isolated nodes, consider the source instance $I(G) = (E, F, H)$ and the target instance $\emptyset$ of the above PDE setting, where $F = V^2 \setminus E$ is the complement of the edge relation of G, and $H = \{r, g, b\}$ is a set of three colors none of which is an element of V. Note that F contains all self-loops on V. Clearly, E, F, and H can be constructed in polynomial time from $G = (V, E)$.

We claim that the graph $G = (V, E)$ is 3-colorable if and only if there is a solution for the source instance (E, F, H) and the target instance $\emptyset$. If the graph is 3-colorable, then every 3-coloring of $G = (V, E)$ gives rise to a solution C such that $C(x, c)$ holds precisely when c is the color of x. Conversely, if there is a solution for the source instance (E, F, H), then we can use the single tgd in Σ_{st} and the first tgd in Σ_{ts} to select, for every node x of G, a color $c(x)$ in $H = \{r, g, b\}$ such that $C(x, c(x))$. To see that this mapping is a 3-coloring of G, suppose that there is an edge $E(x, y)$ such that $c(x) = c(y)$. Then, the second tgd in Σ_{ts} implies that $F(x, y)$ holds, which is a contradiction, since $F = V^2 \setminus E$.

Let q be the Boolean query $\exists x C(x, x)$. It is easy to verify that the graph $G = (V, E)$ is 3-colorable if and only if **certain**$(q, (I(G), \emptyset))$ = `false`. Thus, computing the certain answers of the query $\exists x C(x, x)$ is a coNP-hard problem. □

In Halevy et al. [2005], it was shown that if in a PDMS all storage descriptions are containment descriptions and all peer mappings are inclusion mappings with an acyclic *dependency* graph, then the certain answers of conjunctive queries are computable in polynomial time. The *dependency* graph of a PDMS is the directed graph with nodes that are the relations of the peers, and edges between two relations P and R if there is an inclusion peer mapping $Q_1(\mathcal{A}_1) \subseteq Q_2(\mathcal{A}_2)$ such that P occurs in $Q_1(\mathcal{A}_1)$ and R occurs in $Q_2(\mathcal{A}_2)$. Note that the PDE setting used in the reduction of Theorem 3.8 has inclusion peer mappings with an acyclic dependency graph, yet the problem of computing certain answers is coNP-hard. The jump in complexity arises due to the fact that in PDE settings the source instance can never change, which means that the constraints placed on storage descriptions in the source are not containment descriptions, but equality descriptions.

4. A LARGE TRACTABLE CLASS

In this section, we identify syntactic conditions on PDE settings with no target constraints that yield polynomial-time algorithms for deciding the existence of solutions. As seen in the proof of Theorem 3.8, even such strong topological conditions as the acyclicity of the dependency graph of source and target relations cannot guarantee tractability of these problems. Instead, we consider different conditions that are derived by taking a closer look at the existential quantifiers in the constraints of the PDE setting.

Definition 4.1. Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ be a PDE setting with no target constraints.

—We say that the ith position of a relation symbol T of $\mathbf{T}$ is *marked* if Σ_{st} contains a source-to-target tgd

$$\phi_s(\mathbf{x}) \to \exists \mathbf{y} \psi_t(\mathbf{x}, \mathbf{y})$$

such that $T(z_1, \ldots, z_i, \ldots, z_n)$ is one of the conjuncts of $\psi_t(\mathbf{x}, \mathbf{y})$, and z_i is one of the existentially quantified variables $\mathbf{y}$.

—We say that a variable z is *marked in a target-to-source tgd*

$$\alpha_t(\mathbf{x}) \to \exists \mathbf{w} \beta_s(\mathbf{x}, \mathbf{w})$$

of Σ_{ts} if one of the following two holds:
(1) z appears at a marked position of a conjunct of $\alpha_t(\mathbf{x})$.
(2) z is one of the existentially quantified variables $\mathbf{w}$.

Note that the two conditions in the definition of a marked variable are mutually exclusive.

To illustrate the concepts of marked position and marked variable, let us consider a PDE setting having the following constraints:

$$\begin{array}{ll} \Sigma_{st}: & S(x_1, x_2) \rightarrow \exists y \exists z T(x_1, y, x_2, z) \\ \Sigma_{ts}: & T(w_1, w_2, w_3, w_4) \rightarrow \exists u \exists v (S(w_1, u) \wedge S(w_3, v)) \end{array}$$

In this setting, the marked positions are the second and fourth positions of T, while the marked variables in the target-to-source dependency are w_2, w_4, u, and v.

Let us also consider the PDE setting in the proof of Theorem 3.8 used in the reduction from 3-Colorability:

$$\begin{array}{ll} \Sigma_{st}: & E(x, y) \rightarrow \exists u C(x, u) \\ \Sigma_{ts}: & C(x, u) \rightarrow H(u) \\ & C(x, u) \wedge C(y, u) \rightarrow F(x, y) \end{array}$$

In this setting, the only marked position is the second position of C. The only marked variable in the first tgd in Σ_{ts} is u and the only marked variable in the second tgd in Σ_{ts} is also u.

We now introduce the class $\mathcal{C}_{tract}$, which is the focus of this section. Below, if $\alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{w} \beta_s(\mathbf{x}, \mathbf{w})$ is a tgd in Σ_{ts}, we will refer to $\alpha_t(\mathbf{x})$ as the *left-hand side* of the tgd, and to $\exists \mathbf{w} \beta_s(\mathbf{x}, \mathbf{w})$ as the *right-hand side* of the tgd.

Definition 4.2. Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ be a PDE setting with no target constraints. We say that $\mathcal{P} \in \mathcal{C}_{tract}$ if

(1) for every tgd D in Σ_{ts}, every marked variable of D appears at most once in the left-hand side of D and
(2) for every tgd D in Σ_{ts}, one of the following conditions holds:
—2.1: the left-hand side of D consists of exactly one literal; or
—2.2: for every pair of marked variables x and y of D that appear together in a conjunct of the right-hand side of D, either
(a) x and y appear together in some conjunct of the left-hand side of D or
(b) x and y do not appear at all in the left-hand side of D.

Observe that conditions (2.1) and (2.2) in Definition 4.2 are not mutually exclusive. In particular, if a tgd $\alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{w} \beta_s(\mathbf{x}, \mathbf{w})$ in Σ_{ts} satisfies condition (2.2) of $\mathcal{C}_{tract}$ and $\alpha_t(\mathbf{x})$ is a single literal, then this tgd satisfies condition (2.1) as well. However, a tgd $\alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{w} \beta_s(\mathbf{x}, \mathbf{w})$ in Σ_{ts} may satisfy condition (2.1), but need not satisfy condition (2.2) at the same time; this happens precisely when there are two marked variables x_i and w_j such that x_i is among the variables of $\mathbf{x}$ (i.e., it appears at a marked position of a conjunct of $\alpha_t(\mathbf{x})$), w_j is one of the existentially quantified variables $\mathbf{w}$, and x_i and w_j appear together in some conjunct of $\beta_s(\mathbf{x}, \mathbf{w})$.

Admittedly, the definition of the class $\mathcal{C}_{tract}$ is quite technical. We arrived at it after carefully analyzing the causes of intractability in numerous concrete PDE settings, such as the one used in the reduction from the 3-COLORABILITY problem. To convey some feeling for $\mathcal{C}_{tract}$, we should point out that it is a rather broad class that contains several interesting families of PDE settings as subclasses.

Note that $\mathcal{C}_{tract}$ includes the union of two different (but not disjoint) classes:

(1) the class of all PDE setings in which every tgd in Σ_{ts} satisfies conditions (1) and (2.1);
(2) the class of all PDE settings in which every tgd in Σ_{ts} satisfies conditions (1) and (2.2).

The first of these classes can be described as the class of all PDE settings $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ in which every target-to-source tgd has exactly one literal in its left-hand side which has no repeated variables. Hence, this is the class of PDE settings in which the target-to-source tgds are *local-as-view* (LAV) dependencies, an important class in data integration [Lenzerini 2002].

The second class contains as a subclass the family of all PDE settings $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ in which every source-to-target tgd is a *full tgd*, which means that it is of the form $\phi_s(\mathbf{x}) \rightarrow \psi_t(\mathbf{x})$. Indeed, if every source-to-target tgd is full, then the only marked variables are the ones that are existentially quantified in some target-to-source tgd. As a result, condition (1) holds for such PDE settings. Moreover, if two marked variables appear together in the right-hand side of some target-to-source tgd D, then neither appears in the left-hand side of D, and hence condition (2.2) (b) is satisfied.

We are now ready to state the main result of this section.

THEOREM 4.3. *Let $\mathcal{P}$ be a PDE setting in $\mathcal{C}_{tract}$. Then, the existence-of-solutions problem SOL($\mathcal{P}$) for $\mathcal{P}$ is solvable in polynomial time. Moreover, if a solution exists, then a solution can be computed in polynomial time.*

The proof of Theorem 4.3 is rather complicated, and will be given in the next section; it uses a variant of the chase procedure, which we call the *naive chase*, and homomorphism techniques. In the remainder of this section, we first state some corollaries to Theorem 4.3 and then show that, in a certain sense, $\mathcal{C}_{tract}$ is a maximal class of tractable PDE settings.

COROLLARY 4.4. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ be a PDE setting with no target constraints. If Σ_{st} is a set of full dependencies, then the existence-of-solutions problem SOL($\mathcal{P}$) for $\mathcal{P}$ is solvable in polynomial time.*

COROLLARY 4.5. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ be a PDE setting with no target constraints. If every target-to-source dependency of Σ_{ts} is a LAV dependency (exactly one literal on its left-hand side which has no repeated variables), then the existence-of-solutions problem SOL($\mathcal{P}$) for $\mathcal{P}$ is solvable in polynomial time.*

We now show that the conditions defining $\mathcal{C}_{tract}$ are tight, in the sense that minimal relaxations of them lead to intractability.

4.1 Necessity of Condition (1)

In order to show that condition (1) in the definition of $\mathcal{C}_{tract}$ is indispensable, consider again the PDE setting:

$$\begin{aligned} \Sigma_{st} :&\ E(x, y) \to \exists u C(x, u) \\ \Sigma_{ts} :&\ C(x, u) \to H(u) \\ &\ C(x, u) \wedge C(y, u) \to F(x, y) \end{aligned}$$

This PDE setting satisfies condition (2) of $\mathcal{C}_{tract}$. However, it violates condition (1), because the marked variable u appears twice in the left-hand side of the second dependency of Σ_{ts}. As seen in the proof of Theorem 3.8, the existence-of-solutions problem for this PDE setting is NP-complete.

4.2 Necessity of Condition (2)

In order to show that condition (2) in the definition of $\mathcal{C}_{tract}$ is indispensable, consider the following PDE setting:

$$\begin{aligned} \Sigma_{st} :&\ E(x, y) \to \exists u C(x, u) \\ &\ E(x, y) \to E'(x, y) \\ \Sigma_{ts} :&\ C(x, u) \wedge C(y, v) \wedge E'(x, y) \to D(u, v) \end{aligned}$$

The marked variables in the target-to-source tgd are u and v. Clearly, this PDE setting satisfies condition (1), but fails to satisfy condition (2) in the definition of $\mathcal{C}_{tract}$, since u and v appear in the atom of the right-hand side of the dependency in Σ_{ts} but do not appear together in any conjunct of the left-hand side of that dependency.

Given a graph $G = (V, E)$, form the source instance (E, D) and the target instance $\emptyset$, where $D = \{(b, g), (g, b), (b, r), (r, b), (g, r), (r, g)\}$ is the inequality relation on three colors b, g, r. Then the graph $G = (V, E)$ is 3-colorable if and only if there is a solution for the source instance (E, D) and the target instance $\emptyset$.

4.3 PDEs with Target Constraints

Next, we show that the intractability boundary may be crossed if target constraints are allowed, while the source-to-target and the target-to-source dependencies satisfy the conditions of $\mathcal{C}_{tract}$.

4.3.1 *PDEs with target full tgds.* Consider the following PDE setting:

$$\begin{aligned} \Sigma_{st} :&\ E(x, y) \to \exists u C(x, u) \\ &\ E(x, y) \to E'(x, y) \\ \Sigma_{ts} :&\ D'(u, v) \to D(u, v) \\ \Sigma_{t} :&\ C(x, u) \wedge C(y, v) \wedge E'(x, y) \to D'(u, v) \end{aligned}$$

Note that the target-to-source tgd is a LAV dependency, so conditions (1) and (2) are satisfied.

Given a graph $G = (V, E)$, we form a source instance consisting of E and the inequality relation $D = \{(r, b), (b, r), (r, g), (g, r), (b, g), (g, b)\}$ on three colors r, b, g. Then the graph $G = (V, E)$ is 3-colorable if and only if a solution exists for the source instance (E, D') and the target instance $\emptyset$.

