TPLP: Page 1–29.

doi:10.1017/S1471068423000339

Querying Data Exchange Settings Beyond Positive Queries

MARCO CALAUTTI
DI, University of Milan, Milan, Italy
(*e-mail:* marco.calautti@unimi.it)

SERGIO GRECO, CRISTIAN MOLINARO and IRINA TRUBITSYNA
DIMES, University of Calabria, Arcavacata, Italy
(*e-mails:* greco@dimes.unical.it, cmolinaro@dimes.unical.it, trubitsyna@dimes.unical.it)

submitted 6 February 2023; revised 9 July 2023; accepted 25 July 2023

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 work, but little attention has been paid to more general queries. A few proposals of semantics 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, for example, 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 exact answers under our semantics can be computed by means of logic programs with choice, thus exploiting existing efficient systems. For more efficient computations, we also show that our semantics allows for the construction of a representative target instance, similar in spirit to a universal solution, that can be exploited for computing approximate answers in polynomial time.

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 (Fagin *et al.* 2005). Although it has been formally shown that for positive queries (e.g., conjunctive queries) the notion of solution of (Fagin *et al.* 2005) 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 product orders in a binary relation Ord, with the first argument being the id of an order, and the second argument specifying whether the order has been paid. Moreover, T is the target schema having unary relations AllOrd and Paid, storing all orders and the paid orders, respectively. The schema mapping is described by the following 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 first-order (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 (Fagin *et al.* 2005), 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 (Hernich *et al.* 2011) and (Hernich 2011). In both approaches, the certain answers in the example above are $\{2\}$. However, the works above have their own drawbacks too. In (Hernich *et al.* 2011), 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.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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):

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

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 consider the query Q that 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 (Hernich 2011), 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 (Libkin and Sirangelo 2011), 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 might be the case that the source is not always available, and thus the materialization of a single solution, over which certain answers can be computed, is a desirable requirement. This is especially true when using weakly-acyclic dependencies, which form the standard language for data exchange (Fagin *et al.* 2005). However, none of the semantics above allow for the materialization of such a special solution, for weakly-acyclic settings.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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 it becomes coNP-complete when we focus on weakly-acyclic dependencies.

Moreover, we show that exact answers under supported solutions for *general* queries in weakly-acyclic settings can be computed via an encoding into logic programming with the well-known choice construct, allowing one to use efficient off-the-shelf reasoning systems.

Finally, we also show that if one is not willing to incur the high complexity of exact certain answers for weakly-acyclic settings, then it is possible to construct a target instance in *polynomial time*, which is similar in spirit to a universal solution of (Fagin *et al*. 2005), 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, and Null of *constants*, *variables*, and *labeled nulls*, respectively. 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 a 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 TGD ρ (over a schema S) is a FO 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 EGD η (over a schema S) is a FO 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

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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 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* (UCQs) 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 refer to UCQs also 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, that is, 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 (Fagin *et al.* 2005). 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' (Fagin *et al.* 2005). Letting $Q(J)_{\downarrow} = Q(J) \cap \mathsf{Const}^{|\mathbf{x}|}$, for any instance J and query $Q(\mathbf{x})$, the following result from (Fagin *et al.* 2005) is well-known:

Theorem 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.

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

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

classical solutions is that such solutions are too permissive, that is, 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 by replacing each TGD ρ in Σ 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

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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

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 $\mathcal{S}$, it holds that:

1. $\mathsf{ssol}(I,\mathcal{S}) \subseteq \mathsf{sol}(I,\mathcal{S})$, and
2. $\mathsf{ssol}(I,\mathcal{S}) = \emptyset$ iff $\mathsf{sol}(I,\mathcal{S}) = \emptyset$.

Proof
Item 1 follows by definition. For proving Item 2, it suffices to show that $\mathsf{sol}(I,\mathcal{S}) \neq \emptyset$ implies $\mathsf{ssol}(I,\mathcal{S}) \neq \emptyset$. Let $\mathcal{S} = \langle \mathsf{S},\mathsf{T},\Sigma_{st},\Sigma_t\rangle$ and consider a solution $J \in \mathsf{sol}(I,\mathcal{S})$. We construct from J a supported solution $\hat{J}$ in $\mathsf{ssol}(I,\mathcal{S})$. Let J' be one of the minimal subsets of J such that J' is still a solution of $\mathsf{sol}(I,\mathcal{S})$. Moreover, let $\hat{J}$ be the instance obtained from J', where each null $\perp$ in J' is replaced with a new constant $c_\perp$ not occurring in $\Sigma_{st} \cup \Sigma_t$ and J'. Since $\hat{J}$ and J' are the same instance, up to null renaming, we conclude that $\hat{J}$ is also a solution in $\mathsf{sol}(I,\mathcal{S})$. To see that $\hat{J}$ is a supported solution, consider the following ex-choice γ. For every TGD $\rho \in \Sigma_{st} \cup \Sigma_t$, and every tuple $\mathbf{t}$ of constants such that there exists a homomorphism h from $\mathsf{body}(\rho)$ to $\hat{J}$, and $\mathbf{t} = h(\mathsf{fr}(\rho))$, let $\gamma(\rho,\mathbf{t}) = \{(z,h(z)) \mid z \in \mathsf{exvar}(\rho)\}$. By construction of γ, $I \cup \hat{J}$ satisfies Σ_{st}^γ, and $\hat{J}$

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

satisfies Σ_t^γ. Since $\hat{J}$ is minimal, that is, for every $J'' \subsetneq \hat{J}$, $J'' \notin \mathsf{sol}(I, \mathcal{S})$, from Item 1 of this claim, every $J'' \subsetneq J$ is such that $J'' \notin \mathsf{ssol}(I, \mathcal{S})$, that is, either $I \cup J''$ does not satisfy Σ_{st}^γ or J'' does not satisfy Σ_t^γ. Thus, $\hat{J}$ is a supported solution of $\mathsf{ssol}(I, \mathcal{S})$, and the claim follows. □

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 $\mathcal{S}$, $\mathsf{scert}_\mathcal{S}(I, Q) = \mathsf{cert}_\mathcal{S}(I, Q)$.

Proof
The fact that $\mathsf{cert}_\mathcal{S}(I, Q) \subseteq \mathsf{scert}_\mathcal{S}(I, Q)$, follows from Item 1 of Theorem 2. To prove that $\mathsf{scert}_\mathcal{S}(I, Q) \subseteq \mathsf{cert}_\mathcal{S}(I, Q)$, assume $\mathbf{t} \notin \mathsf{cert}_\mathcal{S}(I, Q)$, which means that there exists a solution J of I w.r.t. $\mathcal{S}$ such that $\mathbf{t} \notin Q(J)$. Since Q is positive, and hence monotone, $\mathbf{t} \notin Q(J)$ iff $\mathbf{t} \notin Q(J')$, where J' is one of the minimal subsets of J such that J' is still a solution of I w.r.t. $\mathcal{S}$. Let $\hat{J}$ be the instance obtained from J', where each null $\perp$ in J' is replaced with a new constant $c_\perp$ not occurring in $\mathbf{t}$, Q, $\Sigma_{st} \cup \Sigma_t$, and J'. With a similar discussion to the one given in the proof of Theorem 2, we conclude that $\hat{J}$ is a supported solution of I w.r.t. $\mathcal{S}$. Since Q is positive, and since $\mathbf{t}$ and Q do not contain any of the constants introduced in J', we conclude that $\mathbf{t} \notin Q(\hat{J})$, which implies that $\mathbf{t} \notin \mathsf{scert}_\mathcal{S}(I, Q)$, and the claim follows. □

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 Theorems 1 and 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, that is, 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

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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 transferred 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 (Kolaitis *et al.* 2006). □

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 (Fagin *et al.* 2005), 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 (Fagin et al. 2005)*)
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.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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 (Fagin *et al.* 2005, 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.

