Querying Data Exchange Settings Beyond Positive Queries

Marco Calautti[1], Sergio Greco[2], Cristian Molinaro[2] and Irina Trubitsyna[2]

[1]Department of Information Engineering and Computer Science, University of Trento, Italy
[2]Department of Computer Science, Modeling, Electronics and Systems Engineering, University of Calabria, Italy

Abstract

Data exchange, the problem of transferring data from a source schema to a target schema, has been studied for several years. The semantics of answering positive queries over the target schema has been defined in early works, but little attention has been paid to more general queries. A few semantics proposals for more general queries exist but they either do not properly extend the standard semantics under positive queries, giving rise to counterintuitive answers, or they make query answering undecidable even for the most important data exchange settings, e.g., with weakly-acyclic dependencies.

The goal of this paper is to provide a new semantics for data exchange that is able to deal with general queries. At the same time, we want our semantics to coincide with the classical one when focusing on positive queries, and to not trade-off too much in terms of complexity of query answering. We show that query answering is undecidable in general under the new semantics, but it is coNP-complete when the dependencies are weakly-acyclic. Moreover, in the latter case, we show that our semantics allow for the construction of a representative target instance, similar in spirit to a universal solution, that can be exploited for computing approximate answers, instead of exact ones.

Keywords

Data Exchange, Semantics, Closed Word Assumption, Approximations

1. Introduction

Data exchange is the problem of transferring data from a source schema to a target schema, where the transfer process is usually described via so-called schema mappings: a set of logical assertions specifying how the data should be moved and restructured. Furthermore, the target schema may have its own constraints to be satisfied. Schema mappings and target constraints are usually encoded via standard database dependencies: *tuple-generating dependencies* (TGDs) and *equality-generating dependencies* (EGDs). Thus, given an instance I over the source schema S, the goal is to materialize an instance J over the target schema T, called *solution*, in such a way that I and J together satisfy the dependencies.

Since multiple solutions might exist, a precise semantics for answering queries is needed. By now, the *certain answers* semantics is the most accepted one. The certain answers to a query is the set of all tuples that are answers to the query in every solution of the data exchange setting [1]. Although it has been formally shown that for positive queries (e.g., conjunctive

Datalog 2.0 2022: 4th International Workshop on the Resurgence of Datalog in Academia and Industry, September 05, 2022, Genova - Nervi, Italy

marco.calautti@unitn.it (M. Calautti); greco@dimes.unical.it (S. Greco); c.molinaro@dimes.unical.it (C. Molinaro); trubitsyna@dimes.unical.it (I. Trubitsyna)

queries) the notion of solution of [1] is the right one to use, for more general queries such solutions become inappropriate, as they easily lead to counterintuitive results.

Example 1. *Consider a data exchange setting denoted by* $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$*, where* S *is the source schema, storing orders about products in a binary relation* Ord*, where the first argument is the id of the order, and the second one specifies whether the order has been paid. Moreover,* T *is the target schema having unary relations* AllOrd *and* Paid*, storing all orders and paid orders, respectively. The schema mapping is described by the source-to-target TGDs* Σ_{st}*:*

$$\rho_1 = \quad \forall x, y \quad \mathsf{Ord}(x, y) \rightarrow \mathsf{AllOrd}(x), \qquad \rho_2 = \quad \forall x \quad \mathsf{Ord}(x, \mathsf{yes}) \rightarrow \mathsf{Paid}(x).$$

In this example, we assume that the set of target dependencies Σ_t *is empty. The above schema mapping states that all orders in the source schema must be copied to the* AllOrd *relation, and all the paid orders must be copied to the* Paid *relation. Assume the source instance is as follows:*

$$I = \{\mathsf{Ord}(1, \mathsf{yes}), \mathsf{Ord}(2, \mathsf{no})\},$$

and assume we want to pose the query Q *over the target schema asking for all the unpaid orders. This can be written as the following FO query:*

$$Q(x) = \mathsf{AllOrd}(x) \wedge \neg \mathsf{Paid}(x).$$

One would expect the answer to be $\{2\}$*, since the schema mapping above is simply copying* I *to the target schema, and hence* $J = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1)\}$ *should be the only candidate solution. However, under the classical notion of solution of [1], also the instance* $J' = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1), \mathsf{Paid}(2)\}$ *is a solution (since* $I \cup J'$ *satisfies the TGDs), and every order in* J' *is paid. Hence, the certain answers to* Q*, which are computed as the intersection of the answers over all solutions, are empty.* □

The issue above arises because the classical notion of solution is too permissive, in that it allows the existence of facts in a solution that have no support from the source (e.g., $\mathsf{Paid}(2)$ in the solution J' of Example 1 above).

Some efforts exist in the literature that provide alternative notions of solutions for which certain answers to general queries become more meaningful. Prime examples are the works of [2] and [3]. In both approaches, the certain answers in the example above are $\{2\}$. However, also the works above have their own drawbacks. In [2], so-called *CWA-solutions* are introduced, which are a subset of the classical solutions with some restrictions. However, these restrictions are so severe that certain answers over such solutions fail to capture certain answers over classical solutions, when focusing on positive queries. Moreover, even when focusing on more general queries, answers can still be counterintuitive, as shown in the following example.

Example 2. *Consider the data exchange setting* $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$*, where* S *stores employees of a company in the unary relation* Emp*. For* some *employees, the city they live in is known, and it is stored in the binary relation* KnownC*. The target schema* T *contains the binary relation* EmpC*, storing employees and the cities they live in, and the binary relation* SameC*, storing pairs*

of employees living in the same city. The sets $\Sigma_{st} = \{\rho_1, \rho_2\}$ *and* $\Sigma_t = \{\rho_3, \eta\}$ *are as follows (for simplicity, we omit the universal quantifiers):*

$$\begin{array}{llll} \rho_1 = & \mathsf{Emp}(x) \rightarrow \exists z\, \mathsf{EmpC}(x, z), & \rho_3 & = \mathsf{EmpC}(x, y), \mathsf{EmpC}(x', y) \rightarrow \mathsf{SameC}(x, x'), \\ \rho_2 = & \mathsf{KnownC}(x, y) \rightarrow \mathsf{EmpC}(x, y), & \eta & = \mathsf{EmpC}(x, y), \mathsf{EmpC}(x, z) \rightarrow y = z. \end{array}$$

The above setting copies employees from the source to the target. The TGD ρ_1 *states that every copied employee* x *must have some city* z *associated, whereas* ρ_2 *states that when the city* y *of an employee* x *is known, this should be copied as well. Moreover, the target schema requires that employees living in the same city should be stored in relation* SameC *(*ρ_3*), and each employee must live in only one city (*η*). Assume the source instance is*

$$I = \{\mathsf{Emp}(\mathsf{john}), \mathsf{Emp}(\mathsf{mary}), \mathsf{KnownC}(\mathsf{john}, \mathsf{miami})\},$$

and assume our query Q *asks for all pairs of employees living in different cities. This can be written as:*

$$Q(x, x') = \exists y \exists y'\, \mathsf{EmpC}(x, y) \wedge \mathsf{EmpC}(x', y') \wedge \neg \mathsf{SameC}(x, x').$$

One would expect that the set of certain answers to Q *is empty, since it is not certain that* john *and* mary *live in different cities. However, no CWA-solution admits* mary *and* john *to live in the same city, and thus* $(\mathsf{john}, \mathsf{mary})$ *is a certain answer under the CWA-solution-based semantics.* □

The approach of [3], where the notion of GCWA*-solution is presented, seems to be the most promising one. For positive queries, certain answers w.r.t. GCWA*-solutions coincide with certain answers w.r.t. classical solutions. Moreover, GCWA*-solutions solve some other limitations of CWA-solutions, like the one discussed in Example 2. However, the practical applicability of this semantics is somehow limited, since the (rather involved) construction of GCWA*-solutions easily makes certain query answering undecidable, even for very simple settings with only two source-to-target TGDs, and no target dependencies.