4.3.2 *PDEs with Target Egds.* Consider the following PDE setting:

$$\begin{aligned}
\Sigma_{st}: &\ E(x, y) \to \exists u \exists v C(x, u, y, v) \\
\Sigma_{ts}: &\ C(x, u, y, v) \to D(u, v) \\
\Sigma_{t}: &\ C(x, u, y, v) \wedge C(x, u', y', v') \to u = u' \\
&\ C(x, u, y, v) \wedge C(y', v', x, u') \to u = u'
\end{aligned}$$

Note that the target-to-source tgd is a LAV dependency, so conditions (1) and (2) are satisfied.

Given a graph $G = (V, E)$, we form, as before, a source instance consisting of E and the inequality relation $D = \{(r, b), (b, r), (r, g), (g, r), (b, g), (g, b)\}$ on three colors r, b, g. Then the graph $G = (V, E)$ is 3-colorable if and only if a solution exists for the source instance (E, D) and the target instance $\emptyset$. In this PDE setting, the two target egds enforce that every node is assigned a unique color.

4.4 PDEs with Disjunctive Target-to-Source Dependencies

Finally, we show that the intractability boundary may also be crossed if we allow disjunctions in the right-hand side of target-to-source tgds. For this, consider the following PDE setting:

$$\begin{aligned}
\Sigma_{st}: &\ E(x, y) \to \exists u C(x, u) \\
&\ E(x, y) \to E'(x, y) \\
\Sigma_{ts}: &\ E'(x, y) \wedge C(x, u) \wedge C(y, v) \to \\
&\ (R(u) \wedge B(v)) \vee (R(u) \wedge G(v)) \vee \\
&\ (B(u) \wedge G(v)) \vee (B(u) \wedge R(v)) \vee \\
&\ (G(u) \wedge R(v)) \vee (G(u) \wedge B(v))
\end{aligned}$$

The source relations are E, R, B, and G, while the target relations are E' and C. Given a graph E, we construct a source instance consisting of E, $R = \{r\}$, $G = \{g\}$, and $B = \{b\}$; we also take the target instance J to be empty. It is easy to see that E is 3-colorable if and only if there is a solution for this PDE setting. Note that Σ_{st} and Σ_{ts} satisfy conditions (1) and (2.2) of $\mathcal{C}_{tract}$, and there are no target constraints.

5. PROOF OF THEOREM 4.3

This section contains the proof of Theorem 4.3. To this effect, for every PDE setting $\mathcal{P}$ in $\mathcal{C}_{tract}$, we present an algorithm that solves the existence-of-solutions problem for $\mathcal{P}$ in polynomial time; moreover, given a source instance I and a target instance J, the algorithm computes a solution for (I, J), if a solution exists.

The proof makes use of the *naive chase*, which we define next. This is a variant of the chase procedure that is different from both the chase defined in [Fagin et al. 2005a] and from the solution-aware chase that was defined earlier in this article. Intuitively, in the naive chase, all tgds are applied simultaneously and nulls are generated independently of each other. Thus, each null is created by a single tgd and is never reused to witness the existential quantifiers of another tgd.

Definition 5.1 (*Naive Chase*). Let U and V be two disjoint schemas and let Σ be a set of U-V tgds of the form $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x})$ is a conjunction of atomic formulas over U and $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulas over V. Let K be a U-instance and let L be a V-instance.

(1) Let d be a tgd $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ in Σ.
   (a) For every homomorphism $h{:}\phi(\mathbf{x}) \rightarrow K$, extend h to h' by defining $h'(y)$ to be a fresh labeled null for each variable $y \in \mathbf{y}$. We say that such a homomorphism h is an *application of* d *to* (K, L).
   (b) We write L_d to denote the V-instance that is the union of the set of facts in $\psi(h'(\mathbf{x}), h'(\mathbf{y}))$, where h' is the above extension of a homomorphism h, and h ranges over all homomorphisms from $\phi(\mathbf{x})$ to K.

(2) The *naive chase of* (K, L) *with* Σ is the V-instance

$$L' = \left(\bigcup_{d \in \Sigma} L_d \right) \cup L.$$

As an example, suppose Σ consists of the following two tgds:

$$S(x, z) \rightarrow \exists y (T(x, y) \wedge T(y, z))$$
$$S(x, z) \rightarrow \exists y \, T(x, y)$$

Let K be the instance consisting of the single fact $S(a, b)$. The naive chase of $(K, \emptyset)$ with Σ is the instance consisting of the three facts $T(a, n_1)$, $T(n_1, b)$, and $T(a, n_2)$.

As for the chase procedure used in Fagin et al. [2005a], we get two different target instances depending on the order in which the tgds are applied. If the first tgd is applied before the second, then the target instance generated by the chase consists of the two facts $T(a, n_1)$ and $T(n_1, b)$. If the second tgd is applied before the first, then the target instance generated by the chase consists of three facts $T(a, n_1)$, $T(n_1, b)$, and $T(a, n_2)$.

Several remarks are in order now. First, observe that every fresh labeled null in the resulting instance that is generated by the naive chase (or the chase for that matter) is generated by *exactly one* application of a tgd in Σ; however, unlike the chase, every labeled null is used *only once* to witness an existential quantifier in the right-hand side of a tgd. This is because a null, once generated, is never reused in other applications of tgds. Second, as seen in the preceding example, the result of chasing an instance (under the definition of Fagin et al. [2005a]) is dependent on the order of applications of tgds and the homomorphisms. The naive chase of an instance, however, is always unique up to renaming of nulls, since the order of applications of tgds in the naive chase is irrelevant by definition. Finally, for a fixed Σ, the size of L' (the result of applying the naive chase) is polynomial in the size of (K, L).

The proof of Theorem 4.3 will make use of two other results (Theorem 5.3) and Theorem 5.4), which we state after introducing some notation and defining some additional concepts.

For the rest of this section, we assume that $\mathcal{P}$ is a PDE setting, I is a source instance, J is a target instance, J_{can} is the naive chase of (I, J) with Σ_{st}, and I_{can} is the naive chase $(J_{can}, \emptyset)$ with Σ_{ts}.

$$\begin{tikzcd}
(I,J) \arrow[rr, "\text{Naive chase with } \Sigma_{st}"] & & J_{can} \arrow[dll, "\text{Naive chase with } \Sigma_{ts}"] \\
I_{can} \arrow[u, dashed, "h"] & &
\end{tikzcd}$$

Fig. 2. Illustration of the method for solving the existence-of-solutions problem.

The concepts of the *graph of the nulls* and of a *block of nulls* of an instance were introduced in Fagin et al. [2005b]. The precise definition is as follows.

Definition 5.2. Let K be an instance.

(1) The *graph of the nulls* of K is the undirected graph defined as follows.
  (a) The nodes are the nulls of K.
  (b) There is an edge between two nulls if and only if the nulls appear together in some fact of K.

(2) A *block of nulls of* K is a connected component of the graph of the nulls of K.

In the sequel, when we write that a quantity is *bounded*, we mean that the quantity is bounded by some constant that depends only on a fixed PDE setting (and not on the instances).

We are now ready to state the two results needed to establish Theorem 4.3.

THEOREM 5.3. *Let $\mathcal{P}$ be a PDE setting that satisfies condition* (1) *of the definition of $\mathcal{C}_{tract}$. Then, there exists a solution for* (I, J) *in $\mathcal{P}$ if and only if there is a homomorphism from I_{can} to I.*

THEOREM 5.4. *Let $\mathcal{P}$ be a PDE setting that satisfies condition* (2) *of the definition of $\mathcal{C}_{tract}$. Then, the size of every block of nulls of I_{can} is bounded.*

Figure 2 illustrates our method, based on Theorem 5.3, for solving the existence-of-solutions problem for a PDE setting. For simplicity, we have depicted I_{can} as the naive chase of J_{can} (instead of the naive chase of $(J_{can}, \emptyset)$) with Σ_{ts}. Our algorithm determines whether a solution exists for a PDE setting that satisfies condition (1) of the definition of $\mathcal{C}_{tract}$ by checking whether there is a homomorphism from I_{can} to I. For this, we show that there is a homomorphism from I_{can} to I if and only if (a) every null-free tuple of I_{can} occurs in I; and (b) for every block of nulls $\mathcal{B}$ of I_{can}, there is a homomorphism from $I_{\mathcal{B}}$ to I, where $I_{\mathcal{B}}$ is the set of all tuples in I_{can} that contain at least one null from $\mathcal{B}$. It remains to show that if a PDE setting also satisfies condition (2) of the definition of $\mathcal{C}_{tract}$ (and, thus, is a PDE setting in $\mathcal{C}_{tract}$), then the preceding test can be carried out in polynomial time. From Theorem 5.4, we know that if a PDE setting satisfies condition (2) of the definition of $\mathcal{C}_{tract}$, then the size of every block of nulls of I_{can} is bounded. From this and the fact that the size of I_{can} is polynomial in the size of (I, J), it follows that the number of blocks of nulls of I_{can} is polynomial. Consequently, one can determine in polynomial time whether there is a homomorphism from I_{can} to I, since there are polynomially many blocks of nulls in I_{can} and the size of each block is bounded. Furthermore, if a solution exists, then our algorithm computes a solution in polynomial time.

The next two sections contain the proofs of Theorems 5.3 and 5.4. The proof of Theorem 4.3 is then given in Section 5.3.

5.1 Condition 1 and the Existence of Solutions

The proof of Theorem 5.3 makes use of an important lemma which we shall explain next. Let $\mathcal{P}$ be a PDE setting which satisfies condition (1) of the definition of $\mathcal{C}_{tract}$. Let $\alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{w} \beta_s(\mathbf{x}, \mathbf{w})$ be a target-to-source tgd in Σ_{ts} and let J_{can} be the naive chase of (I, J) with Σ_{st}. The next lemma states that if $J_{img} = h(J_{can})$, for some constant-preserving function h, then whenever $J_{img} \models \alpha_t(\mathbf{c})$ for some values $\mathbf{c}$ that occur in J_{img}, there are some values $\mathbf{d}$ that occur in J_{can} such that $h(\mathbf{d}) = \mathbf{c}$ and $J_{can} \models \alpha_t(\mathbf{d})$. It is easy to see that this property does not hold in general. Consider the following tgd that maps paths of length two to a single tuple:

$$T_1(x, y) \wedge T_2(y, z) \rightarrow S(x, z).$$

Let $J_{can} = \{T_1(a, N_1), T_2(N_2, c)\}$ and let h be a homomorphism such that $h(a) = a$, $h(N_1) = b$, $h(N_2) = b$ and $h(c) = c$. Let $J_{img} = h(J_{can})$. Hence, $J_{img} = \{T_1(a, b), T_2(b, c)\}$. It is easy to see that $J_{img} \models T_1(x, y) \wedge T_2(y, z)$ but J_{can} does not. Observe that variable y appears in two literals of the tgd and the null values N_1 and N_2 appear at the positions of y in the tuples of J_{can}. A null value in J_{can} can only appear at a marked position. Assuming that the above tgd is in Σ_{ts} of some PDE setting $\mathcal{P}$, the variable y is a marked variable. Thus, $\mathcal{P}$ violates condition (1) of $\mathcal{C}_{tract}$. We show next that, if condition (1) of $\mathcal{C}_{tract}$ is satisfied, we get the desired property. In the lemma below, we use $Dom(I)$ to denote the active domain of I. If $\mathbf{d} = (d_1, \ldots, d_m)$ is a tuple of values, we will write $\mathbf{d} \in Dom(I)$ to denote that $d_i \in Dom(I)$, for every $i \leq m$.

LEMMA 5.5. *Let $\mathcal{P}$ be a PDE setting that satisfies condition* (1) *of the definition of $\mathcal{C}_{tract}$. Let $\forall \mathbf{x}\ \alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{y}\ \beta_s(\mathbf{x}, \mathbf{y})$ be a dependency in Σ_{ts}. Define $J_{img} = h(J_{can})$, where h is a function that preserves constants. If $J_{img} \models \alpha_t(\mathbf{c})$, where $\mathbf{c} \in Dom(J_{img})$, then there exists $\mathbf{d} \in Dom(J_{can})$ such that $h(\mathbf{d}) = \mathbf{c}$ and $J_{can} \models \alpha_t(\mathbf{d})$.*

PROOF. Let $\alpha_t(\mathbf{x})$ denote the conjunction of relational atoms $T_1(\mathbf{x_1}) \wedge \cdots \wedge T_n(\mathbf{x_n})$, where T_i, $1 \leq i \leq n$, is the ith relational symbol in α_t, and $\mathbf{x_i}$ denotes the vector of variables in T_i. Construct a mapping v_1 so that v_1 maps x_i to c_i, where x_i denotes the ith variable in the vector of variables $\mathbf{x}$ of $\alpha_t(\mathbf{x})$, and c_i denotes the ith value in the vector $\mathbf{c}$ of $\alpha_t(\mathbf{c})$. It is easy to see that v_1 is a valid assignment, since $J_{img} \models \alpha_t(\mathbf{c})$. For every $1 \leq i \leq n$, we use $\mathbf{c_i}$ to denote the vector of values $v_1(\mathbf{x_i})$. Therefore, $T_1(v_1(\mathbf{x_1})) \wedge \cdots \wedge T_n(v_1(\mathbf{x_n}))$ is the same as $T_1(\mathbf{c_1}) \wedge \cdots \wedge T_n(\mathbf{c_n})$.