Proof
We provide a polynomial-time reduction from the *Embedding Problem for Finite Semigroups* EMB *(Kolaitis et al. 2006)*. The reduction is an adaptation of the one used for proving Proposition 6.1 in (Hernich *et al.* 2011). Inputs of EMB are pairs of the form A, f, where A is a finite set, and f is a partial function of the form $f : A \times A \to A$. The question is whether there exists a finite set $B \supseteq A$, and a total function $g : B \times B \to B$, such that g is associative,[2] and g extends f, that is, whenever $f(a, b)$ is defined, $g(a, b) = f(a, b)$.

Let us first introduce some notation. Consider a finite set A and a partial function $f : A \times A \to A$. We define the instance:

$$I_{A,f} = \{\mathsf{F}(a, b, c) \mid a, b, c \in A \text{ and } f(a, b) = c\}.$$

Consider now the data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, where $\mathsf{S} = \{\mathsf{F}/3\}$ and $\mathsf{T} = \{\mathsf{G}/3\}$. Intuitively, the relation F collects all the triples a, b, c such that $f(a, b) = c$, whereas the relation G collects all the triples of the extended associative function g. The sets Σ_{st} and Σ_t are defined as $\Sigma_{st} = \{\mathsf{F}(x, y, z) \to \mathsf{G}(x, y, z)\}$ and $\Sigma_t = \{\mathsf{G}(x, y, z) \to \exists x', y', z' \ \mathsf{G}(x', y', z') \wedge \mathsf{Aux}(x, y, z)\}$. Roughly Σ_{st} is in charge of forcing the function stored in G to be an extension of the function stored in F, whereas Σ_t is in charge of adding additional entries to G.

The difference with the construction of (Hernich *et al.* 2011) is in the set Σ_t. Here, the head of the only TGD in Σ_t has an additional auxiliary atom $\mathsf{Aux}(x, y, z)$. Intuitively, since the set Σ_t is in charge of extending the function defined by the relation F by introducing additional terms, in order for these terms to be actually introduced in a supported

[2] A total function $g : B \times B \to B$ is associative if for every $a, b, c \in B$, $g(g(a, b), c) = g(a, g(b, c))$.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

solution, we require that every body variable is also a frontier variable. Regarding our query Q, it is the same as the one in (Hernich *et al*. 2011). Hence, instead of giving the precise expression of Q, we only describe its properties. The query Q over $\mathsf{T} = \{\mathsf{G}/3\}$ is a Boolean query which is true (i.e., the empty tuple is its only output) if either G does not encode a function, that is, it maps the same pair (a, b) to different terms, or G does not encode an associative function, or G does not encode a total function. In other words, Q checks whether G does not encode a solution for EMB.

We are now ready to present the reduction. Let A be a finite set and $f : A \times A \to A$ be a partial function. The reduction constructs the source instance $I_{A,f}$ and the empty tuple $\mathbf{t} = ()$. Clearly, $I_{A,f}$ can be constructed in polynomial time w.r.t. $|I|$. It remains to show that A, f is a "yes"-instance of EMB iff $\mathbf{t} \notin \mathsf{scert}_{\mathcal{S}}(I_{A,f}, Q)$.

(Only if direction) Assume $\mathbf{t} \notin \mathsf{scert}_{\mathcal{S}}(I_{A,f}, Q)$. Then, there exists a supported solution $J \in \mathsf{ssol}(I_{A,f}, \mathcal{S})$ of $I_{A,f}$ w.r.t. $\mathcal{S}$ such that $\mathbf{t} \notin Q(J)$. By definition of supported solution, J is finite and it only contains atoms with relation G. Thus, by definition of $\mathcal{S}$, $\mathbf{t} \notin Q(J)$ implies that J necessarily encodes an extension of f, which is also total and associative.

(If direction) Assume A, f is a "yes"-instance of EMB, and let $B \supseteq A$ be a finite set, and $g : B \times B \to B$ be the total associative function that extends f. Then, consider the instance J over T defined as $J = \{\mathsf{G}(a, b, c) \mid a, b, c \in B \text{ and } g(a, b) = c\}$. It is not difficult to verify that J is a supported solution of $I_{A,f}$ w.r.t. $\mathcal{S}$. Finally, by construction of J, $\mathbf{t} \notin Q(J)$ as needed. □

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. In particular, we prove that in this case, the problem is in coNP and that this complexity bound is tight (i.e., there exist weakly-acyclic settings and queries for which the problem is coNP-hard). We first focus on the upper bound.

Theorem 7
For every weakly-acyclic setting $\mathcal{S}$ and every query Q, $\mathsf{SCERT}(\mathcal{S}, Q)$ is in coNP.

Proof
We provide a non-deterministic polynomial-time procedure for solving the complement of the problem $\mathsf{SCERT}(\mathcal{S}, Q)$, when $\mathcal{S}$ is a weakly-acyclic data exchange setting. That is, given a source instance I of $\mathcal{S}$ and a tuple $\mathbf{t} \in \mathsf{Const}^{ar(Q)}$, the procedure non-deterministically constructs a supported solution J^* of I w.r.t. $\mathcal{S}$ (if one exists), and checks whether $\mathbf{t} \notin Q(J^*)$. Let $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, and consider a source instance I of $\mathcal{S}$, a query Q over T, and a tuple $\mathbf{t} \in \mathsf{Const}^{ar(Q)}$.

The procedure is defined in two parts. The first part is in charge of non-deterministically constructing a supported solution J^*. If the procedure was not able to construct a supported solution (i.e., no such solution exists, or it followed a wrong computation path), the procedure sets $J^* = \bot$. The second part simply verifies whether either $J^* = \bot$, in which case it rejects, or it checks whether $\mathbf{t} \notin Q(J^*)$, in which case accepts, otherwise rejects. The second part can be easily implemented by a deterministic polynomial-time procedure; we now show the first procedure constructing J^*.

This procedure implements a variation of the so-called semi-oblivious chase algorithm; we refer the reader to (Marnette 2009) for more details. In the following, for each TGD $\rho \in \Sigma_{st} \cup \Sigma_t$, let Chosen_ρ be a fresh relation, not occurring in $\mathsf{S} \cup \mathsf{T}$, of arity $|\mathsf{fr}(\rho)|$.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

1. Let $J_0 = I$, and let the current step be $i = 0$.
2. If J_i does not satisfy the EGDs in Σ_t, then let $J^* = \bot$ and halt;
3. If J_i satisfies the EGDs in Σ_t, and no TGD $\rho \in \Sigma_{st} \cup \Sigma_t$ and homomorphism h from $\mathsf{body}(\rho)$ to J_i exist such that $\mathsf{Chosen}_\rho(h(\mathsf{fr}(\rho))) \notin J_i$, then let J^* be J_i after removing all atoms over S and the atoms using the Chosen predicates, and halt.
4. Otherwise, guess a TGD $\rho_i \in \Sigma_{st} \cup \Sigma_t$ and a homomorphism h_i from $\mathsf{body}(\rho_i)$ to J_i such that $\mathsf{Chosen}_{\rho_i}(h_i(\mathsf{fr}(\rho_i))) \notin J_i$, and guess an extension h'_i of h_i such that, for each $z \in \mathsf{exvar}(\rho_i)$, $h'_i(z) = c^i_z$, where either c^i_z is a constant occurring in one of $\mathcal{S}$, I, Q, or a fresh new constant. Finally, let $J_{i+1} = J_i \cup h'_i(\mathsf{head}(\rho_i)) \cup \{\mathsf{Chosen}_{\rho_i}(h_i(\mathsf{fr}(\rho_i)))\}$. Let $i := i + 1$ and goto 2.

To show that the procedure above terminates after a polynomial number of steps, we can use a similar argument to the one given for proving Theorem 3.9 in (Fagin *et al.* 2005). We now show that, for every target instance J of $\mathcal{S}$, a run of the above procedure halting with $J^* = J$ exists iff J is a supported solution of I w.r.t. $\mathcal{S}$, and the claim will follow. We focus on one of the two directions, as the other direction can be proved in a similar way.