Other semantics have been proposed in [4], but they are only defined for data exchange settings without target dependencies. Hence, one needs to assume that the target schema has no dependencies at all.

As a final remark, in a data exchange setting, it is usually assumed that the source is not always available, and thus the materialization of a single solution, over which certain answers can be computed, is a crucial requirement. This is especially true when using weakly-acyclic dependencies, which form the standard language for data exchange [1]. However, none of the semantics above allow for the materialization of such a special solution, for weakly-acyclic settings.

In this paper, we propose a new notion of data exchange solution, dubbed *supported solution*, which allows us to deal with general queries, but at the same time is suitable for practical applications. That is, we show that certain answers under supported solutions naturally generalize certain answers under classical solutions, when focusing on positive queries. Moreover, such solutions do not make any assumption on how values associated to existential variables compare to other values, hence solving issues like the ones of Example 2.

As expected, there is a price to pay to get meaningful answers over general queries: we show that certain answering is undecidable for general settings, but becomes coNP-complete when we focus on weakly-acyclic dependencies.

Morever, we also show that for weakly-acyclic settings, we can construct a target instance in polynomial time, which is similar in spirit to a universal solution of [1], that can be exploited for computing exact answers, for positive queries, and *approximate answers*, for general FO queries, in polynomial time. The latter is achieved by adapting existing approximation algorithms originally defined for querying incomplete databases.

2. Preliminaries

Basics. We consider pairwise disjoint countably infinite sets Const, Var, Null of *constants*, *variables*, and *labeled nulls*. Nulls are denoted by the symbol $\perp$, possibly subscripted. A *term* is a constant, a variable, or a null. We additionally assume the existence of countably infinite set Rel of *relations*, disjoint from the previous ones. A relation R has an *arity*, denoted $ar(R)$, which is a non-negative integer. We also use R/n to say that R is a relation of arity n. A *schema* is a set of relations. A *position* is an expression of the form $R[i]$, where R is a relation and $i \in \{1, \ldots, ar(R)\}$.

An *atom* α (over a schema S) is of the form $R(\mathbf{t})$, where R is an n-ary relation (of S) and $\mathbf{t}$ is a tuple of terms of length n. We use $\mathbf{t}[i]$ to denote the i-th term in $\mathbf{t}$, for $i \in \{1, \ldots, n\}$. An atom without variables is a *fact*. An *instance* I (over a schema S) is a finite set of facts (over S). A *database* D is an instance without nulls. For a set of atoms A, $\mathsf{dom}(A)$ is the set of all terms in A, whereas $\mathsf{var}(A)$ is the set $\mathsf{dom}(A) \cap \mathsf{Var}$. A *homomorphism* from a set of atoms A to a set of atoms B is a function $h : \mathsf{dom}(A) \to \mathsf{dom}(B)$ that is the identity on Const, and such that for each atom $R(\mathbf{t}) = R(t_1, \ldots, t_n) \in A$, $R(h(\mathbf{t})) = R(h(t_1), \ldots, h(t_n)) \in B$.

Dependencies. A *tuple-generating dependency* (TGD) ρ (over a schema S) is a first-order formula of the form $\forall \mathbf{x}, \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$, where $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are disjoint tuples of variables, and φ and ψ are conjunctions of atoms (over S) without nulls, and over the variables in $\mathbf{x}, \mathbf{y}$ and $\mathbf{y}, \mathbf{z}$ respectively. The *body* of ρ, denoted $\mathsf{body}(\rho)$, is $\varphi(\mathbf{x}, \mathbf{y})$, whereas the *head* of ρ, denoted $\mathsf{head}(\rho)$, is $\psi(\mathbf{y}, \mathbf{z})$. We use $\mathsf{exvar}(\rho)$ to denote the tuple $\mathbf{z}$ and $\mathsf{fr}(\rho)$ to denote the tuple $\mathbf{y}$, also called the *frontier of* ρ. An *equality-generating dependency* (EGD) η (over a schema S) is a first-order formula of the form $\forall \mathbf{x}\, \varphi(\mathbf{x}) \rightarrow x = y$, where $\mathbf{x}$ is a tuple of variables, φ a conjunction of atoms (over S) without nulls, and over $\mathbf{x}$, and $x, y \in \mathbf{x}$. The *body of* η, denoted $\mathsf{body}(\eta)$, is $\varphi(\mathbf{x})$, and the *head of* η, denoted $\mathsf{head}(\eta)$, is the equality $x = y$. For clarity, we will omit the universal quantifiers in front of dependencies and replace the conjunction symbol $\wedge$ with a comma. Moreover, with a slight abuse of notation, we sometimes treat a conjunction of atoms as the *set* of its atoms. Consider an instance I. We say that I *satisfies a TGD* ρ if for every homomorphism h from $\mathsf{body}(\rho)$ to I, there is an extension h' of h such that h' is a homomorphism from $\mathsf{head}(\rho)$ to I. We say that I *satisfies an EGD* $\eta = \varphi(\mathbf{x}) \rightarrow x = y$, if for every homomorphism h from $\mathsf{body}(\eta)$ to I, $h(x) = h(y)$. I *satisfies* a set of TGDs and EGDs Σ if I satisfies every TGD and EGD in Σ.

Queries. A *query* $Q(\mathbf{x})$, with free variables $\mathbf{x}$, is a first-order (FO) formula $\varphi(\mathbf{x})$ with free variables $\mathbf{x}$. The *arity* of $Q(\mathbf{x})$, denoted $ar(Q)$, is the number $|\mathbf{x}|$. The *output* of $Q(\mathbf{x})$ over an instance I, denoted $Q(I)$, is the set $\{\mathbf{t} \in \mathsf{dom}(I)^{|\mathbf{x}|} \mid I \models \varphi(\mathbf{t})\}$, where $\models$ is FO entailment.[1] A

[1]We assume active domain semantics, i.e., quantifiers range over the terms in the given instance.

query is *Boolean* if it has arity 0, in which case its output over an instance is either the empty set or the empty tuple $\langle\rangle$. A *conjunctive query* (CQ) is a query of the form $Q(\mathbf{x}) = \exists \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$, where $\varphi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over $\mathbf{x}$ and $\mathbf{y}$. A *union of conjunctive queries* (UCQ) is a query of the form $Q(\mathbf{x}) = \bigvee_{i=1}^{n} Q_i(\mathbf{x})$, where each $Q_i(\mathbf{x})$ is a CQ. We also refer to UCQs as *positive queries*.

Data Exchange Settings. A *data exchange setting* (or simply setting) is a tuple of the form $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, where S, T are disjoint schemas, called *source* and *target* schema, respectively; Σ_{st} is a finite set of TGDs, called the *source-to-target TGDs* of $\mathcal{S}$, such that for each TGD $\rho \in \Sigma_{st}$, $\mathsf{body}(\rho)$ is over S and $\mathsf{head}(\rho)$ is over T; Σ_t is a finite set of TGDs and EGDs over T, called the *target dependencies* of $\mathcal{S}$. We say $\mathcal{S}$ is *TGD-only* if Σ_t contains only TGDs.