Since $J_{img} = h(J_{can})$, it follows that, for every $1 \leq i \leq n$, there is a tuple $T_i(\mathbf{d_i})$ in J_{can} such that $T_i(h(\mathbf{d_i})) = T_i(\mathbf{c_i})$. Hence, we have a sequence of tuples $T_1(\mathbf{d_1}), \ldots, T_n(\mathbf{d_n})$ from J_{can}. We will show that $\{T_1(\mathbf{d_1}), \ldots, T_n(\mathbf{d_n})\} \models T_1(\mathbf{x_1}) \wedge \cdots \wedge T_n(\mathbf{x_n})$. For this, construct a mapping v_2 that maps, for every $1 \leq i \leq n$, and $1 \leq j \leq |\mathbf{x_i}|$, the variable at the jth position of $\mathbf{x_i}$ to the value at the jth position of $\mathbf{d_i}$. We will show next that v_2 is a valid assignment: v_2 maps every relational atom $T_i(\mathbf{x_i})$ to a tuple in J_{can}, and v_2 is a function in that, if a variable z occurs at two distinct places among $\mathbf{x_1}, \ldots, \mathbf{x_n}$, then z is assigned the same value under the mapping scheme we have just described.

From the construction of v_2, it is easy to see that $T_i(v_2(\mathbf{x_i})) = T_i(\mathbf{d_i})$, which is a tuple in J_{can}. Therefore, it remains to be shown that v_2 is a function under the mapping scheme we have described. Suppose there is a variable z that occurs at two different positions, say $\mathbf{x}_i^p$ and $\mathbf{x}_j^q$ (the pth variable in $\mathbf{x_i}$, and the qth variable in $\mathbf{x_j}$), and z is assigned to two different values. That is, $\mathbf{d}_i^p \neq \mathbf{d}_j^q$ (the value of $\mathbf{d}_i$ at position p is not the same as the value of $\mathbf{d}_j$ at position q). Since $\mathcal{P}$ satisfies condition (1) of the definition of $\mathcal{C}_{tract}$, it follows that z cannot be a marked variable. Since nulls can only appear at marked positions in J_{can}, we have that $\mathbf{d}_i^p$ and $\mathbf{d}_j^q$ are constants. Now, since h maps constants to their identity, we have that $\mathbf{c}_i^p \neq \mathbf{c}_j^q$. So v_1 maps z to two different constants, and, therefore, v_1 is not a valid assignment; thus, a contradiction.

Hence, we have $J_{can} \models \alpha_t(\mathbf{d})$, where $\mathbf{d} = v_2(\mathbf{x})$. It is easy to see that $\mathbf{d} \in Dom(J_{can})$ and $h(\mathbf{d}) = \mathbf{c}$ by the construction of the mapping scheme v_2. □

Before we present the proof of Theorem 5.3, we state below a lemma which we will use repeatedly in the proof of Theorem 5.3.

LEMMA 5.6. *Let U and V be disjoint schemas. Let Σ be a set of U-V tgds. Let K, K'' be instances over U, and let L and L'' be instances over V such that L'' contains L, there is a homomorphism from (K, L) to (K'', L''), and $(K'', L'') \models \Sigma$. Then, there is a homomorphism from (K, L') to (K'', L''), where L' is the naive chase of (K, L) with Σ.*

PROOF. For every application of a tgd d in Σ of the form $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$, let h be the homomorphism from $\phi(\mathbf{x})$ to K and let h' be the extension of h that was used in that application. Since there is a homomorphism f from (K, L) to (K'', L''), there is a homomorphism $f \circ h : \phi(\mathbf{x}) \rightarrow K''$. Because $(K'', L'') \models \Sigma$, there must be an extension of $f \circ h$ to h'' so that the image of $\psi(\mathbf{x}, \mathbf{y})$ under h'' are facts in L''. We define g on $Dom(K \cup L')$ as follows: for each application of a tgd d on (K, L), we define $g(h'(y)) = h''(y)$ for every $y \in \mathbf{y}$. For every null or constant v in $Dom(K \cup L)$, we define $g(v) = f(v)$.

We now show that g is a homomorphism from (K, L') to (K'', L''). For facts in (K, L') that are also in (K, L), this is true since f is a homomorphism from (K, L) to (K'', L'') and g agrees with f on values from $Dom(K \cup L)$. Let $T(\mathbf{a}, \mathbf{b})$ be a fact in (K, L') that does not occur in (K, L), where $\mathbf{a}$ denotes the values that occur in K and $\mathbf{b}$ denotes the fresh labeled nulls created during the naive chase. Since $T(\mathbf{a}, \mathbf{b})$ is a fact that does not occur in (K, L), it must have been created through an application of some tgd $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$. Let h be the homomorphism from $\phi(\mathbf{x})$ to K used in that application, let h' be the extension of h that was used in that application, and let h'' be the extension of $f \circ h$, as described earlier. There must be an atom $T(\mathbf{x}_0, \mathbf{y}_0)$ in $\psi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}_0$ and $\mathbf{y}_0$ denote subsets of the variables in $\mathbf{x}$ and $\mathbf{y}$ respectively, such that $h'(\mathbf{x}_0) = \mathbf{a}$ and $h'(\mathbf{y}_0) = \mathbf{b}$. Therefore, under the function g, the fact $T(g(\mathbf{a}), g(\mathbf{b}))$ is equal to $T(g(h'(\mathbf{x}_0)), g(h'(\mathbf{y}_0)))$. Because $h'(x) = h(x)$ for every variable x

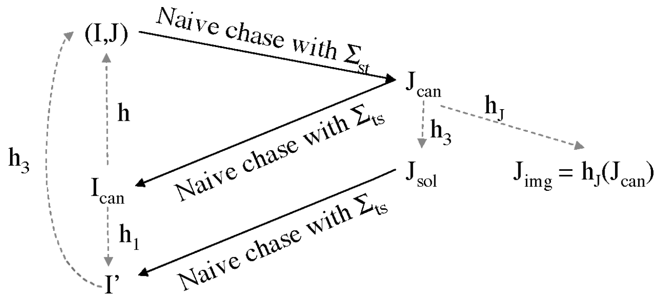
(I,J)
Naive chase with Σ_{st}
J_{can}
h_J
h
h_3
h_3
I_{can}
Naive chase with Σ_{ts}
J_{sol}
$J_{img} = h_J(J_{can})$
h_1
Naive chase with Σ_{ts}
I'

Fig. 3. Illustration of the construction in the proof of Theorem 5.3.

in $\mathbf{x}$ and $g(h'(y)) = h''(y)$ for every variable $y \in \mathbf{y}$, this is in turn equal to $T(g(h(\mathbf{x}_0)), h''(\mathbf{y}_0))$. Since $g(v) = f(v)$ for $v \in Dom(K \cup L)$ and h'' is an extension of $f \circ h$, we arrive at $T(h''(\mathbf{x}_0), h''(\mathbf{y}_0))$. Since $h'' : \psi(\mathbf{x}, \mathbf{y}) \to L''$, it follows that $T(g(\mathbf{a}), g(\mathbf{b}))$ is a fact in L''. □

We are now ready to prove Theorem 5.3. Lemma 5.5 is used in the "if" part of the proof for this theorem and we we will frequently refer to Figure 3 in our proof. Figure 3 is an extension of Figure 2. The instance J_{sol} is a solution for (I, J). For simplicity, we have depicted I' to be the naive chase of J_{sol} (instead of $(J_{sol}, \emptyset)$) with Σ_{ts}. We state Theorem 5.3 again below.

THEOREM 5.3. *Let $\mathcal{P}$ be a PDE setting that satisfies condition* (1) *of the definition of $\mathcal{C}_{tract}$. Then, there exists a solution for (I, J) in $\mathcal{P}$ if and only if there is a homomorphism from I_{can} to I.*

PROOF. ($\Rightarrow$) Let J_{sol} be a solution for (I, J) in $\mathcal{P}$. Let I' be the naive chase of $(J_{sol}, \emptyset)$ with Σ_{ts}. We will show that there is a homomorphism from I_{can} to I by exhibiting two homomorphisms: one from I_{can} to I' (h_1 in Figure 3) and one from I' to I (h_2 in Figure 3). Consequently, a homomorphism from I_{can} to I can be obtained by composing h_1 with h_2.

Since J_{sol} is a solution, we know that $(I, J_{sol}) \models \Sigma_{st}$. Furthermore, J_{sol} contains J. Clearly, there is a homomorphism, which is the identity mapping, from (I, J) to (I, J_{sol}). Hence, by Lemma 5.6, there is a homomorphism from (I, J_{can}) to (I, J_{sol}). In particular, this means that there is a homomorphism from J_{can} to J_{sol}, which is denoted as h_3 in Figure 3.

Since I' is the naive chase of $(J_{sol}, \emptyset)$ with Σ_{ts}, it follows that $(J_{sol}, I') \models \Sigma_{ts}$. We know that there is a homomorphism h_3 from $(J_{can}, \emptyset)$ to (J_{sol}, I'). By applying Lemma 5.6, we obtain a homomorphism from (J_{can}, I_{can}) to (J_{sol}, I'). In particular, this means there is a homomorphism, call it h_1, from I_{can} to I'. Since I' is the naive chase of $(J_{sol}, \emptyset)$ with Σ_{ts} and $(J_{sol}, I) \models \Sigma_{ts}$, there is also a homomorphism, call it h_2, from I' to I.

($\Leftarrow$) Let h be a homomorphism from I_{can} to I. From h, we will construct a function h_J and define J_{img} to be $h_J(J_{can})$. We then show that J_{img} is a solution for (I, J) in $\mathcal{P}$. The function h_J is defined as follows:

—$h_J(x) = h(x)$ if $x \in (Dom(J_{can}) \cap Dom(I_{can}))$;
—$h_J(x) = x$ if $x \in (Dom(J_{can}) - Dom(I_{can}))$.

Observe that, by the definition of J_{img}, h_J is a homomorphism from J_{can} to J_{img}. In order to show that J_{img} is a solution for (I, J) in $\mathcal{P}$, we will show next that $J \subseteq J_{img}$, $(I, J_{img}) \models \Sigma_{st}$ and $(J_{img}, I) \models \Sigma_{ts}$.

Since J_{can} is the naive chase of (I, J) with Σ_{st}, we have that $J \subseteq J_{can}$. We show next that $h_J(J) = J$ and therefore, $J \subseteq J_{img}$. Let c be an element from $Dom(J)$. Since J is an instance without null values, c is a constant. Furthermore, since $J \subseteq J_{can}$, we have that $c \in Dom(J_{can})$. If $c \in Dom(I_{can})$, then $h_J(c) = h(c) = c$, because h maps constants to their identity. If $c \notin Dom(I_{can})$, then $h_J(c) = c$. Therefore it is always the case that $h_J(c) = c$. It follows that $h_J(J) = J$. Since $J \subseteq J_{can}$ and $J_{img} = h_J(J_{can})$, we have that $J \subseteq J_{img}$.

We show next that $(I, J_{img}) \models \Sigma_{st}$. Let $\phi_s(\mathbf{x}_1, \mathbf{x}_2) \rightarrow \exists \mathbf{y} \psi_t(\mathbf{x}_1, \mathbf{y})$ be a tgd in Σ_{st}, where $\mathbf{x}_1$ are the universally quantified variables that occur in ψ_t. Assume that $\mathbf{c}_1, \mathbf{c}_2 \in Dom(I)$ are such that $I \models \phi_s(\mathbf{c}_1, \mathbf{c}_2)$. Note that $\mathbf{c}_1, \mathbf{c}_2$ contain only constants, since I is an instance without null values. Since $(I, J_{can}) \models \Sigma_{st}$, this means $J_{can} \models \psi_t(\mathbf{c}_1, \mathbf{d})$, for some $\mathbf{d} \in Dom(J_{can})$. So $\mathbf{c}_1$ occurs in $Dom(J_{can})$ as well. Since h_J is a homomorphism from J_{can} to J_{img}, we have $J_{img} \models \psi_t(h_J(\mathbf{c}_1), h_J(\mathbf{d}))$. Because $h_J(\mathbf{c}_1) = \mathbf{c}_1$, it follows that $J_{img} \models \psi_t(\mathbf{c}_1, \mathbf{e})$, for some $\mathbf{e} \in Dom(J_{img})$. Hence, $(I, J_{img}) \models \Sigma_{st}$.