Assume there is a run of n steps of the procedure above with $J^* = J$, for some target instance J of $\mathcal{S}$, and let ρ_i, h_i, and c^i_z, for $z \in \mathsf{exvar}(\rho_i)$ be the TGD, homomorphism, and constants guessed at step i in the run. Let γ be the ex-choice such that, for each $i \in \{1, \ldots, n\}$, $\gamma(\rho_i, h_i(\mathsf{fr}(\rho_i))) = \{(z, c^i_z) \mid z \in \mathsf{exvar}(\rho_i)\}$. The fact that γ is indeed an ex-choice follows from the fact that at each step $i \in \{1, \ldots, n\}$, a constant c^i_z is introduced only if $\mathsf{Chosen}_{\rho_i}(h_i(\mathsf{fr}(\rho_i))) \notin J_i$, which in turn implies that no constant has been chosen at some step $j < i$, where $h_j(\mathsf{fr}(\rho_j)) = h_i(\mathsf{fr}(\rho_i))$. By definition of the procedure, J is the instance obtained from J_n where all the atoms with relations in S or of the form Chosen_ρ are removed. Hence, by construction, $I \cup J$ satisfies Σ^γ_{st} and J satisfies all the TGDs in Σ^γ_t. Since $J \neq \bot$, J also satisfies the EGDs in Σ^γ_t. Moreover, no $J' \subsetneq J$ is such that $I \cup J'$ satisfies Σ^γ_{st} and J' satisfies Σ^γ_t. If this is the case, let $\alpha \in J \setminus J'$, and let $i \in \{1, \ldots, n\}$ be the step in the above run where α is added in J_{i+1}. Then, the TGD $\rho' = h_i(\rho_i) \rightarrow h'_i(\mathsf{head}(\rho_i))$ is in $\Sigma^\gamma_{st} \cup \Sigma^\gamma_t$, by construction of γ. However J' does not satisfy ρ'. The latter, together with the previous discussion implies that J is a supported solution of I w.r.t. $\mathcal{S}$. □

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 (Hernich *et al.* 2011; Hernich 2011).

We now move to the lower bound and show that the coNP upper bound is tight.

Theorem 8
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.

Proof
The coNP-hardness is proved via a reduction from 3-colorability to the complement of our problem. We encode the input graph $G = (V, E)$ as the instance

$$I_G = \{\mathsf{V}(u) \mid u \in V\} \cup \{\mathsf{E_s}(u, v) \mid (u, v) \in E\} \cup \{\mathsf{Col}(c) \mid c \in \{\mathsf{r}, \mathsf{g}, \mathsf{b}\}\}.$$

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

Colorings are constructed in the setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, via the source-to-target TGDs (Σ_t is empty):

$$\rho_1 = \mathsf{Col}(x) \rightarrow \mathsf{Col_t}(x),$$
$$\rho_2 = \mathsf{E_s}(x, y) \rightarrow \mathsf{E_t}(x, y),$$
$$\rho_3 = \mathsf{V}(x) \rightarrow \exists z\, \mathsf{HasC}(x, z),$$

where $\mathsf{Col_t}$ collects all colors, $\mathsf{E_t}$ contains the edges of the graph in the target schema, and HasC assigns a term to each node of the graph.

The Boolean query $Q = Q_1 \vee Q_2$ is true over an instance of the target schema iff the instance does not encode a valid 3-coloring. In particular, Q_1 checks whether the "color" used for some node differs from $\mathsf{r}, \mathsf{g}, \mathsf{b}$:

$$Q_1 = \exists x, y\, \mathsf{HasC}(x, y) \wedge \neg \mathsf{Col_t}(y),$$

whereas Q_2 checks whether the nodes of an edge have the same color:

$$Q_2 = \exists x, y, z\, \mathsf{E_t}(x, y) \wedge \mathsf{HasC}(x, z) \wedge \mathsf{HasC}(y, z).$$

We prove that G admits a 3-coloring iff $\mathbf{t} = () \notin \mathsf{scert}_{\mathcal{S}}(I_G, Q)$.

(Only if direction) Assume G admits a 3-coloring μ and consider the instance

$$J = \quad \{\mathsf{HasC}(v, \mu(v)) \mid v \in V\} \cup \{\mathsf{E_t}(u, v) \mid (u, v) \in E\} \cup \{\mathsf{Col_t}(c) \mid c \in \{\mathsf{r}, \mathsf{g}, \mathsf{b}\}\}.$$

It is not difficult to see that J is a supported solution of I_G w.r.t. $\mathcal{S}$. Clearly, $\mathbf{t} \notin Q(J)$ and the claim follows.

(If direction) Assume that G does not admit a 3-coloring, and consider an arbitrary supported solution J of I_G w.r.t. $\mathcal{S}$. Note that for every edge $(u, v) \in E$, we have that $\mathsf{E_t}(u, v) \in J$ and $\mathsf{HasC}(u, c_1), \mathsf{HasC}(v, c_2) \in J$, for some constants c_1, c_2. We distinguish two cases. Assume that there is an edge $(u, v) \in E$ such that $c_1 \notin \{\mathsf{r}, \mathsf{g}, \mathsf{b}\}$ or $c_2 \notin \{\mathsf{r}, \mathsf{g}, \mathsf{b}\}$. Thus, $\mathbf{t} \in Q_1(J)$ which implies $\mathbf{t} \in Q(J)$. Assume now that for every edge $(u, v) \in E$, $c_1, c_2 \in \{\mathsf{r}, \mathsf{g}, \mathsf{b}\}$. Thus, since G does not admit a 3-coloring, for at least one edge $(u, v) \in E$, $c_1 = c_2$. Hence, $\mathbf{t} \in Q_2(J)$, which implies that $\mathbf{t} \in Q(J)$ and the claim follows. □

We point out that the query employed in the proof of the above theorem is a simple Boolean combination of CQs. This kind of FO queries have been studied in the context of incomplete databases, for example, see (Gheerbrant and Libkin 2015). However, differently from the incomplete databases setting, where such queries guarantee query answering in polynomial time, the complexity in our setting is higher, due to the presence of TGDs; the latter is true even for weakly-acyclic TGDs, as shown by Theorem 8 above. Similarly, arbitrary FO queries (e.g., involving also universal quantification) behave very differently depending on the given setting. For example, according to Theorem 7, for *any* FO query, supported certain answers remain in coNP, under weakly-acyclic settings, while for arbitrary settings, the use of universal quantification makes supported certain answering undecidable; the latter is a consequence of the proof of Theorem 6. Hence, one cannot directly conclude much on the complexity of supported certain answers by considering the query alone, as done for querying incomplete databases.

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 (Fagin *et al.* 2005). Hence, the result below follows.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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

5 Exact query answering via logic programming

In this section, we show how to compute supported certain answers exactly by means of a translation into logic programming under the stable model semantics, that is, *answer set programming* (ASP). First, we need to recall the syntax and semantics of logic programs. In particular, we focus on a fragment of logic programs that is enough for our purposes, which is Datalog with (possibly non-stratified) negation, which means we do not allow for function symbols or disjunctive rules.

Syntax. A *literal* L is an expression of the form α or $\neg\alpha$, where α is either an atom without nulls, or the expression $t_1 = t_2$, where t_1, t_2 are variables or constants; we write $t_1 \neq t_2$ for $\neg t_1 = t_2$. We say that L is *positive* (resp., *negative*) if $L = \alpha$ (resp., $L = \neg\alpha$). If a literal contains no variables, it is said to be *ground*.