A *source (resp., target) instance of* $\mathcal{S}$ is an instance I over S (resp., T). We assume that source instances are databases, i.e., they do not contain nulls. Given a source instance I of $\mathcal{S}$, a *solution of* I *w.r.t.* $\mathcal{S}$ is a target instance J of $\mathcal{S}$ such that $I \cup J$ satisfies Σ_{st} and J satisfies Σ_t [1]. We use $\mathsf{sol}(I, \mathcal{S})$ to denote the set of all solutions of I w.r.t. $\mathcal{S}$.

Given a data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, a source instance I of $\mathcal{S}$ and a query Q over T, the *certain answers to* Q *over* I *w.r.t.* $\mathcal{S}$ is the set $\mathsf{cert}_{\mathcal{S}}(I, Q) = \bigcap_{J \in \mathsf{sol}(I,\mathcal{S})} Q(J)$.

To distinguish between the notion of solution (resp., certain answers) above and the one defined in Section 3, we will refer to the former as *classical*.

A *universal solution* of I w.r.t. $\mathcal{S}$ is a solution $J \in \mathsf{sol}(I, \mathcal{S})$ such that, for every $J' \in \mathsf{sol}(I, \mathcal{S})$, there is a homomorphism from J to J' [1]. Letting $Q(J)_{\downarrow} = Q(J) \cap \mathsf{Const}^{|\mathbf{x}|}$, for any instance J and query $Q(\mathbf{x})$, the following is well-known:

Theorem 1 ([1]). *Consider a data exchange setting* $\mathcal{S}$*, a source instance* I *of* $\mathcal{S}$ *and a positive query* Q*. If* J *is a universal solution of* I *w.r.t.* $\mathcal{S}$*, then* $\mathsf{cert}_{\mathcal{S}}(I, Q) = Q(J)_{\downarrow}$.

3. Semantics for General Queries

The goal of this section is to introduce a new notion of solution for data exchange that we call *supported*. As already discussed, the main issue we want to solve w.r.t. classical solutions is that such solutions are too permissive, i.e., they allow for the presence of facts that are not a certain consequence of the source instance and the dependencies. Consider again Example 1. The (classical) solution J' in Example 1 is not supported, since from the source instance I and the dependencies, we cannot conclude that the fact $\mathsf{Paid}(2)$ should occur in the target. On the other hand, the solution $J = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1)\}$ is supported: it contains precisely the facts supported by I and the dependencies, and no more than that. Similarly, considering Example 2, the instance $J = \{\mathsf{EmpC}(\mathsf{john}, \mathsf{miami}), \mathsf{EmpC}(\mathsf{mary}, \mathsf{chicago}), \mathsf{SameC}(\mathsf{john}, \mathsf{mary})\}$ is a solution, but it is not supported, since from the source and the dependencies we cannot certainly conclude that john and mary live in the same city. We now formalize the above intuitions.

Consider a TGD ρ and a mapping h from the variables of ρ to Const. We say that a TGD ρ' is a *ground version of* ρ (via h) if $\rho' = h(\mathsf{body}(\rho)) \rightarrow h(\mathsf{head}(\rho))$.

Definition 1 (ex-choice). *An* ex-choice *is a function* γ*, that given as input a TGD* $\rho = \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$ *and a tuple* $\mathbf{t} \in \mathsf{Const}^{|\mathbf{y}|}$*, returns a set* $\gamma(\rho, \mathbf{t})$ *of pairs of the form* (z, c)*, one for each existential variable* $z \in \mathsf{exvar}(\rho)$*, where* c *is a constant of* Const.

Note that if ρ does not contain existential variables, $\gamma(\rho, \mathbf{t})$ is the empty set.

Intuitively, given a TGD, an ex-choice specifies a valuation for the existential variables of the TGD which depends on a given valuation of its frontier variables.

We now define when a ground version of a TGD indeed assigns existential variables according to an ex-choice.

Definition 2 (Coherence). *Consider a TGD $\rho = \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$, an ex-choice γ and a ground version ρ' of ρ via some mapping h. We say that ρ' is* coherent with γ *if for each existential variable $z \in \mathsf{exvar}(\rho)$, $(z, h(z)) \in \gamma(\rho, h(\mathbf{y}))$.*

For a set Σ of TGDs and EGDs, and an ex-choice γ, Σ^γ denotes the set of dependencies obtained from Σ, where each TGD ρ in Σ is replaced with all ground versions of ρ that are coherent with γ. Note that the set Σ^γ can be infinite. We are now ready to present our notion of solution.

Definition 3 (Supported Solution). *Consider a setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$ and a source instance I of $\mathcal{S}$. A target instance J of $\mathcal{S}$ is a* supported solution of I w.r.t. $\mathcal{S}$ *if there exists an ex-choice γ such that $I \cup J$ satisfies Σ_{st}^γ and J satisfies Σ_t^γ, and there is no other target instance $J' \subsetneq J$ of $\mathcal{S}$ such that $I \cup J'$ satisfies Σ_{st}^γ and J' satisfies Σ_t^γ.* □

Note that a supported solution contains no nulls. We use $\mathsf{ssol}(I, \mathcal{S})$ to denote the set of all supported solutions of I w.r.t. $\mathcal{S}$.

Example 3. *Consider the data exchange setting $\mathcal{S}$ and the source instance I of Example 2. The target instance $J = \{\mathsf{EmpC}(\mathsf{john}, \mathsf{miami}), \mathsf{EmpC}(\mathsf{mary}, \mathsf{chicago})\}$ is a supported solution of I w.r.t. $\mathcal{S}$. Indeed, consider the ex-choice γ such that $\gamma(\rho_1, \mathsf{john}) = \{(z, \mathsf{miami})\}$, and $\gamma(\rho_1, \mathsf{mary}) = \{(z, \mathsf{chicago})\}$. Then, Σ_{st}^γ is*

$$\{\mathsf{KnownC}(\alpha, \beta) \rightarrow \mathsf{EmpC}(\alpha, \beta) \mid \alpha, \beta \in \mathsf{Const}\} \cup$$
$$\{\mathsf{Emp}(\alpha) \rightarrow \mathsf{EmpC}(\alpha, \beta) \mid \alpha \in \mathsf{Const} \wedge (z, \beta) \in \gamma(\rho_1, \alpha)\},$$

whereas Σ_t^γ is the set containing the EGD η of Example 2, and the set of TGDs

$$\{\mathsf{EmpC}(\alpha, \beta), \mathsf{EmpC}(\alpha', \beta) \rightarrow \mathsf{SameC}(\alpha, \alpha') \mid \alpha, \alpha', \beta \in \mathsf{Const}\}.$$

Clearly, $I \cup J$ satisfies Σ_{st}^γ, and J satisfies Σ_t^γ, and any other strict subset J' of J is such that $I \cup J'$ does not satisfy Σ_{st}^γ. Another supported solution is $\{\mathsf{EmpC}(\mathsf{john}, \mathsf{miami}), \mathsf{EmpC}(\mathsf{mary}, \mathsf{miami}), \mathsf{SameC}(\mathsf{john}, \mathsf{mary})\}$. □

With the notion of supported solution in place, it is now straightforward to define the supported certain answers.

Definition 4 (Supported Certain Answers). *Consider a data exchange setting $\mathcal{S}$, a source instance I of $\mathcal{S}$ and a query Q over T. The* supported certain answers to Q over I w.r.t. $\mathcal{S}$ *is the set of tuples* $\mathsf{scert}_{\mathcal{S}}(I, Q) = \bigcap_{J \in \mathsf{ssol}(I, \mathcal{S})} Q(J)$.