Next, we show that $(J_{img}, I) \models \Sigma_{ts}$. Consider a target-to-source tgd in Σ_{ts} of the form $\alpha_t(\mathbf{x}_1, \mathbf{x}_2) \rightarrow \exists \mathbf{y} \beta_s(\mathbf{x}_1, \mathbf{y})$, where $\mathbf{x}_1$ are the universally quantified variables that are used in β_s. Assume that there are some $\mathbf{c}_1$ and $\mathbf{c}_2$ in $Dom(J_{img})$ such that $J_{img} \models \alpha_t(\mathbf{c}_1, \mathbf{c}_2)$. By Lemma 5.5, it follows that there exists $\mathbf{d}_1$ and $\mathbf{d}_2$ in $Dom(J_{can})$ such that $h_J(\mathbf{d}_1) = \mathbf{c}_1$, $h_J(\mathbf{d}_2) = \mathbf{c}_2$ and $J_{can} \models \alpha_t(\mathbf{d}_1, \mathbf{d}_2)$. Since $(J_{can}, I_{can}) \models \Sigma_{ts}$, we have $I_{can} \models \beta_s(\mathbf{d}_1, \mathbf{e})$ for some $\mathbf{e} \in Dom(I_{can})$. This means that $\mathbf{d}_1$ occurs in $Dom(I_{can})$. Under the homomorphism h, we have $I \models \beta_s(h(\mathbf{d}_1), h(\mathbf{e}))$. Since $\mathbf{d}_1$ is also in $Dom(J_{can})$, we have that $\mathbf{d}_1$ occurs in $Dom(J_{can}) \cap Dom(I_{can})$. Therefore, we have $h(\mathbf{d}_1) = h_J(\mathbf{d}_1) = \mathbf{c}_1$ and this gives us $I \models \beta_s(\mathbf{c}_1, h(\mathbf{e}))$. Hence, we conclude that $(J_{img}, I) \models \Sigma_{ts}$.

Since J_{img} contains J, $(I, J_{img}) \models \Sigma_{st}$ and $(J_{img}, I) \models \Sigma_{ts}$, we have that J_{img} is a solution for (I, J) in $\mathcal{P}$, which was to be shown. □

In effect, Theorem 5.3 gives us an algorithm for checking if there exists a solution, and computing a solution if one exists, given the source and target instances (I, J) and a PDE setting $\mathcal{P}$. To check whether a solution exists for I in $\mathcal{P}$, we use the naive chase on $(I, \emptyset)$ with Σ_{st} to obtain J_{can}, and then we use the naive chase on $(J_{can}, \emptyset)$ with Σ_{ts} to obtain I_{can}. Finally, we check whether there exists a homomorphism from I_{can} to I. Indeed, if there is a homomorphism from I_{can} to I, the instance J_{img} that was constructed in the proof of Theorem 5.3 is a solution for (I, J) in $\mathcal{P}$. We therefore arrive at the following corollary.

COROLLARY 5.7. *Let $\mathcal{P}$ be a PDE setting that satisfies condition (1) of the definition of $\mathcal{C}_{tract}$. If there is a homomorphism h from I_{can} to I, then $h_J(J_{can})$ is a solution for (I, J) in $\mathcal{P}$, where h_J is defined as follows:*

—$h_J(x) = h(x)$ *if* $x \in (Dom(J_{can}) \cap Dom(I_{can}))$;
—$h_J(x) = x$ *if* $x \in (Dom(J_{can}) - Dom(I_{can}))$.

A natural question that arises is whether the algorithm proposed earlier for determing whether a solution exists, and computing a solution if one exists, can be performed in polynomial time for a fixed PDE setting $\mathcal{P}$ that satisfies condition (1) of $\mathcal{C}_{tract}$. However, as we have seen with the reduction that illustrates the necessity of condition (2) in the previous section, unless P = NP, this is not true. In the next section, we show how one can determine whether there exists a homomorphism from I_{can} to I in polynomial time if the PDE setting $\mathcal{P}$ also satisfies condition (2) of the definition of $\mathcal{C}_{tract}$.

5.2 Condition 2 and Blocks of Bounded Size

In this section, we prove Theorem 5.4, which states that if every tgd in Σ_{ts} satisfies condition 2 of $\mathcal{C}_{tract}$ then the size of every block of nulls of I_{can} is bounded. The proof of Theorem 5.4 makes use of the following crucial lemma.

LEMMA 5.8. *Let $\mathcal{P}$ be a PDE setting. Let $\mathcal{B}$ be a block of nulls in J_{can} and let $J_{\mathcal{B}}$ denote the set of all tuples in J_{can} that contain at least one null from $\mathcal{B}$. Let $I_{J_{\mathcal{B}}}$ be the naive chase of $(J_{\mathcal{B}}, \emptyset)$ with Σ_{ts}. Then, the number of distinct nulls in $I_{J_{\mathcal{B}}}$ is bounded.*

PROOF. Let the maximum number of existentially quantified variables in a tgd in Σ_{st} be k_1. Similarly, let the maximum number of existentially quantified variables in a tgd in Σ_{ts} be k_1'. Let the maximum number of relational atoms in the right-hand side of a tgd in Σ_{st} be k_2. Note that k_1, k_1', and k_2 are constants that depend only on the PDE setting $\mathcal{P}$. Then the number of nulls in $\mathcal{B}$ is bounded by k_1, while the number of tuples in $J_{\mathcal{B}}$ is bounded by k_2; this is so because in the naive chase each block of nulls is determined by a single application of a tgd. Since the schemas in $\mathcal{P}$ are fixed, the size of $Dom(J_{\mathcal{B}})$ is bounded by some constant k_3 that depends only on $\mathcal{P}$. Next we analyze the maximum number of fresh labeled nulls that can be generated by the naive chase of $(J_{\mathcal{B}}, \emptyset)$ with Σ_{ts}. Let $d : \alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{y} \beta_s(\mathbf{x}, \mathbf{y})$ be a target-to-source tgd in Σ_{ts}. Since the size of $Dom(J_{\mathcal{B}})$ is bounded by k_3, there are at most $k_3^{|\mathbf{x}|}$ homomorphisms from $\alpha_t(\mathbf{x})$ to $J_{\mathcal{B}}$, where $|\mathbf{x}|$ denotes the number of variables in $\mathbf{x}$. In the worst case, each of these homomorphisms generates k_1' nulls. Therefore, there are at most $k_1' * k_3^{|\mathbf{x}|}$ nulls that are generated in all possible applications of d. In turn, this means there can be at most $k_1' * k_3^{k_4} * |\Sigma_{ts}|$ nulls that are generated during the naive chase of $J_{\mathcal{B}}$ with Σ_{ts}, where k_4 denotes the maximum number of universally quantified variables in a tgd in Σ_{ts} and $|\Sigma_{ts}|$ denotes the number of tgds in Σ_{ts}. Since the nulls of $\mathcal{B}$ may be copied to $I_{J_{\mathcal{B}}}$ during the naive chase, there can be at most $k_1 + k_1' * k_3^{k_4} * |\Sigma_{ts}|$ distinct nulls in $I_{J_{\mathcal{B}}}$, which is a constant that depends only on $\mathcal{P}$. □

We are now ready to prove Theorem 5.4, which we state again below.

THEOREM 5.4. *Let $\mathcal{P}$ be a PDE setting that satisfies condition (2) of the definition of $\mathcal{C}_{tract}$. Then the size of every block of nulls of I_{can} is bounded.*

PROOF. We prove the theorem by analyzing the blocks of nulls of I_{can}. Let $\mathcal{B}$ be a block of nulls from I_{can}.

Case 1. *$\mathcal{B}$ does not contain nulls from J_{can}.* In this case, the nulls of $\mathcal{B}$ are freshly generated during the naive chase of $(J_{can}, \emptyset)$ with Σ_{ts}. This means there is exactly one application of a tgd in Σ_{ts} that generates these nulls since different applications of tgds cannot produce a single block of nulls without nulls from J_{can}. Therefore, the size of $\mathcal{B}$ is bounded by the maximum number of existentially quantified variables in a tgd in Σ_{ts}. This is a constant that depends only on $\mathcal{P}$.

Case 2. *$\mathcal{B}$ contains at least one null from J_{can}.* Let n be the null in $\mathcal{B}$ that also occurs in J_{can}. Let $\mathcal{B}'$ be the block of nulls in J_{can} that contains n, and let $J_{\mathcal{B}'}$ be the set of all tuples in J_{can} that contain at least one null from $\mathcal{B}'$. Let $I_{J_{\mathcal{B}'}}$ be the set of tuples that are added to I_{can}, as a consequence of applying tgds on $(J_{\mathcal{B}'}, \emptyset)$, during the naive chase of $(J_{can}, \emptyset)$ with Σ_{ts}. It is easy to see that $I_{J_{\mathcal{B}'}}$ is identical to the naive chase of $(J_{\mathcal{B}'}, \emptyset)$ with Σ_{ts}, up to the renaming of nulls. Henceforth, we refer to $I_{J_{\mathcal{B}'}}$ as the *naive chase* of $(J_{\mathcal{B}'}, \emptyset)$ with Σ_{ts}. Let $p_0, p_1, \ldots, p_k$ be a path in the graph of nulls of $\mathcal{B}$, where $p_0 = n$. We show by induction on the number of nodes in the path that every node in the path occurs in $\mathcal{B}' \cup Var(I_{J_{\mathcal{B}'}})$.

If the number of nodes in the path is 1, the path consists of only n. We know that n occurs in $\mathcal{B}'$. Hence, n occurs in $\mathcal{B}' \cup Var(I_{J_{\mathcal{B}'}})$. Now assume that p_i is a null in $\mathcal{B}' \cup Var(I_{J_{\mathcal{B}'}})$ for every $i < j$, where $1 < j \leq k$. Let $i = j - 1$. We show next that p_{i+1} is also a null in $\mathcal{B}' \cup Var(I_{J_{\mathcal{B}'}})$. Since p_i and p_{i+1} are adjacent in the graph of nulls of $\mathcal{B}$, there must be a fact $T(\mathbf{c})$ in I_{can} that contains both p_i and p_{i+1}. This fact must have been placed in I_{can} by an application of a tgd $d : \alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{y} \beta_s(\mathbf{x}, \mathbf{y})$ in Σ_{ts} with some homomorphism h. Let h' be the extension of h that was used in that application. Hence, the relational atom $T(\mathbf{z})$ occurs in $\beta_s(\mathbf{x}, \mathbf{y})$, where $\mathbf{z} \subseteq \mathbf{x} \cup \mathbf{y}$ and $T(h'(\mathbf{z})) = T(\mathbf{c})$. Let u and v be two variables in $\mathbf{z}$ where $h'(u) = p_i$ and $h'(v) = p_{i+1}$. Note that u and v are marked variables. We prove the induction by analyzing the following cases, depending on whether p_i is a null in $\mathcal{B}'$ or $Var(I_{J_{\mathcal{B}'}})$.

—*Case* (a). *p_i is a null in $\mathcal{B}'$.* This means that u is a marked variable that occurs in $\alpha_t(\mathbf{x})$. There are two cases to consider, depending on whether p_{i+1} is a null that does not occur in J_{can} (i.e., p_{i+1} is a null that is freshly generated during the naive chase of $(J_{can}, \emptyset)$ with Σ_{ts}) or p_{i+1} is a null from J_{can}.

  —*Case* (i). *p_{i+1} is a null not from J_{can}.* This means that v is a variable in $\mathbf{y}$. In turn, this means that d cannot satisfy condition (2.2) of $\mathcal{C}_{tract}$ because u occurs in $\alpha_t(\mathbf{x})$ but v does not. So d must satisfy condition (2.1) of $\mathcal{C}_{tract}$. Thus α_t consists of exactly one literal. Since p_i is a null in $\mathcal{B}'$, it follows that the fact represented by $h(\alpha_t(\mathbf{x}))$ occurs in $J_{\mathcal{B}'}$ and, therefore, p_{i+1} occurs in $Var(I_{J_{\mathcal{B}'}})$.

  —*Case* (ii). *p_{i+1} is a null from J_{can}.* This means that v is a variable that occurs in $\alpha_t(\mathbf{x})$. If d satisfies condition (2.1) of $\mathcal{C}_{tract}$, then we know that p_i and p_{i+1} occur together in some fact in J_{can}. Since p_i occurs in $\mathcal{B}'$, this means that p_{i+1} is also a null in $\mathcal{B}'$. Otherwise, it must be that d satisfies condition (2.2)(a) of $\mathcal{C}_{tract}$. Note that d cannot satisfy condition (2.2)(b) of $\mathcal{C}_{tract}$ because u occurs in $\alpha_t(\mathbf{x})$. With condition (2.2)(a), we know that p_i and p_{i+1} must occur together in some fact in J_{can}. Since p_i occurs in $\mathcal{B}'$, we have that p_{i+1} must also occur in $\mathcal{B}'$.

—*Case* (b). p_i *is a null in* $Var(I_{J_{\mathcal{B}'}})$. Note that the nulls in $Var(I_{J_{\mathcal{B}'}})$ are either nulls from $\mathcal{B}'$ or nulls that are newly generated during the naive chase of $(J_{\mathcal{B}'}, \emptyset)$ with Σ_{ts}. If p_i is a null in $\mathcal{B}'$, then by a similar argument shown in case (a), we have that p_{i+1} occurs in $\mathcal{B}' \cup Var(I_{J_{\mathcal{B}'}})$. If p_i is a newly generated null, then p_i is generated by exactly the application of d with h. Every other application during the naive chase of $(J_{can}, \emptyset)$ with Σ_{ts} will generate fresh nulls that are different from p_i. Hence, every fact in $h(\alpha_t(\mathbf{x}))$ must occur in $J_{\mathcal{B}'}$ in order for p_i to be a null in $Var(I_{J_{\mathcal{B}'}})$. Therefore, all facts in $h'(\beta_s(\mathbf{x}, \mathbf{y}))$ occur in $I_{J_{\mathcal{B}'}}$. In particular, the fact $T(\mathbf{c})$ occurs in $I_{J_{\mathcal{B}'}}$ and so p_{i+1} occurs in $Var(I_{J_{\mathcal{B}'}})$.