A *rule* r is an expression of the form

$$H \text{ :- } A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m.$$

with $n \geq 0$, $m \geq 0$, and where H is either a positive literal or the symbol $\bot$, $A_1, \ldots, A_n$ are positive literals, and $\neg B_1, \ldots, \neg B_m$ are negative literals. We denote $\mathsf{head}(r) = \{H\}$ as the *head* of r, while $\mathsf{body}(r) = \{A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m\}$ is the *body* of r; we use $\mathsf{body}^+(r)$ to denote $\{A_1, \ldots, A_n\}$, and $\mathsf{body}^-(r)$ to denote $\{B_1, \ldots, B_m\}$. If $H = \bot$, we say that r is a *constraint*. If $m = 0$, we say the rule is *positive*; if r contains no variables, it is said to be *ground*. We say the rule r is *safe* if every variable in the rule occurs in some literal of $\mathsf{body}^+(r)$. We will require every rule to be safe (besides being a common requirement, safe rules suffice for our purposes).

As customary, we will consider two kinds of sets of rules:

1. finite sets of rules of the form H :- , with $H \neq \bot$ (notice that such rules must be ground because of safety), which are commonly used to represent databases – a set of this kind will be called an *extensional database*;
2. finite sets of rules of any other form – a set of this kind will be called a *program*.

Semantics. Let $\mathcal{P}$ be a program and ED an extensional database. We will often use $\mathcal{P}_{ED}$ to denote the set $\mathcal{P} \cup ED$. The *Herbrand universe* of $\mathcal{P}_{ED}$, denoted $\mathsf{U}(\mathcal{P}_{ED})$, is the set of all constants occurring in $\mathcal{P}_{ED}$. The *Herbrand base* of $\mathcal{P}_{ED}$, denoted $\mathsf{base}(\mathcal{P}_{ED})$, is the set of all atoms that can be built using relations and constants occurring in $\mathcal{P}_{ED}$. A *ground version* of a rule $r \in \mathcal{P}_{ED}$ is a ground rule r' that can be obtained from r by replacing all occurrences of each variable x of r with some constant from $\mathsf{U}(\mathcal{P}_{ED})$.

The *grounding* of $\mathcal{P}_{ED}$, denoted $\mathsf{ground}(\mathcal{P}_{ED})$, is the set of rules obtained from $\mathcal{P}_{ED}$ by replacing each rule $r \in \mathcal{P}_{ED}$ with all its ground versions.

We say that an instance I *satisfies* a ground positive literal L if either L is of the form $\alpha = \beta$ and α and β are the same constant, or L is an atom occurring in I. Furthermore, we say that I satisfies a ground negative literal $\neg L$, if I does not satisfy L. Finally, I satisfies a set of ground literals if I satisfies each literal in it.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

Consider a rule $r \in \mathsf{ground}(\mathcal{P}_{ED})$ and an instance I. We say that I *satisfies* r if, either r is a constraint and I does not satisfy $\mathsf{body}(r)$, or I satisfies $\mathsf{body}(r)$ implies that I satisfies $\mathsf{head}(r)$ (notice that an empty body is always satisfied).

A *model* of $\mathcal{P}_{ED}$ is an instance M such that $M \subseteq \mathsf{base}(\mathcal{P}_{ED})$ and such that M satisfies each rule of $\mathsf{ground}(\mathcal{P}_{ED})$. We say that M is *minimal* if there is no other model M' of $\mathcal{P}$ such that $M' \subsetneq M$. We use $\mathsf{MM}(\mathcal{P}_{ED})$ to denote the set of all minimal models of $\mathcal{P}$.

The *reduct* of $\mathcal{P}_{ED}$ w.r.t. some instance I is the set of ground rules obtained from $\mathsf{ground}(\mathcal{P}_{ED})$ by removing each rule r for which I does not satisfy $\mathsf{body}^-(r)$, and by removing all negative literals from the body of each rule r for which I satisfies $\mathsf{body}^-(r)$.

An instance M is a *stable model* of $\mathcal{P}_{ED}$ if $M \in \mathsf{MM}(\mathcal{P}'_{ED})$, where $\mathcal{P}'_{ED}$ is the reduct of $\mathcal{P}_{ED}$ w.r.t. M. We use $\mathsf{SM}(\mathcal{P}_{ED})$ to denote the set of all stable models of $\mathcal{P}_{ED}$.

Cautious Reasoning. Consider an extensional database ED, a program $\mathcal{P}$, and a query Q. The *cautious answers* to Q over ED and $\mathcal{P}$ is the set:

$$\mathsf{cans}_{\mathcal{P}}(ED, Q) = \bigcap_{M \in \mathsf{SM}(\mathcal{P}_{ED})} Q(M).$$

The key task we are interested in, regarding logic programs, is computing cautious answers. In particular, we are interested in its data complexity, that is, when the program and the query are fixed; as usual, we focus on the decision version of the problem. In the following, $\mathcal{P}$ and Q denote some program and some query, respectively:

PROBLEM :	$\mathsf{CANS}(\mathcal{P}, Q)$
INPUT :	An extensional database ED and a tuple $\mathbf{t} \in \mathsf{Const}^{ar(Q)}$.
QUESTION :	Is $\mathbf{t} \in \mathsf{cans}_{\mathcal{P}}(ED, Q)$?

It is well known that for every program $\mathcal{P}$ and every query Q, $\mathsf{CANS}(\mathcal{P}, Q)$ is in coNP – e.g., see (Greco *et al*. 1995).

The choice construct. We now extend logic programs with an additional construct, called *choice*. We point out that extending logic programs with the choice is purely for syntactic convenience, as this construct can be implemented by means of standard rules with negation.

The choice construct has been introduced in Datalog in (Saccà and Zaniolo 1990), studied in (Giannotti *et al*. 1991; Greco *et al*. 1995; Greco and Zaniolo 1998; Greco *et al*. 1992), and implemented in the Datalog systems LDL++ (Arni *et al*. 2003) and, in some form, in recent ASP systems (e.g., Potassco (Gebser *et al*. 2011) and DLV (Alviano *et al*. 2010)). It is used to enforce functional dependency (FD) constraints on rules of a logic program.

A *choice rule* r is an expression of the form

$$H \text{ :- } A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m, \mathit{choice}((X), (Y)).$$

where n, m, H, $A_1, \ldots, A_n$, and $B_1, \ldots, B_m$ are all defined as for standard rules, while X and Y denote *disjoint* sets of variables occurring in $\mathsf{body}(r)$.[3] The original definition of choice rule allows for multiple choice constructs in the rule body; here we focus on choice rules with only one choice construct in the body as this is enough for our purposes.

[3] When X (resp., Y) is a singleton, we may use its only element in place of X (resp., Y).

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

Intuitively, the construct $choice((X),(Y))$ prescribes that the set of all consequences derived from r must respect the FD $X \rightarrow Y$.

The formal semantics of choice rules is given in terms of a translation to standard rules using negation. In particular, the choice rule r defined above is a shorthand for writing the following set of rules; in what follows, $\mathbf{x}$ and $\mathbf{y}$ are the tuples of all variables in X and Y, respectively, in some arbitrary order.

$$r^{(1)}: \quad \mathsf{Range}_r(\mathbf{y}) \text{ :- } A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m.$$
$$r^{(2)}: \quad H \text{ :- } A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m, \mathsf{Chosen}_r(\mathbf{x}, \mathbf{y}).$$
$$r^{(3)}: \quad \mathsf{Chosen}_r(\mathbf{x}, \mathbf{y}) \text{ :- } A_1, \ldots, A_n, \neg B_1, \ldots, \neg B_m, \neg \mathsf{DiffChoice}_r(\mathbf{x}, \mathbf{y}).$$
$$r_i^{(4)}: \quad \mathsf{DiffChoice}_r(\mathbf{x}, \mathbf{y}) \text{ :- } \mathsf{Chosen}_r(\mathbf{x}, \mathbf{w}), \mathsf{Range}_r(\mathbf{y}), \mathbf{y}[i] \neq \mathbf{w}[i], \ \forall i \in \{1, \ldots, |Y|\}.$$

In the above rules, Range_r, Chosen_r, and $\mathsf{DiffChoice}_r$ are fresh relations not occurring in $\mathcal{P}$, which are used only to rewrite the rule r.