Example 4. *Consider the data exchange setting $\mathcal{S}$, the source instance I, and the query Q of Example 1. It is not difficult to see that the only supported solution of I w.r.t. $\mathcal{S}$ is the instance $J = \{\mathsf{AllOrd}(1), \mathsf{AllOrd}(2), \mathsf{Paid}(1)\}$. Thus, the supported certain answers to Q over I w.r.t. $\mathcal{S}$ are $\mathsf{scert}_{\mathcal{S}}(I, Q) = Q(J) = \{2\}$. Consider now the data exchange setting $\mathcal{S}$, the source instance I, and the query Q of Example 2. Then, one can verify that $\mathsf{scert}_{\mathcal{S}}(I, Q) = \emptyset$.* □

We now start establishing some important results regarding supported solutions and supported certain answers. The following theorem states that supported solutions are a refined subset of the classical ones, but whether a supported solution exists is still tightly related to the existence of a classical one.

Theorem 2. *Consider a data exchange setting $\mathcal{S}$. For every source instance I of $\mathcal{S}$, its holds that* (1) $\mathsf{ssol}(I, \mathcal{S}) \subseteq \mathsf{sol}(I, \mathcal{S})$, *and* (2) $\mathsf{ssol}(I, \mathcal{S}) = \emptyset$ *iff* $\mathsf{sol}(I, \mathcal{S}) = \emptyset$.

Regarding certain answers, we show that supported solutions indeed enjoy an important property: supported certain answers and classical certain answers coincide, when focusing on positive queries. Note that this does not necessarily follow from Theorem 2.

Theorem 3. *Consider a setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$ and a positive query Q over T. For every source instance I of $\mathcal{S}$, $\mathsf{scert}_{\mathcal{S}}(I, Q) = \mathsf{cert}_{\mathcal{S}}(I, Q)$.*

From the above, we conclude that for positive queries, certain query answering can be performed as done in the classical setting, and thus all important results from that setting, like query answering via universal solutions, carry over.

Corollary 1. *Consider a setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$ and a positive query Q over T. If J is a (classical) universal solution of I w.r.t. $\mathcal{S}$, then $\mathsf{scert}_{\mathcal{S}}(I, Q) = Q(J)_{\downarrow}$.*

Proof. It follows from Theorem 1 and Theorem 3. □

We now move to the complexity analysis of the two most important data exchange tasks: deciding whether a supported solution exists, and computing the supported certain answers to a query.

4. Complexity

In data exchange, it is usually assumed that a setting $\mathcal{S}$ does not change over time, and a given query Q is much smaller than a given source instance. Thus, for understanding the complexity of a data exchange problem, it is customary to assume that $\mathcal{S}$ and Q are fixed, and only I is considered in the complexity analysis, i.e., we consider the *data complexity* of the problem. Hence, the problems we are going to discuss will always be parametrized via a setting $\mathcal{S}$, and a query Q (for query answering tasks). The first problem we consider is deciding whether a supported solution exists; $\mathcal{S}$ is a fixed data exchange setting.

PROBLEM :	EXISTS-SSOL($\mathcal{S}$)
INPUT :	A source instance I of $\mathcal{S}$.
QUESTION :	Is $\mathsf{ssol}(I, \mathcal{S}) \neq \emptyset$?

The above problem is very important in data exchange, as one of the main goals is to actually construct a target instance that can be exploited for query answering purposes. Hence, knowing in advance whether at least a supported solution exists is of paramount importance.

Thanks to Item 2 of Theorem 2, all the complexity results for checking the existence of a classical solution can be directly transfered to our problem.

Theorem 4. *There exists a data exchange setting $\mathcal{S}$ such that* EXISTS-SSOL($\mathcal{S}$) *is undecidable.*

Proof. It follows from Theorem 2 and from the fact that there exists a data exchange setting $\mathcal{S}$ such that checking whether a classical solution exists is undecidable [5]. □

Despite the negative result above, we also inherit positive results from the literature, when focusing on some of the most important data exchange scenarios, known as *weakly-acyclic*. Such settings only allow target TGDs to belong to the language of weakly-acyclic TGDs, which have been first introduced in the seminal paper [1], and is now well-established as the main language for data exchange purposes.

We start by introducing the notion of weak-acyclicity. We recall that for a schema S, $\mathsf{pos}(\mathsf{S})$ denotes the set of all positions $R[i]$, where $R/n \in \mathsf{S}$ and $i \in \{1, \ldots, n\}$, and for a TGD $\rho = \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$, $\mathsf{fr}(\rho)$ denotes the tuple $\mathbf{y}$.

Definition 5 (Dependency Graph [1])**.** *Consider a set Σ of TGDs over a schema S. The* dependency graph *of Σ is a directed graph $\mathsf{dg}(\Sigma) = (N, E)$, where $N = \mathsf{pos}(\mathsf{S})$ and E contains only the following edges. For each $\rho \in \Sigma$, for each $x \in \mathsf{fr}(\rho)$, and for each position π in $\mathsf{body}(\rho)$ where x occurs:*

- *there is a* normal *edge $(\pi, \pi') \in E$, for each position π' in $\mathsf{head}(\rho)$ where x occurs, and*
- *there is a* special *edge $(\pi, \pi') \in E$, for each position π' in $\mathsf{head}(\rho)$ where an existentially quantified variable $z \in \mathsf{exvar}(\rho)$ occurs.* □

Definition 6. *A set of TGDs Σ is* weakly-acyclic *if no cycle in $\mathsf{dg}(\Sigma)$ contains a special edge. A data exchange setting $\langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$ is weakly-acyclic if the set of TGDs in Σ_t is weakly-acyclic.*□

Example 5. *The settings of Examples 1 and 2 are weakly-acyclic, whereas the data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, where $\mathsf{S} = \{S/2\}$, $\mathsf{T} = \{T/2\}$, $\Sigma_{st} = \{S(x, y) \rightarrow T(x, y)\}$, and $\Sigma_t = \{T(x, y) \rightarrow \exists z\, T(y, z)\}$ is not, since $(T[2], T[2])$ is a special edge in $\mathsf{dg}(\Sigma_t)$.* □

The following result follows.

Theorem 5. *For every weakly-acyclic data exchange setting $\mathcal{S}$,* EXISTS-SSOL($\mathcal{S}$) *is in* PTIME.

Proof. It follows from Theorem 2 and [1, Corollary 3.10]. □

We now move to the second crucial task: computing supported certain answers. Since this problem outputs a set, it is standard to focus on its decision version. For a fixed data exchange setting $\mathcal{S}$ and a fixed query Q, we consider the following decision problem:

PROBLEM :	$\mathsf{SCERT}(\mathcal{S}, Q)$
INPUT :	A source instance I of $\mathcal{S}$ and a tuple $\mathbf{t} \in \mathsf{Const}^{ar(Q)}$.
QUESTION :	Is $\mathbf{t} \in \mathsf{scert}_{\mathcal{S}}(I, Q)$?

One can easily show that the above problem is logspace equivalent to the one of computing the supported certain answers.

We start by studying the problem in its full generality, and show that there is a price to pay for query answering with general queries.

Theorem 6. *There exists a data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, with Σ_t having only TGDs, and a query Q over T, such that $\mathsf{SCERT}(\mathcal{S}, Q)$ is undecidable.*

Although the complexity result above tells us that computing supported certain answers might be infeasible in some settings, we can show that for weakly-acyclic settings, the complexity is more manageable.