Since the nodes in every path reachable from n in the graph of nulls of $\mathcal{B}$ occur in $\mathcal{B}' \cup Var(I_{J_{\mathcal{B}'}})$ and since every null in $\mathcal{B}$ is reachable from n by some path, it follows that $\mathcal{B} \subseteq \mathcal{B}' \cup Var(I_{J_{\mathcal{B}'}})$. The size of $\mathcal{B}'$ is bounded by the maximum number of existentially quantified variables in a tgd in Σ_{st}; moreover, by Lemma 5.8, the size of $Var(I_{J_{\mathcal{B}'}})$ is bounded. It follows that the size of $\mathcal{B}$ is bounded. □

In the next section, we describe a polynomial time algorithm using the results we have developed so far: the algorithm determines whether a solution exists for (I, J) in a fixed peer data exchange setting $\mathcal{P}$ that satisfies both conditions of $\mathcal{C}_{tract}$, and computes a solution if a solution exists.

5.3 Proof of Theorem 4.3

Using Theorem 5.3, one can determine whether a solution exists for (I,J) in $\mathcal{P}$ by checking whether there is a homomorphism from I_{can} to I. We show next in Proposition 5.9 how to decompose the task of checking whether a homomorphism exists from I_{can} to I. In addition, Corollary 5.7 gives us a procedure that makes use of the homomorphism from I_{can} to I for computing a solution, if a solution exists. Putting these results together, we now have an algorithm, depicted in Figure 4, for determining whether a solution exists for an instance (I, J) in a fixed peer data exchange setting $\mathcal{P}$ that satisfies both conditions of $\mathcal{C}_{tract}$. This algorithm also computes a solution, if a solution exists.

PROPOSITION 5.9. *There is a homomorphism from K_1 to K_2 if and only if* (1) *every null-free fact of K_1 is in K_2 and* (2) *for every block of nulls $\mathcal{B}$ of K_1 there exists a homomorphism from $K_{\mathcal{B}}$ to K_2, where $K_{\mathcal{B}}$ is the set of all tuples in K_1 that contain at least one null from $\mathcal{B}$.*

PROOF. (⇒) Suppose there is a homomorphism h from K_1 to K_2. Let $R(\mathbf{t})$ be a null-free fact of K_1. Since h is a constant-preserving homomorphism, it follows that $R(\mathbf{t})$ is also fact in K_2. For every block of nulls $\mathcal{B}$ of K_1, define a mapping $h_{\mathcal{B}}$ from $Dom(K_{\mathcal{B}})$ to $Dom(I)$ as follows:

$$h_{\mathcal{B}}(x) = h(x) \text{ for every } x \in Dom(K_{\mathcal{B}}).$$

Let $R(\mathbf{t})$ be a fact in $K_{\mathcal{B}}$. Since $R(h_{\mathcal{B}}(\mathbf{t})) = R(h(\mathbf{t}))$ and h is a homomorphism from K_1 to K_2, we have that $R(h(\mathbf{t}))$ is a fact in K_2. Therefore, $h_{\mathcal{B}}$ is a homomorphism from $K_{\mathcal{B}}$ to K_2, which was to be shown.

```
Algorithm ComputeASolution_P(I, J) : T
 Let $J_{can}$ be the naive chase of $(I, J)$ with $\Sigma_{st}$.
 Let $I_{can}$ be the naive chase of $(J_{can}, \emptyset)$ with $\Sigma_{ts}$.
 Let $h = \emptyset$.
  For each block of nulls $\mathcal{B}$ of $I_{can}$
       Let $I_{\mathcal{B}}$ denote the set of all tuples in $I_{can}$ containing at least a null from $\mathcal{B}$.
       If there is no homomorphism from $I_{\mathcal{B}}$ to $I$
          Return "No solution"
       Else
          Let $h'$ denote a homomorphism from $I_{\mathcal{B}}$ to $I_{can}$.
          Extend $h$ with $h'$: Define $h = h \cup h'$.
       End If
  End For
  Let $I_c$ denote the set of all null-free tuples in $I_{can}$.
  If there is a fact in $I_c$ that does not occur in $I$
       Return "No solution"
  EndIf
  Define $h_J(x) = h(x)$ if $x \in Dom(J_{can}) \cap Dom(I_{can})$, and $h_J(x) = x$ otherwise.
  Return $h_J(J_{can})$.
```

Fig. 4. Algorithm ComputeASolution$_{\mathcal{P}}$.

($\Leftarrow$) Suppose every null-free fact of K_1 occurs in K_2 and there are n blocks of nulls in K_1. Let $\mathcal{B}_i$ denote the ith block of nulls of K_1 and let h_i denote the homomorphism from $K_{\mathcal{B}_i}$ to K_2. Let h be defined as follows:

$$h(x) = \begin{cases} h_i(x) & \text{if } x \text{ is in } Dom(K_{\mathcal{B}_i}),\ 1 \leq i \leq n, \\ x & \text{if } x \text{ is a constant in } K_1. \end{cases}$$

Since a null in K_1 occurs in exactly one of the blocks of nulls and $h_j(c) = c$ for every constant c that occurs in $K_{\mathcal{B}_j}$ and for every $j \leq n$, we have that h is a function.

We show next that h is a homomorphism from K_1 to K_2. If $R(\mathbf{t})$ is a null-free fact of K_1, then $R(h(\mathbf{t})) = R(\mathbf{t})$ and by assumption, $R(\mathbf{t})$ occurs in K_2. Suppose $R(\mathbf{t})$ is a fact K_1 that contains at least one null. Then $R(\mathbf{t})$ belongs to exactly one $K_{\mathcal{B}_i}$, for some $1 \leq i \leq n$. Since h_i is a homomorphism from $K_{\mathcal{B}_i}$ to K_2 and $R(h(\mathbf{t})) = R(h_i(\mathbf{t}))$, we conclude that $R(h(\mathbf{t}))$ is a fact in K_2. Therefore, h is a homomorphism from K_1 to K_2, which was to be shown. □

We are now ready to prove Theorem 4.3, which we state again below.

THEOREM 4.3. *Let $\mathcal{P}$ be a PDE setting in $\mathcal{C}_{tract}$. Then, the existence-of-solutions problem SOL($\mathcal{P}$) for $\mathcal{P}$ is solvable in polynomial time. Moreover, if a solution exists, then a solution can be computed in polynomial time.*

PROOF. We will show that algorithm ComputeASolution$_{\mathcal{P}}$ of Figure 4 correctly determines whether a solution exists for (I, J) and computes a solution if it exists. We will show that this algorithm runs in polynomial time in the size of (I, J).

The correctness of ComputeASolution$_{\mathcal{P}}$ follows from Theorem 5.3, Proposition 5.9, and Corollary 5.7. Assume that, for every block of nulls $\mathcal{B}$ of I_{can}, there is a homomorphism from $I_{\mathcal{B}}$ to I, where $I_{\mathcal{B}}$ is the set of all tuples in I_{can} that each contains at least a null from $\mathcal{B}$. Furthermore, every null-free tuple of I_{can} occurs in I. Then, by Proposition 5.9, there is a homomorphism from I_{can} to I. Hence, by Theorem 5.3, there is a solution for (I, J) in $\mathcal{P}$. If there is no homomorphism from $I_{\mathcal{B}}$ of to I for some $\mathcal{B}$ or some null-free tuple of I_{can} does not occur in I, then, by Proposition 5.9, there is no homomorphism from I_{can} to I. Hence, by Theorem 5.3, there is no solution for (I, J) in I. The extension of h with h' is well-defined at each iteration, since $h'(c) = c$ for every constant c from I_{can} and every block of nulls of I_{can} are pairwise disjoint. From Corollary 5.7, it follows that $h_J(J_{can})$ is a solution as defined in the algorithm.

Next, we show that $\texttt{ComputeASolution}_{\mathcal{P}}$ runs in polynomial time in the size of (I, J) by providing an upper bound on the running time of the algorithm. We first show that the size of J_{can} is polynomial in the size of (I, J). Let $\phi_s(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi_t(\mathbf{x}, \mathbf{y})$ be a source-to-target tgd in Σ_{st}. Let the number of variables in $\mathbf{x}$ be k_1. Then there are at most $|Dom(I)|^{k_1}$ homomorphisms from $\phi_s(\mathbf{x})$ to I. Since each application of a tgd adds a fixed number of facts to the target, it follows that all applications of the tgd adds a polynomial number of facts to the target. Since there is also a fixed number of tgds in Σ_{st}, the size of J_{can} is a polynomial in the size of (I, J). Let this polynomial be $p(n)$, where n is the size of (I, J). By a similar argument, there is also a polynomial in the size of J_{can} that bounds the size of I_{can}. Let this polynomial be $q(m)$ where m denotes the size of J_{can}. Hence, the size of I_{can} is at most $q(p(n))$. It follows that the number of blocks of nulls in I_{can} is at most $q(p(n))$. From Theorem 5.4, we know that every block of nulls $\mathcal{B}$ of I_{can} is bounded. Hence, checking whether there exists a homomorphism from $I_{\mathcal{B}}$ to I can be performed in polynomial time: First, we define the following mapping f. For every null N of $I_{\mathcal{B}}$, we define $f(N)$ to be a value from I. For every constant c of $I_{\mathcal{B}}$, we define $f(c) = c$. If $f(I_{\mathcal{B}})$ is a subset of I, we conclude that there is a homomorphism from $I_{\mathcal{B}}$ to I. Otherwise, we repeat the process with another definition for f. If there is no homomorphism from $I_{\mathcal{B}}$ to I for every f defined in this way, we conclude that there is no homomorphism from $I_{\mathcal{B}}$ to I. Since $|Dom(I)| \leq n$, there are at most $n^{|\mathcal{B}|}$ possible definitions for f. Computing $f(I_{\mathcal{B}})$ takes at most $q(p(n))$ time. Checking whether $f(I_{\mathcal{B}})$ is a subset of I takes at most $q(p(n)) * n$ time. Hence, it takes $n^{|\mathcal{B}|} * (q(p(n)) + q(p(n)) * n)$ time to check whether there is a homomorphism from $I_{\mathcal{B}}$ to I. Let this polynomial be $r(n)$. Extending h with h' takes at most $|Dom(I_{\mathcal{B}})|$ time. So each iteration takes at most $r(n) + q(p(n))$ time.

Since the algorithm iterates through the blocks of nulls of I_{can}, there are at most $q(p(n))$ iterations. Therefore, the for-loop of the algorithm takes at most $q(p(n)) * (r(n) + q(p(n)))$ time, which is a polynomial in the size of (I, J). Also, it takes $q(p(n)) * n$ time to check whether every null-free tuple of I_{can} occurs in I. Finally, to construct $h_J(J_{can})$, we scan each value v of J_{can} and apply h on v if $v \in Dom(J_{can}) \cap Dom(I_{can})$. Since I_{can} and J_{can} are both polynomial in the size of (I, J), it follows that $h_J(J_{can})$ can be constructed in polynomial time. □

6. QUERY ANSWERING

As seen in Theorem 3.8, there are PDE settings and conjunctive queries such that testing for the existence of solutions is a NP-complete problem and computing the certain answers of these queries is a coNP-complete problem. For

PDE settings in $\mathcal{C}_{tract}$, however, we showed that testing for the existence of solutions is solvable in polynomial time. It is natural to ask whether, for PDE settings in $\mathcal{C}_{tract}$, the certain answers of conjunctive queries are polynomial-time computable. In this section, we show that, unless P = NP, this is not true. Specifically, we show that there is a PDE setting $\mathcal{P}$ in $\mathcal{C}_{tract}$ and a conjunctive query q such that computing the certain answers of q in $\mathcal{P}$ is a coNP-hard problem. After this, we identify syntactic conditions between PDE settings and conjunctive queries that guarantee the tractability of computing the certain answers for PDE settings in $\mathcal{C}_{tract}$. It turns out that these conditions are satisfied by every conjunctive query in PDE settings in which the source-to-target tgds are full tgds. Consequently, there is a polynomial-time algorithm for computing the certain answers of arbitrary (but fixed) conjunctive queries in PDE settings in which the source-to-target tgds are full tgds.

