Querying Data Exchange Settings Beyond Positive Queries

Marco Calautti[1], Sergio Greco[2], Hebatalla Hammad[3], Tariq Mahmood[2], Cristian Molinaro[2] and Irina Trubitsyna[2]

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

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, 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. 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.

By now, the *certain answers* semantics is the most accepted one for answering queries. 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 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.

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy
marco.calautti@unimi.it (M. Calautti); greco@dimes.unical.it (S. Greco); hebatalla.hammad@unitn.it (H. Hammad); mahmood.tariq@dimes.unical.it (T. Mahmood); c.molinaro@dimes.unical.it (C. Molinaro); trubitsyna@dimes.unical.it (I. Trubitsyna)
© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3478/paper31.pdf

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.

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* (john, 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.

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.

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 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

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

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 now 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 :	$\mathsf{EXISTS\text{-}SSOL}(\mathcal{S})$
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* $\mathsf{EXISTS\text{-}SSOL}(\mathcal{S})$ *is undecidable.*

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 refer to [1] for more details on the definition of weak-acyclicity.

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

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] and can be accomplished via the well-known chase procedure (e.g., see [5]). 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. Next Steps

For future work, it would be interesting to see if the good complexity guarantees we obtain for weakly-acyclic dependencies are preserved when considering more complex acyclicity conditions (e.g., see [6, 7, 8]), thus enlarging the applicability of our semantics. Moreover, we would like to experimentally evaluate our techniques by means of a carefully designed benchmark in the spirit of other efforts such as the one of [9]. Since explaining query answering has recently drawn considerably attention under existential rule languages (e.g., see [10, 11, 12, 13, 14, 15]), and knowledge representation in general (e.g., in the context of argumentation [16]) 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 [17, 18], possibly considering other ways of expressing preferences, e.g. by means of CP-nets [19, 20].

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] E. Tsamoura, D. Carral, E. Malizia, J. Urbani, Materializing knowledge bases via trigger graphs, Proc. VLDB Endow. 14 (2021) 943–956.

[6] B. C. Grau, I. Horrocks, M. Krötzsch, C. Kupke, D. Magka, B. Motik, Z. Wang, Acyclicity notions for existential rules and their application to query answering in ontologies, J. Artif. Intell. Res. 47 (2013) 741–808.

[7] M. Calautti, G. Gottlob, A. Pieris, Non-uniformly terminating chase: Size and complexity, in: PODS, 2022, pp. 369–378.

[8] M. Calautti, A. Pieris, Semi-oblivious chase termination: The sticky case, Theory Comput. Syst. 65 (2021) 84–121.

[9] M. Calautti, M. Console, A. Pieris, Benchmarking approximate consistent query answering, in: L. Libkin, R. Pichler, P. Guagliardo (Eds.), PODS, 2021, pp. 233–246.

[10] T. Lukasiewicz, E. Malizia, M. V. Martinez, C. Molinaro, A. Pieris, G. Simari, Inconsistency-tolerant query answering for existential rules, Artif. Intell. 307 (2022) 103685.

[11] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for negative query answers under inconsistency-tolerant semantics, in: Proc. IJCAI, 2022, pp. 2705–2711.

[12] 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.

[13] 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.

[14] T. Lukasiewicz, E. Malizia, C. Molinaro, Explanations for inconsistency-tolerant query answering under existential rules, in: Proc. AAAI, 2020, pp. 2909–2916.

[15] İ. İ. Ceylan, T. Lukasiewicz, E. Malizia, A. Vaicenavicius, Explanations for query answers under existential rules, in: Proc. IJCAI, 2019, pp. 1639–1646.

[16] 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.

[17] M. Calautti, L. Caroprese, S. Greco, C. Molinaro, I. Trubitsyna, E. Zumpano, Existential active integrity constraints, Expert Syst. Appl. 168 (2021) 114297.

[18] M. Calautti, S. Greco, C. Molinaro, I. Trubitsyna, Preference-based inconsistency-tolerant query answering under existential rules, Artif. Intell. 312 (2022) 103772.

[19] T. Lukasiewicz, E. Malizia, Complexity results for preference aggregation over (m)cp-nets: Pareto and majority voting, Artif. Intell. 272 (2019) 101–142.

[20] T. Lukasiewicz, E. Malizia, Complexity results for preference aggregation over (m)cp-nets: Max and rank voting, Artif. Intell. 303 (2022) 103636.