Theorem 7. *For every weakly-acyclic setting $\mathcal{S}$ and every query Q, $\mathsf{SCERT}(\mathcal{S}, Q)$ is in* coNP*, and there exists a weakly-acyclic setting $\mathcal{S}$ that is TGD-only and a query Q such that $\mathsf{SCERT}(\mathcal{S}, Q)$ is* coNP-hard*.*

We point out that the above result is in contrast with all the data exchange semantics discussed in the introduction, for which computing certain answers is undecidable, even for weakly-acyclic settings [2, 3].

We conclude this section by recalling that for positive queries, supported certain answers coincide with the classical ones (Theorem 3), and computing (classical) certain answers for weakly-acyclic settings, under positive queries, is tractable [1]. Hence, the result below follows.

Corollary 2. *For every weakly-acyclic setting $\mathcal{S}$ and every positive query Q, $\mathsf{SCERT}(\mathcal{S}, Q)$ is in* PTIME*.*

5. Query Answering via Materialization

As already discussed in the introduction, it is crucial in data exchange, whenever it is possible, to materialize a target instance starting from the source instance and the schema mapping, in such a way that supported certain query answers can be computed by considering the target instance alone. The goal of this section is thus to study the problem of materializing such an instance, when focusing on our notion of supported solutions.

It would be very useful if such a special target instance could be computed in polynomial-time, already for weakly-acyclic settings. However, due to Theorem 7, this would imply PTIME $=$ coNP. Hence, we need something different.

We introduce a special instance that enjoys the following properties: the answers over this instance are an approximation (i.e., a subset) of the supported certain answers, for general queries, but they coincide with supported certain answers, for positive queries. We also show that we can compute such an instance in polynomial time for weakly-acyclic settings.

Our approach relies on conditional instances [6], which we introduce in the following.

Conditional instances. A *valuation* ν is a mapping from $\mathsf{Const} \cup \mathsf{Null}$ to Const that is the identity on Const. A *condition* ϕ is an expression that can be built using the standard logical connectives $\wedge$, $\vee$, $\neg$, $\Rightarrow$, and expressions of the form $t = u$, where $t, u \in \mathsf{Const} \cup \mathsf{Null}$. We will also use $t \neq u$ as a shorthand for $\neg(t = u)$. We write $\nu \models \phi$ to state that ν satisfies ϕ, and

$\phi \models \psi$ if all valuations satisfying ϕ satisfy the condition ψ. A *conditional fact* is a pair $\langle \alpha, \phi \rangle$, where α is a fact and ϕ is a condition. A *conditional instance* $\mathcal{I}$ is a finite set of conditional facts. We also denote $\mathcal{I}[1] = \{\alpha \mid \langle \alpha, \phi \rangle \in \mathcal{I}\}$. A *possible world* of a conditional instance $\mathcal{I}$ is an instance I such that there exists a valuation ν with $I = \{\nu(\alpha) \mid \langle \alpha, \phi \rangle \in \mathcal{I} \text{ and } \nu \models \phi\}$. We use $\mathsf{pw}(\mathcal{I})$ to denote the set of all possible worlds of $\mathcal{I}$.

Definition 7. *Consider a conditional instance $\mathcal{I}$ and a query Q. The* conditional certain answers *of Q over $\mathcal{I}$ is the set* $\mathsf{con\text{-}cert}(\mathcal{I}, Q) = \bigcap_{J \in \mathsf{pw}(\mathcal{I})} Q(J)$. □

We are now ready to introduce our main tool.

Definition 8 (Approximate Conditional Solution)**.** *Consider a data exchange setting $\mathcal{S}$ and a source instance I of $\mathcal{S}$. A conditional instance $\mathcal{J}$ is an* approximate conditional solution *of I w.r.t. $\mathcal{S}$, if for every query Q:*

1. $\mathsf{ssol}(I, \mathcal{S}) \subseteq \mathsf{pw}(\mathcal{J})$*, and thus* $\mathsf{con\text{-}cert}(\mathcal{J}, Q) \subseteq \mathsf{scert}_{\mathcal{S}}(I, Q)$*, and*
2. *if Q is positive,* $\mathsf{con\text{-}cert}(\mathcal{J}, Q) = \mathsf{scert}_{\mathcal{S}}(I, Q)$. □

That is, an approximate conditional solution is a conditional instance that allows to compute approximate answers for general queries, and exact answers for positive queries.

It is easy to observe that there are settings $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$ and source instances I for which an approximate conditional solution might not exist, even if $\mathcal{S}$ is weakly-acyclic. This is due to the presence of EGDs in Σ_t.

However, for weakly-acyclic settings without EGDs, an approximate conditional solution always exists, and we present a polynomial-time algorithm that is able to construct one. We plan to deal with general weakly-acyclic settings with EGDs in a future work.

The algorithm is a variation of the well-known chase algorithm, which iteratively introduces new facts, starting from a source instance, whenever a TGD is not satisfied, i.e., it triggers the TGD (e.g., see [7]). This variation also allows for a conditional triggering of TGDs, where new atoms are introduced, under the condition that some terms in the body coincide.

Normal TGDs. To simplify the discussion, we consider an extension of TGDs that allow for equality predicates in the body. We will use these TGDs to rewrite standard TGDs in the following normal form. A *normal form TGD* ρ is an expression of the form $\varphi(\mathbf{x}, \mathbf{y}) \wedge \eta(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$, where φ and ψ are conjunctions of atoms, φ uses only variables and each variable in φ occurs once in φ. The formula η is a conjunction of equalities of the form $x = t$, where x is a variable in $\mathbf{x}$ or $\mathbf{y}$, and t is either a variable in $\mathbf{x}$ or $\mathbf{y}$, or a constant. The above equalities denote which variables should be considered to be the same and which positions should contain a constant. A (set of) standard TGDs Σ can be converted in normal form in the obvious way. We denote $\mathsf{norm}(\Sigma)$ as the (set of) TGDs in normal form obtained from Σ.

In the following, fix a conditional instance $\mathcal{I}$, a TGD ρ with $\mathsf{norm}(\rho) = \varphi(\mathbf{x}, \mathbf{y}) \wedge \eta(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$, and a homomorphism h from $\varphi(\mathbf{x}, \mathbf{y})$ to $\mathcal{I}[1]$. We use $h(\eta(\mathbf{x}, \mathbf{y}))$ to denote the condition obtained from $\eta(\mathbf{x}, \mathbf{y})$ by replacing each variable x therein with $h(x)$. Letting $h(\varphi(\mathbf{x}, \mathbf{y})) = \{\alpha_1, \ldots, \alpha_n\}$, we use $\Phi^{\mathcal{I}}_{\rho,h}$ to denote the set of all conditions of the form $h(\eta(\mathbf{x}, \mathbf{y})) \wedge \phi_1 \wedge \cdots \wedge \phi_n$, such that $\langle \alpha_i, \phi_i \rangle \in \mathcal{I}$, for each $i \in \{1, \ldots, n\}$.