THEOREM 6.1. *There exists a PDE setting $\mathcal{P}$ in $\mathcal{C}_{tract}$ and a Boolean conjunctive query q such that computing the certain answers of q in $\mathcal{P}$ is a* coNP-*complete problem.*

PROOF. Let $\mathcal{P}$ be the following peer data exchange setting. The source schema $\mathbf{S}$ consists of one binary relation E and one unary relation H, and the target schema $\mathbf{T}$ consists of two binary relations C and F. There are no target dependencies, that is, $\Sigma_t = \emptyset$. Finally, the constraints between $\mathbf{S}$ and $\mathbf{T}$ are as follows:

$$\begin{aligned}\Sigma_{st}: \ & E(x, y) \rightarrow \exists v C(x, v)\\ & E(x, y) \rightarrow F(x, y)\\ \Sigma_{ts}: \ & C(x, v) \rightarrow H(v)\end{aligned}$$

Clearly, $\mathcal{P}$ is a PDE setting in $\mathcal{C}_{tract}$. In fact, Σ_{ts} is a LAV setting with no repeated variables in the left-hand side. Let q be the conjunctive query

$$\exists x \exists y \exists v (C(x, v) \wedge C(y, v) \wedge F(x, y)).$$

From Theorem 3.7, we know that the problem of computing the certain answers of q in $\mathcal{P}$ is in coNP. We establish the coNP-hardness of this problem via a reduction from 3-COLORABILITY. Given a graph $G = (V, E)$, consider the source instance $I = (E, H)$, where $H = \{r, g, b\}$ is a set of three colors, and the target instance $\emptyset$. We now claim that G is 3-colorable if and only if $\mathbf{certain}(q, (I, \emptyset)) =$ `false`.

If G is 3-colorable, we can construct a solution J_{sol} for $(I, \emptyset)$ in $\mathcal{P}$ as follows. Using a fixed 3-coloring of G, we form C by assigning each node of G its color in that 3-coloring; we also take $F = E$. By construction, there are no two F-adjacent nodes having the same color. Thus, $q(J_{sol}) =$ `false`, which implies that $\mathbf{certain}(q, (I, \emptyset)) =$ `false`.

Conversely, assume that $\mathbf{certain}(q, (I, \emptyset)) =$ `false`. Hence, there is some solution J_{sol} for $(I, \emptyset)$ in $\mathcal{P}$ such that $q(J_{sol}) =$ `false`. For each node of G, pick a color from $H = \{r, g, b\}$ using the first dependency in Σ_{st} and the dependency in Σ_{ts}. Since $q(J_{sol}) =$ `false`, no two F-adjacent nodes of G get the same color. By the second dependency in Σ_{st}, we have that $E \subseteq F$. Consequently, no two E-adjacent nodes get the same color, which means that G is 3-colorable. □

We now consider syntactic conditions between PDE settings and conjunctive queries, and use them to introduce a restricted class of conjunctive queries relative to a given PDE setting.

Definition 6.2. Let $\mathcal{P}$ be a PDE setting with no target constraints and let q be a conjunctive query over the target schema of $\mathcal{P}$. We say that a variable z is *marked in* q if z appears at a position of a literal R of q that is a marked position for R in $\mathcal{P}$.

Definition 6.3. Let $\mathcal{P}$ be a PDE setting with no target constraints and let q be a conjunctive query over the target schema of $\mathcal{P}$. We say that $q \in \mathcal{Q}_{tract}(\mathcal{P})$ if every free variable of q is not a marked variable and every marked variable of q appears exactly once in q.

As an example, consider the PDE setting $\mathcal{P}$ and the conjunctive query q in the proof of Theorem 6.1. Clearly, $q \notin \mathcal{Q}_{tract}(\mathcal{P})$, since the marked variable v appears twice in q. In contrast, the conjunctive query

$$\exists x \exists y \exists z \exists v (C(x, v) \wedge F(x, y) \wedge F(y, z) \wedge F(z, x))$$

is in $\mathcal{Q}_{tract}(\mathcal{P})$, since there are no free variables and v occurs only once in the above query.

Note that if $\mathcal{P}$ is a PDE setting with no target constraints and such that Σ_{st} consists of full source-to-target tgds, then *every* conjunctive query over the target schema of $\mathcal{P}$ is in $\mathcal{Q}_{tract}(\mathcal{P})$, since in this case no position is marked.

THEOREM 6.4. *If $\mathcal{P}$ is a PDE setting in $\mathcal{C}_{tract}$ and q is a conjunctive query in $\mathcal{Q}_{tract}(\mathcal{P})$, then there is a polynomial-time algorithm for computing the certain answers of q in $\mathcal{P}$.*

PROOF. We will show that the problem of computing the certain answers of q in $\mathcal{P}$ is reducible to the existence-of-solutions problem for a PDE setting $\mathcal{P}'$ in $\mathcal{C}_{tract}$; the latter problem is tractable by Theorem 4.3.

Assume that $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ and that q is a k-ary conjunctive query of the form $\exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ are the free variables of q and $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over $\mathbf{T}$. Here we assume that $k \geq 1$; Boolean queries can be handled using a similar and, in fact, simpler argument. We construct another PDE setting $\mathcal{P}' = (\mathbf{S}', \mathbf{T}', \Sigma'_{st}, \Sigma'_{ts}, \emptyset)$ as follows:

—$\mathbf{S}' = \mathbf{S} \cup \{S_1, S_2\}$, where S_1 is a new k-ary relation symbol and S_2 is a new unary relation symbol.
—$\mathbf{T}' = \mathbf{T} \cup \{T_1\}$, where T_1 is a new k-ary relation symbol.
—$\Sigma'_{st} = \Sigma_{st} \cup \{S_1(\mathbf{x}) \rightarrow T_1(\mathbf{x})\}$.
—$\Sigma'_{ts} = \Sigma_{ts} \cup \{T_1(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y}) \rightarrow \exists z S_2(z)\}$.

Since the query q is in $\mathcal{Q}_{tract}(\mathcal{P})$, every free variable in q is not a marked variable and no marked variable in q appears more than once in $\psi(\mathbf{x}, \mathbf{y})$. It follows that the target-to-source tgd $T_1(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y}) \rightarrow \exists z S_2(z)$ satisfies condition (1) in the definition of $\mathcal{C}_{tract}$; furthermore, it is clear that it satisfies condition (2.2) in the definition of $\mathcal{C}_{tract}$ as well. From this fact and the hypothesis that $\mathcal{P}$ is in $\mathcal{C}_{tract}$, it follows that $\mathcal{P}'$ is also in $\mathcal{C}_{tract}$.

Suppose we are given a source instance I, a target instance J, and a k-tuple $\mathbf{t}$ of constants from I. Form the source instance $I' = I \cup \{S_1(\mathbf{t})\}$ and the target instance $J' = J$ (that is, $S_2 = \emptyset$ in I' and $T_1 = \emptyset$ in J'). We claim that $\mathbf{t} \in$ **certain**$(q, (I, J))$ if and only if there is no solution for (I', J') in $\mathcal{P}'$.

Assume that $\mathbf{t} \in$ **certain**$(q, (I, J))$. Toward a contradiction, assume also that there is an instance J'_{sol} over $\mathbf{T}'$ such that J'_{sol} is a solution for (I', J') in $\mathcal{P}'$. Let J_{sol} be the restriction of J'_{sol} to $\mathbf{T}$, that is, J_{sol} is obtained from J'_{sol} by removing the interpretation of the relation symbol T_1. Since J'_{sol} is a solution for (I', J') in $\mathcal{P}'$ and since S_1 and T_1 are not among the relation symbols of $\mathbf{S}$ and $\mathbf{T}$, we have that J_{sol} is a solution for (I, J) in $\mathcal{P}$. Since $\mathbf{t} \in$ **certain**$(q, (I, J))$, it follows that $\mathbf{t} \in q(J_{sol})$, which means that there is a tuple $\mathbf{b}$ of elements in J_{sol} such that $J_{sol} \models \psi(\mathbf{t}, \mathbf{b})$. From this, it follows that $J'_{sol} \models T_1(\mathbf{t}) \wedge \psi(\mathbf{t}, \mathbf{b})$, which, in turn, implies that $I' \models \exists z S_2(z)$, since $(J'_{sol}, I') \models \forall \mathbf{x} \forall \mathbf{y}(T_1(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y}) \rightarrow \exists z S_2(z))$. By construction, however, $S_2 = \emptyset$ in I'; thus we have arrived at a contradiction.

Assume that there is no solution for (I', J') in $\mathcal{P}'$. Toward a contradiction, assume also that $\mathbf{t} \notin$ **certain**$(q, (I, J))$. It follows that there is a solution J_{sol} for (I, J) in $\mathcal{P}$ such that $\mathbf{t} \notin q(J_{sol})$, which means that $J_{sol} \models \forall \mathbf{y} \neg \psi(\mathbf{t}, \mathbf{y})$. Let $J'_{sol} = J_{sol} \cup \{T_1(\mathbf{t})\}$. Clearly, $J' \subseteq J'_{sol}$. Furthermore, $(J'_{sol}, I') \models \forall \mathbf{x} \forall \mathbf{y}(T_1(\mathbf{x}) \wedge \psi(\mathbf{x}, \mathbf{y}) \rightarrow \exists z S_2(z))$ because $\mathbf{t}$ is the only tuple in T_1 in J'_{sol} and $J_{sol} \models \forall \mathbf{y} \neg \psi(\mathbf{t}, \mathbf{y})$. Since J_{sol} is a solution for (I, J) in $\mathcal{P}$ and since S_1 and T_1 do not appear in any tgd of Σ_{st} or of Σ_{ts}, we have that $(I', J'_{sol}) \models \Sigma_{st}$ and $(J'_{sol}, I') \models \Sigma_{ts}$. It follows that J'_{sol} is a solution for (I', J') in $\mathcal{P}'$, which is a contradiction. □

As an immediate consequence of Theorem 6.4 and the preceding observation about full tgds, we obtain the following result.

COROLLARY 6.5. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a PDE setting such that Σ_{st} is a set of full source-to-target tgds and $\Sigma_t = \emptyset$. For every target conjunctive query q, there is a polynomial-time algorithm for computing the certain answers of q in $\mathcal{P}$.*

7. UNIVERSAL SOLUTIONS AND UNIVERSAL BASES IN PEER DATA EXCHANGE

In the preceding section, we saw that computing the certain answers of fixed conjunctive queries in peer data exchange settings can be a coNP-complete problem, even when there are no target dependencies and the existence-of-solutions problem is solvable in polynomial time. This state of affairs for peer data exchange contrasts sharply with data exchange, where, as shown in Fagin et al. [2005a], the certain answers of conjunctive queries are computable in polynomial time for data exchange settings $\mathcal{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ in which Σ_{st} is a set of source-to-target tgds and Σ_t is the union of a set of target egds with a weakly acyclic set of target tgds. This result hinges on the following three facts:

(1) If a solution exists for a given source instance I, then a *universal solution* for I exists, that is, a solution J_u for I such that, for every solution J' for I, there is a homomorphism from J_u to J'.
(2) There is a polynomial-time algorithm based on the chase procedure, such that, given a source instance I, the algorithm determines whether a solution

for I exists and, if a solution exists, it returns a universal solution J_u for I.

(3) Every universal solution is *adequate* for computing the certain answers of target conjunctive queries. This means that if q is a target conjunctive query, I is a source instance, and J_u is a universal solution for I, then $\textbf{certain}(q, I) = q(J_u)_\downarrow$, where $q(J_u)_\downarrow$ is the subset of $q(J_u)$ consisting of all null-free tuples in $q(J_u)$.

Note that if $\mathcal{S}$ is a data exchange setting with no target dependencies ($\Sigma_t = \emptyset$), then for every instance I, a solution exists. It follows that in this case, for every source instance I, there is a polynomial-time computable universal solution J_u for I; moreover, J_u is adequate for computing the certain answers of conjunctive queries. It should also be noted that results in Calì et al. [2003, 2004], imply that the certain answers of fixed target conjunctive queries are computable in polynomial time for data exchange settings in which the target constraints consist of key constraints and of restricted types of inclusion dependencies (such as foreign key constraints). In such settings, the target tgds need not form a weakly acyclic set, yet the certain answers can be computed by using a finite part of a potentially infinite target obtained via the chase procedure.

To gain a deeper insight into the differences between data exchange and peer data exchange, we now introduce the concept of a *universal solution* for peer data exchange.

Definition 7.1. Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a peer data exchange setting, I a source instance, and J a target instance. We say that a target instance J_u is a *universal solution for* (I, J) if J_u is a solution for (I, J) and for every solution J' for (I, J), there is a homomorphism from J_u to J'.

It is clear that a universal solution for a source instance I in a data exchange setting $\mathcal{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$ is a universal solution for $(I, \emptyset)$ in the peer data exchange setting $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \emptyset, \Sigma_t)$ (i.e., $\Sigma_{ts} = \emptyset$).

The next example shows that there is a peer data exchange setting in the class $\mathcal{C}_{tract}$ and a source instance for which solutions exist, but universal solutions do not. Moreover, no solution is adequate for computing the certain answers of target conjunctive queries.

Example 2. Let $\mathbf{S}$ be a schema consisting of two unary relation symbols P and R, and let $\mathbf{T}$ be a schema consisting of a binary relation symbol T. Consider the following PDE setting with $\mathbf{S}$ as its source schema and $\mathbf{T}$ as its target schema:

$$\begin{aligned} \Sigma_{st}:&\ P(x) \rightarrow \exists z T(x, z) \\ \Sigma_{ts}:&\ T(x, z) \rightarrow P(x) \\ &\ T(x, z) \rightarrow R(z) \end{aligned}$$

Clearly this PDE setting is in $\mathcal{C}_{tract}$. If $I = \{P(a), R(b_1), \ldots, R(b_n)\}$ and $J = \emptyset$, then a target instance J' is a solution for (I, J) if and only if $J' = \{T(a, b_k) : k \in K\}$ for some nonempty subset K of $\{1, \ldots, n\}$. In particular, there are exponentially many solutions for (I, J). It is easy to see, however, that none of these solutions is universal, since homomorphisms must map each constant from I

to itself. For instance, the solution $\{T(a, b_1)\}$ is not universal because there is no homomorphism from $\{T(a, b_1)\}$ to the solution $\{T(a, b_2)\}$.