5.1 Implementing supported certain answers via logic programming with choice

The goal of this section is to prove the following key result.

Theorem 9
For every weakly-acyclic data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, and every query Q over T, there exists a program $\mathcal{P}$ such that $\mathsf{SCERT}(\mathcal{S}, Q)$ reduces to $\mathsf{CANS}(\mathcal{P}, Q)$ in polynomial time.

The rest of this section is devoted to prove the above claim. In particular, we show how to convert a weakly-acyclic data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, together with a source instance I of $\mathcal{S}$ and a query Q over T, into an extensional database ED and a program $\mathcal{P}$ using choice rules, in such a way that $\mathcal{P}$ depends only on $\mathcal{S}$ and such that $\mathsf{scert}_{\mathcal{S}}(I, Q) = \mathsf{cans}_{\mathcal{P}}(ED, Q)$.

The main idea of the translation is to derive a program together with an extensional database such that the stable models correspond to a subset of the supported solutions that is enough for computing supported certain answers. For this, we rely on the following useful result that one can extract from the proof of Theorem 7. For a set S of terms and a set of instances $\mathcal{I}$, we use $\mathcal{I}_{\downarrow S}$ to denote the set of instances $\{I \in \mathcal{I} \mid \mathsf{dom}(I) \subseteq S\}$.

Lemma 10
Consider a weakly-acyclic data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$. There exists a polynomial pol such that, for every source instance I of $\mathcal{S}$, and every query Q over T, the following holds:

$$\mathsf{scert}_{\mathcal{S}}(I, Q) = \bigcap_{J \in \mathsf{ssol}(I, \mathcal{S})_{\downarrow S}} Q(J),$$

where S is the set of all constants occurring in $\mathcal{S}$, I and Q, plus some fixed, arbitrarily chosen constants $c_1, \ldots, c_{\mathsf{pol}(|I|)}$ not occurring anywhere in $\mathcal{S}$, I, or Q.

Proof
The claim easily follows by construction of the non-deterministic procedure building the instance J^* in the proof of Theorem 7, from the fact that it terminates after a polynomial

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

number of steps w.r.t. I, and the fact that it halts with $J^* \neq \bot$ iff J^* is a supported solution in $\mathsf{ssol}(I, \mathcal{S})$. □

The result above tells us that considering supported solutions of a certain polynomial size suffices for computing supported certain answers. The stable models of the program together with the extensional database we are going to define will correspond to such supported solutions.

Definition 7 (*Translation*)
Consider a data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, a source instance I of $\mathcal{S}$, a query Q over T, and the set of constants S as defined in Lemma 10 w.r.t. $\mathcal{S}$, I and Q.

We use $\mathsf{LP}(\mathcal{S})$ to denote the set consisting of the following rules.

1. For each TGD ρ of the form $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \exists \mathbf{z}\, \beta_1 \wedge \cdots \wedge \beta_m$ in $\Sigma_{st} \cup \Sigma_t$, with $\mathbf{y} = \mathsf{fr}(\rho)$, if $k = |\mathbf{z}| = 0$, the following rules are introduced:

$$\beta_i \text{ :- } \alpha_1, \ldots, \alpha_n, \quad i \in \{1, \ldots, m\}, \tag{1}$$

otherwise, the following rules are introduced:

$$\mathsf{ExChoice}_\rho(\mathbf{y}, \mathbf{z}) \text{ :- } \alpha_1, \ldots, \alpha_n, \mathsf{Dom}(\mathbf{z}[1]), \ldots, \mathsf{Dom}(\mathbf{z}[k]), \mathit{choice}((Y), (Z)), \tag{2}$$

$$\beta_i \text{ :- } \mathsf{ExChoice}_\rho(\mathbf{y}, \mathbf{z}), \quad i \in \{1, \ldots, m\}, \tag{3}$$

where Y and Z are the sets of all variables in $\mathbf{y}$ and $\mathbf{z}$, respectively, and Dom is a fresh predicate.
2. For each EGD $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow x = y$ in Σ_t, the following constraint is introduced:

$$\bot \text{ :- } \alpha_1, \ldots, \alpha_n, x \neq y \tag{4}$$

We use $\mathsf{ED}(\mathcal{S}, I, Q)$ to denote the extensional database consisting of the following rules.

1. For each constant $c \in S$, the following rule is introduced:

$$\mathsf{Dom}(c) \text{ :- } . \tag{5}$$

2. For each fact $\alpha \in I$, the following rule is introduced:

$$\alpha \text{ :- } . \tag{6}$$

Example 6
Considering the data exchange setting $\mathcal{S}$ and the source instance I of $\mathcal{S}$ from Example 2, we have that $\mathsf{LP}(\mathcal{S})$ is the following logic program:

$$\mathsf{ExChoice}_{\rho_1}(x, z) \text{ :- } \mathsf{Emp}(x),\ \mathsf{Dom}(z), \mathit{choice}((x), (z)).$$
$$\mathsf{EmpC}(x, z) \text{ :- } \mathsf{ExChoice}_{\rho_1}(x, z).$$

$$\mathsf{EmpC}(x, y) \text{ :- } \mathsf{KnownC}(x, y).$$
$$\mathsf{SameC}(x, x') \text{ :- } \mathsf{EmpC}(x, y),\ \mathsf{EmpC}(x', y).$$
$$\bot \text{ :- } \mathsf{EmpC}(x, y),\ \mathsf{EmpC}(x, z),\ y \neq z.$$

Intuitively, the choice rule associated to the TGD ρ_1 is in charge of non-deterministically assigning a certain value to the existential variables of ρ_1, for each value its frontier variables can take, that is, the choice rule essentially builds an ex-choice for ρ_1. Once the

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

ex-choice is constructed, the rule $\mathsf{EmpC}(x, z)$:- $\mathsf{ExChoice}_{\rho_1}(x, z)$ simply propagates these choices to the head of ρ_1, as needed. All other TGDs have no existential quantification, and so use no choice construct. Finally, the only EGD η is converted to a constraint, so that the stable models of the logic program satisfy η.

We are now ready to prove Theorem 9.

Proof of Theorem 9. Given an instance I over a schema S, and a schema $\mathsf{S}' \subseteq \mathsf{S}$, we use $I[\mathsf{S}']$ to denote the restriction of I to only its facts referring to relations in S'. Notice that for every query Q over S', the following holds: $Q(I) = Q(I[\mathsf{S}'])$.

Consider a data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, a source instance I of $\mathcal{S}$, a query Q over T, and the set of constants S as defined in Lemma 10 w.r.t. $\mathcal{S}$, I, and Q.

Let $\mathcal{P} = \mathsf{LP}(\mathcal{S})$ and $ED = \mathsf{ED}(\mathcal{S}, I, Q)$.

We want to show $\mathsf{cans}_{\mathcal{P}}(ED, Q) = \mathsf{scert}_{\mathcal{S}}(I, Q)$. Leveraging Lemma 10, we show that $\{M[\mathsf{T}] \mid M \in \mathsf{SM}(\mathcal{P}_{ED})\} = \mathsf{ssol}(I, \mathcal{S})_{\downarrow S}$.

(1) In the following, we show $\{M[\mathsf{T}] \mid M \in \mathsf{SM}(\mathcal{P}_{ED})\} \subseteq \mathsf{ssol}(I, \mathcal{S})_{\downarrow S}$. Let $X \in \{M[\mathsf{T}] \mid M \in \mathsf{SM}(\mathcal{P}_{ED})\}$ and M be a stable model in $\mathsf{SM}(\mathcal{P}_{ED})$ such that $X = M[\mathsf{T}]$.