Example 6. *Consider the TGD ρ_3 of Example 2. The normal form TGD* $\mathsf{norm}(\rho_3)$ *is* $\mathsf{EmpC}(x,y), \mathsf{EmpC}(x',y'), y = y' \rightarrow \mathsf{SameC}(x,x')$. *Consider now the conditional instance* $\mathcal{I} = \{\langle \mathsf{EmpC}(\mathsf{john}, \mathsf{miami}), \bot_1 = a\rangle, \langle \mathsf{EmpC}(\mathsf{mary}, \bot_2), \mathsf{true}\rangle\}$, *where a is a constant. Then, the mapping* $h = \{x/\mathsf{john}, y/\mathsf{miami}, x'/\mathsf{mary}, y'/\bot_2\}$ *is a homomorphism from* $\{\mathsf{EmpC}(x,y), \mathsf{EmpC}(x',y')\}$ *to* $\mathcal{I}[1]$. *Moreover,* $\Phi^{\mathcal{I}}_{\rho_3,h} = \{\bot_2 = \mathsf{miami} \wedge \bot_1 = a\}$. □

We are now ready to define the notion of conditional chase step. In what follows, for a conditional instance $\mathcal{I}$, a TGD ρ with $\mathsf{norm}(\rho) = \varphi(\mathbf{x},\mathbf{y}) \wedge \eta(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y},\mathbf{z})$ and a homomorphism h from $\varphi(\mathbf{x},\mathbf{y})$ to $\mathcal{I}[1]$, we use $\mathsf{result}(\mathcal{I}, \rho, h)$ to denote the set of atoms obtained from $\mathsf{head}(\mathsf{norm}(\rho))$, where each frontier variable x in $\mathsf{fr}(\mathsf{norm}(\rho))$ is replaced with $h(x)$, and each existential variable z in $\mathsf{exvar}(\mathsf{norm}(\rho))$ is replaced with a fresh null not occurring in $\mathcal{I}$.

Definition 9 (Conditional Chase Step)**.** *Consider a conditional instance $\mathcal{I}$, a TGD ρ, and let* $\mathsf{norm}(\rho) = \varphi(\mathbf{x},\mathbf{y}) \wedge \eta(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y},\mathbf{z})$. *A* conditional chase step *of $\mathcal{I}$ w.r.t. ρ is an expression of the form* $\mathcal{I} \xrightarrow{\rho,h,\phi} \mathcal{J}$, *where* (i) *$h$ is a homomorphism from $\varphi(\mathbf{x},\mathbf{y})$ to $\mathcal{I}[1]$,* (ii) $\phi \in \Phi^{\mathcal{I}}_{\rho,h}$ *is such that* $\phi \not\models \mathsf{false}$, *and* (iii) $\mathcal{J} = \mathcal{I} \cup \{\langle \alpha, \phi\rangle \mid \alpha \in \mathsf{result}(\mathcal{I}, \rho, h)\}$. □

Example 7. *Consider the conditional instance $\mathcal{I}$, the homomorphism h and the TGD ρ_3 of Example 6. Then,* $\mathcal{I} \xrightarrow{\rho_3,h,\phi} \mathcal{J}$ *is a conditional chase step, where ϕ is the condition* $\bot_2 = \mathsf{miami} \wedge \bot_1 = a$, *and* $\mathcal{J} = \mathcal{I} \cup \{\langle \mathsf{SameC}(\mathsf{john}, \mathsf{mary}), \phi\rangle\}$. □

With the notion of conditional chase step at hand, we can define conditional chase sequences, which are sequences of conditional chase steps. For this we need one additional notion. A *conditional tuple* is a pair $\langle \mathbf{t}, \phi\rangle$, where $\mathbf{t}$ is a tuple of constants and nulls, and ϕ a condition. For two conditional tuples $\langle \mathbf{t}, \phi\rangle, \langle \mathbf{u}, \psi\rangle$, with $|\mathbf{t}| = |\mathbf{u}| = n$, we write $\langle \mathbf{t}, \phi\rangle \sqsubseteq \langle \mathbf{u}, \psi\rangle$ if $\phi \models \psi$ and $\phi \models \mathbf{t} = \mathbf{u}$, where $\mathbf{t} = \mathbf{u}$ is a shortand for the condition $\bigwedge_{i=1}^{n} \mathbf{t}[i] = \mathbf{u}[i]$. We write $\langle \mathbf{t}, \phi\rangle \not\sqsubseteq \langle \mathbf{u}, \psi\rangle$, if $\langle \mathbf{t}, \phi\rangle \sqsubseteq \langle \mathbf{u}, \psi\rangle$ does not hold.

Intuitively, $\langle \mathbf{t}, \phi\rangle, \langle \mathbf{u}, \psi\rangle$ should be understood to be two tuples, each of them belonging to a set of "worlds", described by the valuations that satisfy their conditions. Moreover, $\langle \mathbf{t}, \phi\rangle \sqsubseteq \langle \mathbf{u}, \psi\rangle$ means that every world in which $\mathbf{t}$ occurs, is also a world in which $\mathbf{u}$ occurs ($\phi \models \psi$), and in each such world, $\mathbf{t}$ and $\mathbf{u}$ are the same tuples.

Definition 10 (Conditional Chase Sequence)**.** *Consider a TGD-only data exchange setting* $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t\rangle$ *and a source instance I of $\mathcal{S}$. A* conditional chase sequence of I w.r.t. $\mathcal{S}$ *is a (possibly infinite) sequence of conditional instances* $(\mathcal{J}_i)_{i\geq 0}$, *where for each* $i \geq 0$, $\mathcal{J}_i \xrightarrow{\rho_i,h_i,\phi_i} \mathcal{J}_{i+1}$, *and* (i) $\mathcal{J}_0 = \{\langle \alpha, \mathsf{true}\rangle \mid \alpha \in I\}$, (ii) $\rho_i \in \Sigma_{st} \cup \Sigma_t$, *for* $i \geq 0$, *and* (iii) *for every* $j < i$, *if* $\rho = \rho_i = \rho_j$, *then* $\langle h_i(\mathsf{fr}(\rho)), \phi_i\rangle \not\sqsubseteq \langle h_j(\mathsf{fr}(\rho)), \phi_j\rangle$. □

Intuitively, condition *(iii)* of the definition above is required to prevent the chase sequence to apply superfluous steps. An example follows.

Example 8. *Consider the data exchange setting* $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t\rangle$, *with* $\mathsf{S} = \{A/1, B/1\}$, $\mathsf{T} = \{R/2, S/1, T/1\}$, *where the sets* $\Sigma_{st} = \{\rho_1, \rho_2\}$ *and* $\Sigma_t = \{\rho_3\}$ *are such that* $\rho_1 = A(x) \rightarrow \exists z\, R(x,z)$, $\rho_2 = B(x) \rightarrow S(x)$, *and* $\rho_3 = R(x,y), S(y) \rightarrow T(x)$. *Given* $I = \{A(a), B(b_1), B(b_2)\}$, *the following is a conditional chase sequence of I w.r.t. $\mathcal{S}$:*

$$\mathcal{J}_0 = \{\langle A(a), \mathsf{true}\rangle, \langle B(b_1), \mathsf{true}\rangle, \langle B(b_2), \mathsf{true}\rangle\}, \qquad \mathcal{J}_1 = \mathcal{J}_0 \cup \{\langle R(a, \bot), \mathsf{true}\rangle\},$$

$\mathcal{J}_2 = \mathcal{J}_1 \cup \{\langle S(b_1), \mathsf{true}\rangle\}$, $\mathcal{J}_3 = \mathcal{J}_2 \cup \{\langle S(b_2), \mathsf{true}\rangle\}$,
$\mathcal{J}_4 = \mathcal{J}_3 \cup \{\langle T(a), \bot = b_1\rangle\}$, $\mathcal{J}_5 = \mathcal{J}_4 \cup \{\langle T(a), \bot = b_2\rangle\}$.□

For a TGD-only setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t\rangle$ and a source instance I of $\mathcal{S}$, a *finite* conditional chase sequence $(\mathcal{J}_i)_{0\leq i\leq n}$ of I w.r.t. $\mathcal{S}$ is *maximal* if there is no conditional instance $\mathcal{J}_{n+1}$, such that $(\mathcal{J}_i)_{0\leq i\leq n+1}$ is a conditional chase sequence of I w.r.t. $\mathcal{S}$. We call $\mathcal{J}_n$ the *result* of the maximal sequence.