Let q be the target conjunctive query $T(x, y)$. Clearly, $\mathbf{certain}(q, (I, J)) = \emptyset$, while at the same time if J' is a solution for (I, J), then $q(J')_{\downarrow} \neq \emptyset$. Thus no solution for (I, J) is adequate for computing the certain answers of q.

Since universal solutions may not exist even for PDE settings in $\mathcal{C}_{tract}$, we introduce the concept of a *universal basis* as a relaxation of the concept of a universal solution. We note that Nash et al. [2006] independently introduced the concept of a *universal set solution* for data exchange, which is the same concept, but in the context of data exchange with arbitrary embedded dependencies (not just source-to-target dependencies and target dependencies).

Definition 7.2. Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a peer data exchange setting. If I is a source instance and J is a target instance, then a *universal basis for* (I, J) *in* $\mathcal{P}$ is a finite, nonempty set U of solutions for (I, J) such that, for every solution J_{sol} for (I, J), there is a $J_u \in U$ such that a homomorphism from J_u to J_{sol} exists.

It is clear that every finite set of solutions containing a universal basis is also a universal basis. In the special case of data exchange, a singleton set with a universal solution for a source instance I as its member forms a universal basis for I. In the preceding Example 2, the set $U_0 = \{\{T(a, b_i)\} : 1 \leq i \leq n\}$ forms a universal basis for (I, J); however, no proper subset of U is a universal basis for (I, J). In the same example, since there are finitely many solutions for (I, J), the set $\mathrm{SOL}(I, J)$ of all solutions for (I, J) is a universal basis for (I, J). Thus, in this example, the universal bases for (I, J) are precisely the sets U of solutions for (I, J) such that $U_0 \subseteq U \subseteq \mathrm{SOL}(I, J)$.

We also introduce the concept of an *adequate set* for computing the certain answers of queries in peer data exchange settings.

Definition 7.3. Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a peer data exchange setting, I a source instance, J a target instance, and q a target query. We say that a set U of solutions for (I, J) is *adequate for computing the certain answers of* q in $\mathcal{P}$ if $\mathbf{certain}(q, (I, J)) = \bigcap_{J' \in U} q(J')_{\downarrow}$.

It turns out that there is a tight connection between universal bases and adequate sets.

PROPOSITION 7.4. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \Sigma_t)$ be a peer data exchange setting, I a source instance, J a target instance, and U a finite, nonempty set of solutions for (I, J). Then the following statements are equivalent:*

(1) *U is a universal basis for (I, J).*
(2) *U is adequate for computing the certain answers of every finite union of target conjunctive queries.*

PROOF. Assume first that U is a universal basis for (I, J). Let q be a finite union $(q_1 \vee \cdots \vee q_m)$ of target conjunctive queries $q_1, \ldots, q_m$. We have to show that if $\mathbf{t}$ is a tuple of constants from I, then $\mathbf{t} \in \mathbf{certain}(q, (I, J))$ if and only if $\mathbf{t} \in \bigcap_{J' \in U} q(J')_{\downarrow}$. Since U is a set of solutions for (I, J), it is obvious that if $\mathbf{t} \in$ **certain**$(q, (I, J))$, then $\mathbf{t} \in \bigcap_{J' \in U} q(J')_{\downarrow}$. Toward the other direction, assume that $\mathbf{t} \in \bigcap_{J' \in U} q(J')_{\downarrow}$. Let J_{sol} be an arbitrary solution for (I, J). Since U is a universal basis for (I, J), there is a solution $J_u \in U$ and a homomorphism h from J_u to J_{sol}. Since unions of conjunctive queries are preserved under homomorphisms and since $\mathbf{t} \in q(J_u)$, we have that $h(\mathbf{t}) \in q(J_{sol})$. Moreover, $h(\mathbf{t}) = \mathbf{t}$, since homomorphisms map constants from I to themselves. Thus, $\mathbf{t} \in q(J_{sol})$ and so $\mathbf{t} \in$ **certain**$(q, (I, J))$, since J_{sol} was taken to be an arbitary solution.

The proof of the other direction makes use of a connection, first discovered by Chandra and Merlin [1977], between homomorphisms and *canonical* conjunctive queries. If K is an instance with n elements (constants or labeled nulls) in its active domain, then the *canonical conjunctive query of* K is the Boolean conjunctive query q^K obtained by taking the conjunction of all the facts of K and then replacing all labeled nulls by existentially quantified variables. For example, if

$$K = \{E(a, b), E(b, n_1), E(n_1, n_2), E(n_2, a)\},$$

where a, b are constants and n_1, n_2 are labeled nulls, then q^K is the Boolean conjunctive query

$$\exists x_1 \exists x_2 (E(a, b) \wedge E(b, x_1) \wedge E(x_1, x_2) \wedge E(x_2, a)).$$

Chandra and Merlin [1977] showed that, if K and K' are two instances over the same schema, then there is a homomorphism from K to K' if and only if $q^K(K') = \texttt{true}$. With this basic result at hand, we are now ready to complete the proof.

Assume that U is adequate for computing the certain answers of finite unions of target conjunctive queries. Let $J_1, \ldots, J_m$ be an enumeration of all members of U and let q be the finite union $(q^{J_1} \vee \cdots \vee q^{J_m})$ of the canonical conjunctive queries of $J_1, \ldots, J_m$. Clearly, for every $i \leq m$, we have that $q(J_i) = \texttt{true}$, which implies that $\bigcap_{J' \in U} q(J')_{\downarrow} = \texttt{true}$. Consequently, by the adequacy of U, we have that **certain**$(q, (I, J)) = \texttt{true}$. This implies that if J_{sol} is an arbitrary solution for (I, J), then $q(J_{sol}) = \texttt{true}$, which, in turn, implies that there is some $i \leq m$ such that $q^{J_i}(J_{sol}) = \texttt{true}$. It follows that there is a homomorphism from J_i to J_{sol} and, thus, U is a universal basis for (I, J). □

Several remarks are in order now. First, an inspection of the proof of the preceding Proposition 7.4 reveals that, if U is a universal basis, then U is adequate for computing the certain answers of every target query that is preserved under homomorphisms; in particular, U is adequate for computing the certain answers of unions of conjunctive queries. Moreover, for U to be a universal basis, it suffices to be adequate for computing the certain answers of finite unions of Boolean target conjunctive queries. It should also be noted that Proposition 7.4 can be construed as analogous to Proposition 4.2 in Fagin et al. [2005a] about data exchange; the latter proposition asserts that, in the case of data exchange, a solution is universal if and only if it is adequate for computing the certain answers of conjunctive queries.

Next, we show that in peer data exchange settings with no target constraints, universal bases exist if and only if solutions exist. Moreover, we establish an upper bound on the size of the smallest universal basis.

THEOREM 7.5. *Let $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ be a peer data exchange setting with no target constraints. Then, for every source instance I and every target instance J, the following statements are equivalent:*

(1) *There is a solution J' for (I, J) in $\mathcal{P}$.*
(2) *There is a universal basis U for (I, J) such that the cardinality of U is at most exponential in the sizes of I and J, and every member of U is of size at most polynomial in the sizes of I and J.*

Moreover, there is an exponential time algorithm that, given a source instance I and a target instance J, determines if a solution for (I, J) exists and, when a solution exists, constructs a universal basis for (I, J) as above.

PROOF. Consider the peer data exchange setting $\mathcal{S} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \emptyset, \emptyset)$ obtained from $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ by removing all target-to-source dependencies in Σ_{ts}. The proofs of the results in Fagin et al. [2005a] about the chase procedure in data exchange can be extended in a straightforward manner to show that there is a polynomial-time algorithm that, given a source instance I and a target instance J, returns a canonical universal solution J_{can} for (I, J) in $\mathcal{S}$. In particular, J_{can} has the following properties: (a) $J \subseteq J_{can}$; (b) $(I, J) \models \Sigma_{st}$; (c) if J_{sol} is a solution for (I, J) in $\mathcal{S}$, then there is a homomorphism from J_{can} to J_{sol}. Note that the naive chase, which we introduced in Section 5, could also be used to derive a (different) universal solution for (I, J) in polynomial time.

We are now ready to describe an exponential-time algorithm that, given a source instance I and a target instance J, determines whether a solution for (I, J) in the peer data exchange setting $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ exists and, if it does, returns a universal basis for (I, J) in $\mathcal{P}$.

Step 1. By chasing (I, J) with Σ_{st}, construct a canonical universal solution J_{can} for (I, J) in $\mathcal{S}$.

Note that the active domain $Dom(J_{can})$ of J_{can} is the union of a subset of the active domain $Dom(I)$ of I, the active domain $Dom(J)$ of J, and the set N of nulls generated by the chase procedure. Let B be a set of new values such that $B \cap N = \emptyset$ and $|B| = |N|$.

Step 2. For every mapping $h : Dom(J_{can}) \to Dom(I) \cup Dom(J) \cup B$ such that $h(c) = c$ for every $c \in Dom(I) \cup Dom(J)$, form a target instance $h(J_{can})$ so that h is an onto homomorphism from J_{can} to $h(J_{can})$. This means that the facts of $h(J_{can})$ are precisely the tuples $(h(d_1), \ldots, h(d_k))$ such that $(d_1, \ldots, d_k)$ is a fact of J_{can}.

Step 3. For every target instance $h(J_{can})$ created in step 2, test whether $h(J_{can})$ is a solution for (I, J) in $\mathcal{P}$. If no such instance is a solution for (I, J) in $\mathcal{P}$, then return: "no solution for (I, J) in $\mathcal{P}$ exists"; otherwise, return the set

$$U = \{h(J_{can}) : h(J_{can}) \text{ is a solution for } (I, J) \text{ in } \mathcal{P}\}$$

of all such solutions.

We claim that the algorithm runs in time exponential in the size of I and the size of J. To see this, note that step 1 can be carried out in polynomial time in the size of I and the size of J; in particular, J_{can} and B are of polynomial size. step 2 and step 3 can be carried out in exponential time in the size of I and the size of J, since there are exponentially many mappings h from $Dom(J_{can})$ to $Dom(I) \cup Dom(J) \cup B$ and for each such mapping h considered, we can test in polynomial time whether $h(J_{can})$ is a solution for (I, J) in $\mathcal{P}$. Moreover, each target instance $h(J_{can})$ in U is of size polynomial in the sizes of I and J.

Clearly, if the algorithm terminates by returning a set U, then a solution for (I, J) in $\mathcal{P}$ exists. So, to show the correctness of this algorithm, it suffices to show that if a solution for (I, J) exists, then the algorithm returns a set U that is a universal basis for (I, J) in $\mathcal{P}$.

Let J_{sol} be a solution for (I, J) in $\mathcal{P}$. Since J_{sol} is also a solution for (I, J) in $\mathcal{S}$ and J_{can} is a universal solution for (I, J) in $\mathcal{S}$, there is a homomorphism h' from J_{can} to J_{sol}. Note that, since homomorphisms map constants from I and J to themselves, we have that $h'(c) = c$ for every $c \in Dom(J_{can}) \cap (Dom(I) \cup Dom(J))$. Let $h'(J_{can})$ be the image of J_{can} under h', that is, $h'(J_{can})$ is the subinstance of J_{sol} consisting of all tuples $(h'(d_1), \ldots, h'(d_k))$ such that $(d_1, \ldots, d_k)$ is a fact of J_{can}.

We claim that $h'(J_{can})$ is a solution for (I, J) in $\mathcal{P}$. To begin with, we have that $J \subseteq h'(J_{can})$, since $J \subseteq J_{can}$ and h' is the identity mapping on J. We now have to show that $(I, h'(J_{can})) \models \Sigma_{st} \cup \Sigma_{ts}$. Suppose first that $\forall \mathbf{x}(\phi_s(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi_t(\mathbf{x}, \mathbf{y}))$ is a source-to-target tgd in Σ_{st} and $\mathbf{a}$ is a tuple of values from I such that $I \models \phi_s(\mathbf{a})$. Since J_{can} is a solution for (I, J) in $\mathcal{S}$, there is a tuple $\mathbf{b} = (b_1, \ldots, b_m)$ of values in J_{can} such that $J_{can} \models \psi_t(\mathbf{a}, \mathbf{b})$. Since h' is a homomorphism from J_{can} to J_{sol} and $h'(\mathbf{a}) = \mathbf{a}$, it follows that $h(J_{can}) \models \psi_t(\mathbf{a}, h'(\mathbf{b}))$, where $h'(\mathbf{b}) = (h'(b_1), \ldots, h'(b_m))$. This shows that $(I, h(J_{can})) \models \Sigma_{st}$. Finally, if $\forall \mathbf{x}(\alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{y} \beta_s(\mathbf{x}, \mathbf{y}))$ is a target-to-source tgd in Σ_{ts} and $\mathbf{c}$ is a tuple of values from $h'(J_{can})$ such that $h'(J_{can}) \models \alpha_t(\mathbf{c})$, then there is a tuple $\mathbf{d}$ of values from I such that $I \models \beta_s(\mathbf{c}, \mathbf{d})$. This is so because $h'(J_{can})$ is a subinstance of J_{sol} and $(J_{sol}, I) \models \forall \mathbf{x}(\alpha_t(\mathbf{x}) \rightarrow \exists \mathbf{y} \beta_s(\mathbf{x}, \mathbf{y}))$. This shows that $(I, h'(J_{can})) \models \Sigma_{ts}$, which the proof of the claim that $h'(J_{can})$ is a solution for (I, J) in $\mathcal{P}$.