Let γ be an ex-choice defined as follows: given a TGD $\rho = \varphi(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{y}, \mathbf{z})$ in $\Sigma_{st} \cup \Sigma_t$ and a tuple $\mathbf{t} \in \mathsf{Const}^{|\mathbf{y}|}$, γ returns a set $\gamma(\rho, \mathbf{t})$ of pairs of the form (z_i, c), one for each existential variable $z_i \in \mathbf{z}$, where c is defined as follows: if $\mathsf{ExChoice}_{\rho}(\mathbf{t}, c_1, \ldots, c_k) \in M$, then $c = c_i$, otherwise c is an arbitrary constant of S.

It is easy to see that $\mathsf{U}(\mathcal{P}_{ED}) = S$, and thus X contains only constants in S. Moreover, $I \cup X$ satisfies Σ_{st}^{γ}, because otherwise M would not satisfy some ground version of the rules derived from the TGDs in Σ_{st}^{γ}. Also, X satisfies Σ_t^{γ}, because otherwise M would not satisfy some ground version of the rules derived from the TGDs/EGDs in Σ_t^{γ}.

Since every stable model is also a minimal model, the minimality of M ensures that there is no $J' \subsetneq X$ such that $I \cup J'$ satisfies Σ_{st}^{γ} and J' satisfies Σ_t^{γ}. Thus, X is a supported solution of I w.r.t. $\mathcal{S}$ containing only constants in S.

(2) We now show $\{M[\mathsf{T}] \mid M \in \mathsf{SM}(\mathcal{P}_{ED})\} \supseteq \mathsf{ssol}(I, \mathcal{S})_{\downarrow S}$. Let $J \in \mathsf{ssol}(I, \mathcal{S})_{\downarrow S}$ and γ be the ex-choice for which $I \cup J$ satisfies Σ_{st}^{γ} and J satisfies Σ_t^{γ}. Let $X = I \cup J \cup \{\mathsf{Dom}(c) \mid c \in S\}$. We show that $X \in \mathsf{SM}(\mathcal{P}_{ED})$.

First, X satisfies each ground rule in $\mathsf{ground}(\mathcal{P})$ of the form β_i :- $\alpha_1, \ldots, \alpha_n$ (cf. (1) in Definition 7), because otherwise the TGD of the form $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta_1 \wedge \cdots \wedge \beta_m$ in Σ_{st}^{γ} or Σ_t^{γ} would not be satisfied by $I \cup J$ or J, respectively.

Also, X satisfies the ground rules in $\mathsf{ground}(\mathcal{P})$ of the form (2)–(3) in Definition 7, derived from a TGD having existential variables, because otherwise such a TGD would not be satisfied by either $I \cup J$ or J, or J would not be minimal.

Further, X satisfies each ground constraint of the form $\bot$:- $\alpha_1, \ldots, \alpha_n, x \neq y$ (cf. (4) in Definition 7) in $\mathsf{ground}(\mathcal{P})$ as otherwise J would not satisfy the EGD $\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow x = y$ in Σ_t^{γ}.

Then, X satisfies each rule in ED of the form (5) of Definition 7 because $\{\mathsf{Dom}(c) \mid c \in S\} \subseteq X$.

Finally, X satisfies each rule in ED of the form (6) of Definition 7 because X contains I.

By the minimality of J we obtain the minimality of X, and thus, X is a stable model of $\mathcal{P}_{ED}$. Noting that $X[\mathsf{T}] = J$, we conclude that $J \in \{M[\mathsf{T}] \mid M \in \mathsf{SM}(\mathcal{P}_{ED})\}$. □

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

6 Approximate query answering via materialization

As already discussed in the introduction, there might exist scenarios where it is desirable 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 8, 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 (Imielinski and Lipski 1984), 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 8
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 9 (*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 con-

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

struct one. We show how to deal with general weakly-acyclic settings with EGDs in Section 7.

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, that is, it triggers the TGD. 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 7
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 10 (*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)* $\mathcal{J} = \mathcal{I} \cup \{\langle \alpha, \phi\rangle \mid \alpha \in \mathsf{result}(\mathcal{I},\rho,h)\}$.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

Example 8
Consider the conditional instance $\mathcal{I}$, the homomorphism h, and the TGD ρ_3 of Example 7. Then, $\mathcal{I} \xrightarrow{\rho_3,h,\phi} \mathcal{J}$ is a conditional chase step, where ϕ is the condition $\perp_2 = \mathsf{miami} \wedge \perp_1 = a$, 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 shorthand 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 11 (*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. That is, at a certain step, a fact is produced only if the possible worlds in which it occurs is not a subset of the possible worlds in which the same fact has already been introduced by previous steps. An example follows.

Example 9
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. $\mathcal{S}$:

$$\mathcal{J}_0 = \{\langle A(a), \mathsf{true}\rangle, \langle B(b_1), \mathsf{true}\rangle, \langle B(b_2), \mathsf{true}\rangle\}, \qquad \mathcal{J}_1 = \mathcal{J}_0 \cup \{\langle R(a, \perp), \mathsf{true}\rangle\},$$
$$\mathcal{J}_2 = \mathcal{J}_1 \cup \{\langle S(b_1), \mathsf{true}\rangle\}, \qquad \mathcal{J}_3 = \mathcal{J}_2 \cup \{\langle S(b_2), \mathsf{true}\rangle\},$$
$$\mathcal{J}_4 = \mathcal{J}_3 \cup \{\langle T(a), \perp = b_1\rangle\}, \qquad \mathcal{J}_5 = \mathcal{J}_4 \cup \{\langle T(a), \perp = b_2\rangle\}.$$

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 10
Consider the conditional chase sequence $\mathcal{J}_0, \ldots, \mathcal{J}_5$ of Example 9. 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 11. 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

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

to allow the fact $T(a)$ to be present for two different reasons (either because $\perp = b_1$ or $\perp = b_2$), 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 11
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. $\mathcal{S}$.

Proof
To prove the claim, it is enough to prove that each supported solution $J \in \mathsf{ssol}(I, \mathcal{S})$ is such that $I \cup J$ is a possible world of $\mathcal{J}$, and that each possible world J of $\mathcal{J}$ contains a supported solution. We prove first that each $J \in \mathsf{ssol}(I, \mathcal{S})$ is such that $I \cup J$ is a possible world of $\mathcal{J}$.

Let γ be the ex-choice witnessing that J is a supported solution. Then, J can be characterized as the result of a procedure that computes a sequence $J_0, J_1, \ldots J_m$ such that $J_0 = I$, $J_m = J$, and each J_i with $i > 0$ is obtained from J_{i-1} by adding the head of a TGD in $\Sigma^{\gamma}_{st} \cup \Sigma^{\gamma}_{t}$ whose body is contained in J_{i-1} (i.e., the first part of the procedure in the proof of Theorem 7) – notice that such a procedure ensures also the minimality of J. For each step of the aforementioned procedure, there must be a corresponding conditional chase step in the sequence yielding $\mathcal{J}$, which in turn induces $I \cup J$ as a possible world.