Example 9. *Consider the conditional chase sequence $\mathcal{J}_0, \ldots, \mathcal{J}_5$ of Example 8. The sequence is maximal, since any conditional chase step of the form $\mathcal{J}_5 \xrightarrow{\rho,h,\phi} \mathcal{J}$, for some conditional instance $\mathcal{J}$, cannot satisfy condition* (iii) *of Definition 10. The sequence $\mathcal{J}_0, \ldots, \mathcal{J}_4$ is not maximal because although a conditional atom of the form $\langle T(a), \phi\rangle$ is already present in $\mathcal{J}_4$, an additional conditional atom of the same form needs to be introduced in $\mathcal{J}_5$. This is needed to allow the fact $T(a)$ to be present for two different reasons (either because $\bot = b_1$ or $\bot = b_2$), and both reasons should occur in the result of the sequence.* □

We are now ready to present the main result of this section. In what follows, given a schema S and a conditional instance $\mathcal{I}$, $\mathcal{I}_{|\mathsf{S}}$ denotes the restriction of $\mathcal{I}$ to its conditional facts with relations in S.

Theorem 8. *Consider a TGD-only setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t\rangle$ and a source instance I of $\mathcal{S}$. If $\mathcal{J}$ is the result of a maximal conditional chase sequence of I w.r.t. $\mathcal{S}$, then $\mathcal{J}_{|\mathsf{T}}$ is an approximate conditional solution of I w.r.t. $\mathcal{S}$.*

We can further show that for TGD-only weakly-acyclic settings, a maximal conditional chase sequence always exists, and its length is polynomial. Moreover, its result can be computed in polynomial time.

Theorem 9. *Consider a data exchange setting $\mathcal{S}$ that is TGD-only and weakly-acyclic, and a source instance I of $\mathcal{S}$. Every conditional chase sequence $s = (\mathcal{J}_i)_{0\leq i\leq n}$ of I w.r.t. $\mathcal{S}$ is such that n is a polynomial of $|I|$, and the result $\mathcal{J}_n$ of s can be computed in polynomial time w.r.t. $|I|$.*

Querying Approximate Conditional Solutions. What now remains to show is how we can compute the conditional certain answers over an approximate conditional solution, e.g., obtained via the conditional chase. It is known that the problem of computing the conditional certain answers of a query Q is coNP-hard in general, even when we assume all the conditions in the given conditional instance are true [6]. Hence, given a data exchange setting $\mathcal{S}$ and a source instance I of $\mathcal{S}$, if an approximate conditional instance $\mathcal{J}$ of I w.r.t. $\mathcal{S}$ can be computed in polynomial time w.r.t. $|I|$, one cannot always compute $\mathsf{con\text{-}cert}(\mathcal{J}, Q)$, in polynomial time. Hence, we require an additional step of approximation.

To this end, we exploit an existing algorithm presented in [8] to compute approximate certain answers over incomplete databases. Here we only recall the main properties of the algorithm. For more details, we refer the reder to [8].

For a query Q, the function $\dot{Q}_t$ from conditional instances to sets of tuples is defined in [8], and it is such that the following holds.

Theorem 10 ([8]). *Given a conditional instance $\mathcal{J}$ over some schema S and a query Q over S, then* 1) $\dot{Q}_t(\mathcal{J}) \subseteq \mathsf{con\text{-}cert}(\mathcal{J}, Q)$; 2) *if Q is positive,* $\dot{Q}_t(\mathcal{J}) = \mathsf{con\text{-}cert}(\mathcal{J}, Q)$; 3) *if every condition in $\mathcal{J}$ is a conjunction of equalities, then $\dot{Q}_t(\mathcal{J})$ is computable in polynomial time w.r.t. $|\mathcal{J}|$.*

From the result above, Theorem 9, and Definition 10, we obtain the following crucial result.

Corollary 3. *Consider a TGD-only weakly-acyclic setting $\mathcal{S}$. For every source instance I of $\mathcal{S}$, an approximate conditional solution $\mathcal{J}$ of I w.r.t. $\mathcal{S}$ can be constructed in polynomial time, and for every query Q, $\dot{Q}_t$ is such that* 1) $\dot{Q}_t(\mathcal{J}) \subseteq \mathsf{con\text{-}cert}(\mathcal{J}, Q) \subseteq \mathsf{scert}_{\mathcal{S}}(I, Q)$; 2) *if Q is positive,* $\dot{Q}_t(\mathcal{J}) = \mathsf{con\text{-}cert}(\mathcal{J}, Q) = \mathsf{scert}_{\mathcal{S}}(I, Q)$; 3) $\dot{Q}_t(\mathcal{J})$ *is computable in polynomial time w.r.t. $|\mathcal{J}|$.*

6. Connections with Other Work and Next Steps

Conditional instances and, more in general, incomplete databases, have already been employed in the context of data exchange. However, in most of previous work, incomplete databases are used to encode source and target instances with incomplete information. In [9], the authors extend the standard data exchange framework by allowing source and target instances to be incomplete databases, encoded via some representation system, such as conditional instances. There, the main goal is to study the semantics of data exchange under the assumption that the source and target instances can be incomplete. In contrast, in our work, we focus on the classical data exchange setting, where source and target instances are standard (complete) databases. Here we employ incomplete databases, in particular conditional instances, only as a *tool* to compute the (approximate) certain answers of a query over our set of supported solutions, which are standard databases as well.

In Section 5 we have seen how a conditional extension of the chase procedure can be employed to compute in polynomial time, for weakly-acyclic settings, an approximate conditional solution. The idea of extending the chase procedure with conditional TGD applications is not new and has been explored in previous work. In particular, the work of [10] introduces a conditional version of the chase procedure which is similar to ours. The main difference is that the conditional chase of [10] is much simpler, since it is an extension of the simplest variant of the chase algorithm, called oblivious chase, while ours can be seen as an extension of the more refined semi-oblivious (a.k.a. skolem) chase (see., e.g., [11, 12, 13, 14, 15] for more details). For this reason, it is not difficult to show that when considering weakly-acyclic settings, the conditional chase of [10] is not guaranteed to terminate, while termination for weakly-acyclic settings is a crucial property for our purposes, since we need to be able to construct a finite conditional instance in this case.

The problem of dealing with non-monotonic queries has been investigated beyond data exchange, as for example for Ontology-Mediated Query Answering (OMQA). In this setting, we are given an instance (database) D, an ontology Σ encoded in some logical formalism (e.g., via TGDs), and a query $Q(\mathbf{x})$, and the goal is to compute all the certain answers of $Q(\mathbf{x})$ w.r.t. D and Σ, i.e., the tuples that are answers to Q in *every model* of the logical theory $D \cup \Sigma$. A relevant work in this scenario is the one in [16], where the authors define the query language EQL-Lite($\mathcal{Q}$), parametrized with a standard (positive) query language $\mathcal{Q}$ (e.g., UCQs), and supports a limited form of negation. In particular, an expression ψ in EQL-Lite($\mathcal{Q}$) is of the form $\psi := \mathbb{K}\,\rho \mid \psi_1 \wedge \psi_2 \mid \neg\psi_1 \mid \exists x\, \psi_1$, where $\rho \in \mathcal{Q}$, and ψ_1, ψ_2 are EQL-Lite($\mathcal{Q}$) expressions.