Next we claim that $h'(J_{can})$ is isomorphic to a target instance of the form $h(J_{can})$ for some mapping $h : Dom(J_{can}) \rightarrow Dom(J_{can}) \cup B$ such that $h(c) = c$ for every $c \in Dom(I) \cup Dom(J)$, as in step 2 of the algorithm. Intuitively, $h(J_{can})$ is obtained from $h'(J_{can})$ by renaming the images of the nulls of J_{can} by elements of B. More precisely, let $D = \{h'(w) : w \in N\}$, that is, D is the set of the images of the nulls of J_{can} under h'. Since $|D| \leq |N| \leq |B|$, there is a one-to-one mapping $h'' : D \rightarrow B$. Then the desired mapping $h : Dom(J_{can}) \rightarrow Dom(J_{can}) \cup B$ is defined as follows: if $c \in Dom(I) \cup Dom(J)$, then $h(c) = c$; if $c \in N$, then $h(c) = h''(h'(c))$.

Since $h(J_{can})$ is isomorphic to $h'(J_{can})$ and $h'(J_{can})$ is a solution for (I, J) in $\mathcal{P}$, we have that $h(J_{can})$ is also a solution for (I, J) in $\mathcal{P}$ and, thus, it is a member of U. Moreover, there is a homomorphism from $h(J_{can})$ to J_{sol}, since $h(J_{can})$ is isomorphic to the subinstance $h'(J_{can})$ of J_{sol}. $\square$

Suppose that $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ is a peer data exchange setting such that Σ_{st} is a set of full source-to-target tgds. In this case, J_{can} has no nulls and, consequently, the universal basis returned by the algorithm in the proof of Theorem 7.5 is the singleton $U = \{J_{can}\}$. This yields a different proof of Corollary 6.5. It also yields a much simpler algorithm for computing the certain answers of target conjunctive queries in such peer data exchange settings, namely, evaluate the query on J_{can}. These observations are summarized in the following result.

COROLLARY 7.6. *Suppose that $\mathcal{P} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_{ts}, \emptyset)$ is a PDE setting such that Σ_{st} is a set of full source-to-target tgds. Then, for every source instance I and every target instance J, the following statements are equivalent:*

(1) *There is a solution J' for (I, J) in $\mathcal{P}$.*
(2) *J_{can} is a solution for (I, J) in $\mathcal{P}$, where J_{can} is the result of chasing (I, J) with Σ_{st}.*

Moreover, if a solution for (I, J) exists, then J_{can} is adequate for computing the certain answers of target conjunctive queries. Consequently, for every target conjunctive query q, there is a polynomial-time algorithm for computing the certain answers of q in $\mathcal{P}$.

It should be noted that there are PDE settings in $\mathcal{C}_{tract}$ for which every universal basis has exponential size. For example, consider the following PDE setting:

$$\begin{aligned} \Sigma_{st} &: E(x, y) \to \exists v C(x, v) \\ \Sigma_{ts} &: C(x, v) \to H(v) \\ & C(x, v) \to \exists y E(x, y) \end{aligned}$$

Assume that $G = (V, E)$ is an arbitrary graph and $H = \{r, g, b\}$ is the set of three colors r, g, b. Then a solution exists for the source instance (E, H) and the target instance $\emptyset$. Actually, for every function $c : V \to H$, the target instance $C = \{(v, c(v)) : v \in V\}$ is a solution. Note that there are $3^{|V|}$ such solutions and no homomorphism exists between any two distinct solutions. It follows that every universal basis must contain all such solutions and, thus, must be of size at least $3^{|V|}$.

Finally, it should be pointed out that the proof of Theorem 7.5 does not go through if target dependencies are allowed in the peer data exchange setting. The problem is that, in the presence of target tgds, the instance $h'(J_{can})$ may not be a solution for (I, J). If, however, Σ_t is a set of target egds, then $h'(J_{can})$ is a solution, because it is a subinstance of the solution J_{sol} and satisfaction of egds is preserved under subinstances. Thus Theorem 7.5 also holds if target egds are allowed. It remains an open problem to study the existence of universal bases for peer data exchange settings in which both target egds and target tgds are allowed.

8. CONCLUSIONS

We have introduced a conceptually simple, yet powerful, framework for data sharing among independent peers. Peer data exchange models a scenario in which a target peer receives data from an autonomous source and has no authority to modify the data of the source peer. At the same time, the target peer may specify what data it is willing to receive, and the exchange makes use of source-to-target and target-to-source schema mappings. Peer data exchange is both a generalization of data exchange and a special case of peer data management. This intermediate strength of peer data exhange is reflected in the computational complexity of the two main algorithmic problems associated with it: testing for the existence of solutions and computing the certain answers of target queries. Indeed, we have shown that, in peer data exchange, the existence-of-solutions problem is NP-complete and the data complexity of the certain answers of target conjunctive queries is coNP-complete. In contrast, these two problems are tractable for fairly general data exchange settings and undecidable for full-fledged peer data management systems.

We have also explored the boundary between tractability and intractability for the two main algorithmic problems in peer data exchange. To this effect, we identified a broad class of PDE settings with no target constraints for which the existence-of-solutions problem is solvable in polynomial time. Moreover, for every peer data exchange setting in this class, we have found a maximal collection of target conjunctive queries whose certain answers can be computed in polynomial time. An important consequence of our results is that for peer data exchange settings with no target constraints and such that all source-to-target tgds are full, both the existence-of-solutions and the computation of the certain answers of every target conjunctive query are tractable problems. Finally, we have shed additional light on the differences between peer data exchange and data exchange by introducing and studying the notion of a universal basis of solutions in peer data exchange, which is a relaxation of the notion of a universal solution in data exchange.

It remains an open problem to identify broad classes of peer data exchange settings with target constraints for which testing for the existence of solutions and computing the certain answers of target conjunctive queries are tractable problems. In a different direction, it would be worth exploring alternative semantics in peer data exchange. Specifically, semantics based on repairs [Arenas et al. 1999; Bertossi and Bravo 2004] may be a meaningful alternative for query answering when no solutions exists. Moreover, "good approximate" solutions may turn out to be useful for actual exchange of data when solutions do exist, but none is universal.

ACKNOWLEDGMENTS

We wish to thank the three anonymous referees for their careful reviews of an earlier version of this article.

REFERENCES

Abiteboul, S. and Duschka, O. M. 1998. Complexity of answering queries using materialized views. In *Proceedings of the ACM Symposium on Principles of Database Systems* (PODS). 254–263.

Arenas, M., Bertossi, L., and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the ACM Symposium on Principles of Database Systems* (PODS). 68–79.

Beeri, C. and Vardi, M. 1984. A proof procedure for data dependencies. *J. Assoc. Comput. Mach. 31*, 4, 718–741.

BERNSTEIN, P., GIUNCHIGLIA, F., KEMENTSIETSIDIS, A., MYLOPOULOS, J., SERAFINI, L., AND ZAIHRAYEU, I. 2002. Data management for peer-to-peer computing: A vision. In *Proceedings of the International Workshop on the Web and Databases* (WebDB). 89–94.

BERTOSSI, L. AND BRAVO, L. 2004. Query answering in peer-to-peer data exchange systems. In *Proceedings of the EDBT Workshop on Peer-to-Peer Computing and Databases*. 476–485.

CALÌ, A., CALVANESE, D., DE GIACOMO, G., AND LENZERINI, M. 2004. Data integration under integrity constraints. *Inform. Syst. 29*, 147–163.

CALÌ, A., LEMBO, D., AND ROSATI, R. 2003. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proceedings of the ACM Symposium on Principles of Database Systems* (PODS). 260–271.

CALVANESE, D., DAMAGGIO, E., DE GIACOMO, G., LENZERINI, M., AND ROSATI, R. 2004a. Semantic data integration in P2P systems. In *Databases, Information Systems, and Peer-to-Peer Computing*. Lecture Notes in Computer Science, vol. 2944. Springer-Verlag, Berlin, Germany, 77–90.

CALVANESE, D., DE GIACOMO, G., LEMBO, D., LENZERINI, M., AND ROSATI, R. 2005. Inconsistency tolerance in P2P data integration: An epistemic logic approach. In *Database Programming Languages*. Lecture Notes in Computer Science, vol. 3774. Springer-Verlag, Berlin, Germany, 90–105.

CALVANESE, D., DE GIACOMO, G., LENZERINI, M., AND ROSATI, R. 2004b. Logical foundations of peer-to-peer data integration. In *Proceedings of the ACM Symposium on Principles of Database Systems* (PODS). 241–251.

CHANDRA, A. K. AND MERLIN, P. M. 1977. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the ACM Symposium on Theory of Computing* (STOC). 77–90.

DEUTSCH, A. AND TANNEN, V. 2003. Reformulation of XML queries and constraints. In *Proceedings of the International Conference on Database Theory* (ICDT). 225–241.

FAGIN, R., KOLAITIS, P. G., MILLER, R. J., AND POPA, L. 2003. Data exchange: Semantics and query answering. In *Proceedings of the International Conference on Database Theory* (ICDT). 207–224.

FAGIN, R., KOLAITIS, P. G., MILLER, R. J., AND POPA, L. 2005a. Data exchange: Semantics and query answering. *Theoret. Comput. Sci.* (special issue with selected papers from ICDT 2003) *336*, 1, 89–124.

FAGIN, R., KOLAITIS, P. G., AND POPA, L. 2005b. Data exchange: Getting to the core. *ACM Trans. Database Syst.* (special issue with selected papers from PODS 2003) *30*, 1, 174–210.

FRANCONI, E., KUPER, G., LOPATENKO, A., AND SERAFINI, L. 2003. A robust logical and computational characterisation of peer-to-peer database systems. In *Proceedings of the VLDB Workshop on Databases, Information Systems and Peer-to-Peer Computing*.

FRANCONI, E., KUPER, G., LOPATENKO, A., AND ZAIHRAYEU, I. 2004. The coDB robust peer-to-peer database system. In *Proceedings of the Symposium on Advanced Database Systems*. 382–393.

GRAHNE, G. 1991. *The Problem of Incomplete Information in Relational Databases*. Lecture Notes in Computer Science, vol. 554. Springer-Verlag, Berlin, Germany.

GRAHNE, G. AND MENDELZON, A. 1999. Tableau techniques for querying information sources through global schemas. In *Proceedings of the International Conference on Database Theory* (ICDT). 332–347.

HALEVY, A., IVES, Z., SUCIU, D., AND TATARINOV, I. 2005. Schema mediation for large-scale semantic data sharing. *VLDB J. 14*, 1, 68–83.

KOLAITIS, P. G., PANTTAJA, J., AND TAN, W. 2006. The complexity of data exchange. In *Proceedings of the ACM Symposium on Principles of Database Systems* (PODS). To appear.

LENZERINI, M. 2002. Data integration: A theoretical perspective. In *Proceedings of the ACM Symposium on Principles of Database Systems* (PODS). 233–246.

LI, C. 2004. Raccoon: A peer-based system for data integration and sharing. In *Proceedings of the International Conference on Data Engineering* (ICDE). 852. (System Demonstration).

NASH, A., DEUTSCH, A., AND REMMEL, J. 2006. Data exchange, data integration, and chase. Tech. rep. CS2006-0859, University of California, San Diego, San Diego, CA.

O'DONOVAN, C., MARTIN, M. J., GATTIKER, A., GASTEIGER, E., BAIROCH, A., AND APWEILER, R. 2002. High-quality protein knowledge resource: Swiss-prot and trembl. *Briefings Bioinformat. 3*, 3, 275–284.

POPA, L., VELEGRAKIS, Y., MILLER, R. J., HERNÁNDEZ, M. A., AND FAGIN, R. 2002. Translating Web data. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*. 598–609.

Tatarinov, I. 2004. Semantic data sharing with a peer data management system. Ph.D. dissertation. University of Washington, Seattle, Washington.

Tatarinov, I. and Halevy, A. Y. 2004. Efficient query reformulation in peer data management systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 539–550.

van der Meyden, R. 1998. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, J. Chomicki and G. Saake, Eds. Kluwer, Dordrecht, The Netherlands, 307–356.

Received October 2005; revised May 2006; accepted July 2006