Regarding whether each possible world J of $\mathcal{J}$ contains a supported solution, consider a possible world $J \in \mathsf{pw}(\mathcal{J})$. By construction of $\mathcal{J}$, $J = I \cup J'$, for some instance J' over T, since all conditional facts in $\mathcal{J}$, which correspond to the facts in I, have the always true condition. Moreover, by construction of $\mathcal{J}$, $I \cup J'$ satisfies Σ_{st}, and J' satisfies Σ_t. Hence, if we consider the set of TGDs $\Sigma^{*}_{st} \cup \Sigma^{*}_{t}$, where Σ^{*}_{st} and Σ^{*}_{t} are the sets of all ground versions of the TGDs in Σ_{st} and Σ_t, respectively, we have that $I \cup J'$ satisfies $\Sigma^{*}_{st} \cup \Sigma^{*}_{t}$ and J' satisfies Σ^{*}_{t}. However, since $\Sigma^{\gamma}_{st} \subseteq \Sigma^{*}_{st}$, and $\Sigma^{\gamma}_{t} \subseteq \Sigma^{*}_{t}$, for any ex-choice γ, we must have that J' must contain a supported solution in $\mathsf{ssol}(I, \mathcal{S})$, as needed. □

Example 11
Consider the setting $\mathcal{S}$, the source instance I of $\mathcal{S}$, and the conditional chase sequence $\mathcal{J}_0, \ldots, \mathcal{J}_5$ of Example 9. From Theorem 11, we conclude that $\mathcal{J}_5$ is an approximate conditional solution for 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 12
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. $|I|$.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

Proof

To prove that the length of a conditional chase sequence is bounded by a polynomial, it suffices to follow an argument similar to the one given in (Fagin *et al.* 2005) for proving that the length of a standard chase sequence is polynomial, for weakly-acyclic settings. Let $s = (\mathcal{J}_i)_{0 \leq i \leq n}$ be a conditional chase sequence of I w.r.t. $\mathcal{S}$, with $\mathcal{J}_i \xrightarrow{\rho_i, h_i, \phi_i} \mathcal{J}_{i+1}$, for $i \in \{0, \ldots, n-1\}$. Since n is a polynomial of $|I|$, we just need to show that for each $i \in \{0, \ldots, n-1\}$, $\mathcal{J}_{i+1}$ can be constructed in polynomial time. To this end, it suffices to focus on condition (ii) of Definition 10 and condition (iii) of Definition 11. Since n is polynomial, the maximum number of terms occurring in each condition ϕ_i is polynomial. Thus, each ϕ_i contains at most polynomially many equalities, and we can easily check whether $\phi_i \not\models \mathsf{false}$, by simply computing the closure of all equalities in ϕ_i, and checking whether an equality of the form $a = b$ can be derived, where a, b are distinct constants. Similarly, for each $i \in \{0, \ldots, n-1\}$, and every $j < i$, we can check whether $\langle h_i(\mathsf{fr}(\rho)), \phi_i \rangle \not\sqsubseteq \langle h_j(\mathsf{fr}(\rho)), \phi_j \rangle$, by using a similar approach. □

Querying Approximate Conditional Solutions. What now remains to show is how we can compute the conditional certain answers over an approximate conditional solution, for example, 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 (Imielinski and Lipski 1984). 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 (Greco *et al.* 2019) to compute approximate certain answers over incomplete databases. Here we only recall the main properties of the algorithm. For more details, we refer the reader to (Greco *et al.* 2019).

For a query Q, the function $\dot{Q}_t$ from conditional instances to sets of tuples is defined in (Greco *et al.* 2019), and it is such that the following holds.

Theorem 13

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. $|\mathcal{J}|$.

The theorem above implies that the approximation algorithm provides so-called *correctness guarantees* (Item 1 of the theorem), that is, the algorithm always constructs a subset of the conditional certain answers, and thus, only returns correct answers. This is the standard notion for measuring the quality of the set of approximate answers these algorithms are able to compute, in the context of querying incomplete databases – for example, see (Libkin 2016; Guagliardo and Libkin 2016; Console *et al.* 2016). To the best of our knowledge, none of the existing approximation algorithms from the literature provide other kinds of theoretical guarantees, for example, w.r.t. to "how complete" the set of approximate answers is.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

From the result above, Theorem 12, and Definition 11, 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. $|\mathcal{J}|$.

7 Dealing with EGDs

We now show how to deal with weakly-acyclic settings with EGDs, when it comes to construct approximate conditional solutions.

Consider a weakly-acyclic data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$ and a source instance I. We assume that $\mathsf{ssol}(I, \mathcal{S}) \neq \emptyset$. Checking whether $\mathsf{ssol}(I, \mathcal{S}) = \emptyset$ is feasible in polynomial time, for weakly-acyclic settings (Theorem 5), and if $\mathsf{ssol}(I, \mathcal{S})$ is empty, no approximate conditional solution can be constructed.

The goal is to first construct an approximate conditional solution $\mathcal{J}$ for the data exchange setting $\mathcal{S}^{\exists}$ obtained from $\mathcal{S}$ by removing the set Σ_E of all EGDs from Σ_t. Then, we show that for every query Q, the EGDs in Σ_E can be embedded in Q, obtaining a query Q', in such a way that

$$\mathsf{con\text{-}cert}(\mathcal{J}, Q') = \bigcap_{J \in \mathsf{pw}(\mathcal{J}) \text{ and } J \text{ satisfies } \Sigma_E} Q(J).$$

As we will see, this will imply that $\mathsf{con\text{-}cert}(\mathcal{J}, Q') \subseteq \mathsf{scert}_{\mathcal{S}}(I, Q)$.

Thus, modulo a rewriting of Q, we can exploit $\mathcal{J}$ to compute an approximation of the supported certain answers of Q. Despite our efforts, we were not able to prove that Q' is also such that $\mathsf{con\text{-}cert}(\mathcal{J}, Q') = \mathsf{scert}_{\mathcal{S}}(I, Q)$, when Q is positive. It is an open question that we hope to answer in a future work.

In what follows, for a data exchange setting $\mathcal{S}$, $\mathcal{S}^{\exists}$ denotes the setting obtained from $\mathcal{S}$ by removing the set Σ_E of all EGDs from Σ_t.

Lemma 14
Consider a weakly-acyclic data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, and assume I is a source instance of $\mathcal{S}$ such that $\mathsf{ssol}(I, \mathcal{S}) \neq \emptyset$. Moreover, let $\mathcal{J}$ be an approximate conditional solution of I w.r.t. $\mathcal{S}^{\exists}$. Then, for every query Q, there exists a query Q', which depends only on Q and the set of EGDs Σ_E in $\mathcal{S}$, such that

$$\mathsf{con\text{-}cert}(\mathcal{J}, Q') = \bigcap_{J \in \mathsf{pw}(\mathcal{J}) \text{ and } J \text{ satisfies } \Sigma_E} Q(J).$$

Proof
Let $k = ar(Q)$. The goal is to construct, for a given query Q, a query Q' such that, for every target instance J, whenever all the EGDs in Σ_E are satisfied by J, then $Q'(J) = Q(J)$, and $Q'(J) = \mathcal{C}^k$ otherwise, where $\mathcal{C}$ is the set of all constants occurring in J, $\mathcal{S}$,

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

and Q. That is, if J does not satisfy Σ_E, the query Q' outputs every possible tuple of length k, using constants from J, $\mathcal{S}$, and Q. Clearly, if Q' enjoys the above property, the claim will follow immediately. We now explain how the query Q' can be constructed, starting from Q and Σ_E. The query Q' is made of two subqueries, that are put together via a union. That is:

$$Q' = Q_1 \vee Q_2.$$

Q_1 is such that for every target instance J, if J satisfies Σ_E, then $Q_1(J) = Q(J)$, and $Q_1(J) = \emptyset$, otherwise. On the other hand, Q_2 is such that for every target instance J, if J satisfies Σ_E, then $Q_2(J) = \emptyset$, and $Q_2(J) = \mathcal{C}^k$, otherwise. It remains to show how Q_1 and Q_2 are constructed. For each EGD $\eta \in \Sigma_E$, we let Q_η be the boolean query such that for every instance J over T, $Q_\eta(J) = \{()\}$, if η satisfies J, and $Q_\eta(J) = \emptyset$, otherwise. Furthermore, we use $Q_\eta^\neg$ to denote the complement of Q_η, that is $Q_\eta^\neg(J) = \{()\}$ iff $Q_\eta(J) = \emptyset$. All the above queries can be easily written in FO. Finally, we let Q_{dom} be the query of arity k, such that, for every target instance J, $Q_{\mathsf{dom}}(J)$ is the set of all tuples of length k over the constants in J, $\mathcal{S}$, and Q. The above query can be encoded with a UCQ. Then, we have

$$Q_1(x_1, \ldots, x_k) = Q(x_1, \ldots, x_k) \wedge \bigwedge_{\eta \in \Sigma_E} Q_\eta,$$

and

$$Q_2(x_1, \ldots, x_k) = Q_{\mathsf{dom}}(x_1, \ldots, x_k) \wedge \bigvee_{\eta \in \Sigma_E} Q_\eta^\neg.$$

By construction, $Q_1(J) = Q(J)$ if J satisfies Σ_E and $Q_1(J) = \emptyset$, otherwise, and $Q_2(J) = \emptyset$, if J satisfies Σ_E, and $Q_2(J) = \mathcal{C}^k$, otherwise. □

From the result above, and from the fact that the supported solutions of a data exchange setting $\mathcal{S}$ correspond to the supported solutions of $\mathcal{S}^\exists$ that also satisfy the EGDs of $\mathcal{S}$, we obtain the main result of this section.