Here, the epistemic operator $\mathbb{K}$ is applied to expressions $\rho \in \mathcal{Q}$ and returns the certain answers of ρ w.r.t. the input database D and the ontology Σ. The main instantiation of EQL-Lite that the authors study is EQL-Lite(UCQ), i.e., where $\mathcal{Q}$ coincides with the set of all UCQs.

From the above definition, we observe that negation is applied only to (a combination of) the certain answers of positive queries. This gives a semantics to negation that fundamentally differs from ours, as illustrated in the following example.

Consider the data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, where S stores employees of a company in the unary relation Emp. The target schema T contains a unary relation Emp' storing employees, the ternary relation Addr assigning to each employee her work and home address, and the unary relation $\mathsf{WorkFromHome}$, storing employees working from home. Assume we have $\Sigma_{st} = \{\rho_1 = \mathsf{Emp}(x) \rightarrow \mathsf{Emp}'(x), \rho_2 = \mathsf{Emp}(x) \rightarrow \exists z \, \exists w \, \mathsf{Addr}(x, z, w)\}$ and $\Sigma_t = \{\rho_3 = \mathsf{Addr}(x, y, y) \rightarrow \mathsf{WorkFromHome}(x)\}$.

The above setting copies employees from the source to the target via the TGD ρ_1, while the TGD ρ_2 states that each employee must have a work and home address, denoted via the existential variables z and w, respectively. Finally, the TGD ρ_3 states that if the work and home address of an employee coincide, then this employee works from home.

Assume the source instance is $I = \{\mathsf{Emp}(\mathsf{john})\}$, and let Q be the query asking for all employees who do not work from home, i.e., $Q(x) = \mathsf{Emp}'(x) \wedge \neg \mathsf{WorkFromHome}(x)$.

According to [16], the query Q corresponds to the EQL-Lite(UCQ) expression $Q'(x) = \mathbb{K}\,\mathsf{Emp}'(x) \wedge \neg \mathbb{K}\,\mathsf{WorkFromHome}(x)$. Letting $D = I$, and $\Sigma = \Sigma_{st} \cup \Sigma_t$, roughly, the above means that an empolyee is an answer to the query Q' if she is present in *all* models of $D \cup \Sigma$ and such that there is *at least one model* in which the employee does not work from home. Under this interpretation, the answer to Q' is john. However, under our semantics, the answer to Q is empty. Hence, the fundamental difference is that negation, under EQL-Lite, is interpreted as negating classical certain answering, and thus an expression $\neg \mathbb{K}\, \psi$ is "satisfied" when at least one model/solution does not entail ψ, while in our case, we consider the given query as a whole, and require it to be satisfied in *every* valid solution.

We conclude by discussing avenues for further research. First, we would like to extend the conditional chase to weakly-acyclic settings with EGDs, and identify relevant data exchange settings for which computing the supported certain answers is tractable. Moreover, we plan to experimentally evaluate the current approximation approaches both in terms of quality and of running time via a dedicated benchmark, as did e.g., in the context of approximate consistent query answering [17]. Since explaining query answering has recently drawn considerably attention under existential rule languages (e.g., see [18, 19, 20, 21, 22]), and knowledge representation in general (e.g., in the context of argumentation [23]) an interesting direction for future work is to address such issue in our setting. Also, it would be interesting to account for user preferences when answering queries, as recently done in [24].

References

[1] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: semantics and query answering, TCS 336 (2005) 89–124.

[2] A. Hernich, L. Libkin, N. Schweikardt, Closed world data exchange, TODS 36 (2011) 14:1–14:40.
[3] A. Hernich, Answering Non-Monotonic Queries in Relational Data Exchange, LMCS Volume 7, Issue 3 (2011).
[4] L. Libkin, C. Sirangelo, Data exchange and schema mappings in open and closed worlds, JCSS 77 (2011) 542–571.
[5] P. G. Kolaitis, J. Panttaja, W. C. Tan, The complexity of data exchange, in: PODS, 2006, pp. 30–39.
[6] T. Imielinski, W. Lipski, Incomplete information in relational databases, J. ACM 31 (1984) 761–791.
[7] E. Tsamoura, D. Carral, E. Malizia, J. Urbani, Materializing knowledge bases via trigger graphs, Proc. VLDB Endow. 14 (2021) 943–956.
[8] S. Greco, C. Molinaro, I. Trubitsyna, Approximation algorithms for querying incomplete databases, Inf. Syst. 86 (2019) 28–45.
[9] M. Arenas, J. Pérez, J. L. Reutter, Data exchange beyond complete data, J. ACM 60 (2013) 28:1–28:59.
[10] G. Grahne, A. Onet, On conditional chase termination, in: AMW, 2011.
[11] M. Calautti, G. Gottlob, A. Pieris, Chase termination for guarded existential rules, in: PODS, 2015, pp. 91–103.
[12] G. Grahne, A. Onet, Anatomy of the chase, Fundam. Inform. 157 (2018) 221–270.
[13] M. Calautti, A. Pieris, Oblivious chase termination: The sticky case, in: ICDT, 2019, pp. 17:1–17:18.
[14] M. Calautti, A. Pieris, Semi-oblivious chase termination: The sticky case, Theory Comput. Syst. 65 (2021) 84–121.
[15] M. Calautti, G. Gottlob, A. Pieris, Non-uniformly terminating chase: Size and complexity, in: PODS, 2022, pp. 369–378.
[16] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Eql-lite: Effective first-order query processing in description logics., in: IJCAI, 2007, pp. 274–279.
[17] M. Calautti, M. Console, A. Pieris, Benchmarking approximate consistent query answering, in: L. Libkin, R. Pichler, P. Guagliardo (Eds.), PODS, 2021, pp. 233–246.
[18] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for negative query answers under inconsistency-tolerant semantics, in: Proc. IJCAI, 2022, pp. 2705–2711.
[19] İ. İ. Ceylan, T. Lukasiewicz, E. Malizia, C. Molinaro, A. Vaicenavicius, Preferred explanations for ontology-mediated queries under existential rules, in: Proc. AAAI, 2021, pp. 6262–6270.
[20] İ. İ. Ceylan, T. Lukasiewicz, E. Malizia, C. Molinaro, A. Vaicenavicius, Explanations for negative query answers under existential rules, in: Proc. KR, 2020, pp. 223–232.
[21] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for inconsistency-tolerant query answering under existential rules, in: Proc. AAAI, 2020, pp. 2909–2916.
[22] İ. İ. Ceylan, T. Lukasiewicz, E. Malizia, A. Vaicenavicius, Explanations for query answers under existential rules, in: Proc. IJCAI, 2019, pp. 1639–1646.
[23] G. Alfano, M. Calautti, S. Greco, F. Parisi, I. Trubitsyna, Explainable acceptance in probabilistic abstract argumentation: Complexity and approximation, in: KR, 2020, pp. 33–43.
[24] M. Calautti, S. Greco, C. Molinaro, I. Trubitsyna, Preference-based inconsistency-tolerant query answering under existential rules, Artif. Intell. 312 (2022) 103772.