Theorem 15
Consider a weakly-acyclic data exchange setting $\mathcal{S} = \langle \mathsf{S}, \mathsf{T}, \Sigma_{st}, \Sigma_t \rangle$, and assume I is a source instance of $\mathcal{S}$ such that $\mathsf{ssol}(I, \mathcal{S}) \neq \emptyset$. Moreover, let $\mathcal{J}$ be an approximate conditional solution of I w.r.t. $\mathcal{S}^\exists$. Then, for every query Q, there exists a query Q', which depends only on Q and the set of EGDs Σ_E in $\mathcal{S}$, such that

$$\mathsf{con\text{-}cert}(\mathcal{J}, Q') \subseteq \mathsf{scert}_{\mathcal{S}}(I, Q).$$

Proof
From Lemma 14, there exists a query Q', depending only on Q and Σ_E, such that

$$\mathsf{con\text{-}cert}(\mathcal{J}, Q') = \bigcap_{J \in \mathsf{pw}(\mathcal{J}) \text{ and } J \text{ satisfies } \Sigma_E} Q(J). \tag{7}$$

From the definition of approximate conditional solution, we have that $\mathsf{ssol}(I, \mathcal{S}^\exists) \subseteq \mathsf{pw}(\mathcal{J})$. Moreover, by definition of supported solution, $\mathsf{ssol}(I, \mathcal{S}) = \{J \in \mathsf{ssol}(I, \mathcal{S}) \mid J \text{ satisfies } \Sigma_E\}$. Hence, $\mathsf{ssol}(I, \mathcal{S}) \subseteq \{J \in \mathsf{pw}(\mathcal{J}) \mid J \text{ satisfies } \Sigma_E\}$. The latter inclusion and equation (7) let us conclude that $\mathsf{con\text{-}cert}(\mathcal{J}, Q') \subseteq \mathsf{scert}_{\mathcal{S}}(I, Q)$. □

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

The above results tell us that we can still materialize a target instance, even for weakly-acyclic settings that allow for EGDs. Moreover, modulo a rewriting of the query Q, the constructed target instance allows for the construction of a subset of supported certain answers of Q.

8 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 (Arenas *et al*. 2013), 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. Adapting our notion of supported solution to the setting of data exchange with incomplete instances is a non-trivial task which we will consider for future work.

In Section 6, we have seen how a conditional extension of the chase procedure, working on a normalized form of TGDs, can be employed to compute in polynomial time, for weakly-acyclic settings, an approximate conditional solution. A similar normal form to the one we employ in our paper is presented in (Gheerbrant and Sirangelo 2019). However, in that work, the normal form is applied to *queries*, and the goal is to compute so-called best answers of UCQs over incomplete databases, while in our case, we employ a normal form for *TGDs*, which we then use to simplify the definition of the conditional chase. Finally, 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 (Grahne and Onet 2011) introduces a conditional version of the chase procedure which is similar to ours. The main difference is that the conditional chase of (Grahne and Onet 2011) 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., (Calautti *et al*. 2015; Grahne and Onet 2018; Calautti and Pieris 2019; 2021; Calautti *et al*. 2022) for more details). For this reason, it is not difficult to show that when considering weakly-acyclic settings, the conditional chase of (Grahne and Onet 2011) 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. 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, that is, 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 (Calvanese *et al*. 2007),

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

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), that is, 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, that is, $Q(x) = \mathsf{Emp}'(x) \wedge \neg\mathsf{WorkFromHome}(x)$.

According to (Calvanese *et al*. 2007), 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 employee 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 would like to identify other quality measures of our approximation algorithm using techniques such as the ones introduced in (Libkin 2018). We also plan to experimentally evaluate both our translation to logic programs for computing exact answers, as well as our materialization-based approaches for computing approximate answers by means of a dedicated benchmark, as done for example, in the context of approximate consistent query answering (Calautti *et al*. 2021).

To conclude, we mention that explaining query answering has recently drawn considerable attention under existential rule languages (e.g., see (Lukasiewicz *et al*. 2022; Ceylan *et al*. 2021; 2020; Lukasiewicz *et al*. 2020; Ceylan *et al*. 2019)), and knowledge representation in general (e.g., in the context of argumentation (Alfano *et al*. 2020)). Hence,

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

an interesting direction for future work is to address such issues in our setting. Also, it would be interesting to account for user preferences when answering queries, as recently done in (Calautti *et al.* 2022) for ontology-mediated queries.

Competing interests

The author(s) declare none.

References

Alfano, G., Calautti, M., Greco, S., Parisi, F. and Trubitsyna, I. 2020. Explainable acceptance in probabilistic abstract argumentation: Complexity and approximation. In *KR*, 33–43.

Alviano, M., Faber, W., Leone, N., Perri, S., Pfeifer, G. and Terracina, G. 2010. The disjunctive datalog system DLV. In *Datalog Reloaded - First International Workshop, Datalog*, O. de Moor, G. Gottlob, T. Furche and A. J. Sellers, Eds. Lecture Notes in Computer Science, vol. 6702. Springer, 282–301.

Arenas, M., Pérez, J. and Reutter, J. L. 2013. Data exchange beyond complete data. *Journal of the ACM 60,* 4, 28:1–28:59.

Arni, F., Ong, K., Tsur, S., Wang, H. and Zaniolo, C. 2003. The deductive database system LDL++. *Theory and Practice of Logic Programming 3,* 1, 61–94.

Calautti, M., Console, M. and Pieris, A. 2021. Benchmarking approximate consistent query answering. In *PODS*, L. Libkin, R. Pichler and P. Guagliardo, Eds., 233–246.

Calautti, M., Gottlob, G. and Pieris, A. 2015. Chase termination for guarded existential rules. In *PODS*, 91–103.

Calautti, M., Gottlob, G. and Pieris, A. 2022. Non-uniformly terminating chase: Size and complexity. In *PODS*, 369–378.

Calautti, M., Greco, S., Molinaro, C. and Trubitsyna, I. 2022. Preference-based inconsistency-tolerant query answering under existential rules. *Artificial Intelligence 312*, 103772.

Calautti, M. and Pieris, A. 2019. Oblivious chase termination: The sticky case. In *ICDT*, 17:1–17:18.

Calautti, M. and Pieris, A. 2021. Semi-oblivious chase termination: The sticky case. *Theory of Computing Systems 65,* 1, 84–121.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M. and Rosati, R. 2007. Eql-lite: Effective first-order query processing in description logics. In *IJCAI*, 274–279.

Ceylan, İ. İ., Lukasiewicz, T., Malizia, E., Molinaro, C. and Vaicenavicius, A. 2020. Explanations for negative query answers under existential rules. In *Proc. KR*, 223–232.

Ceylan, İ. İ., Lukasiewicz, T., Malizia, E., Molinaro, C. and Vaicenavicius, A. 2021. Preferred explanations for ontology-mediated queries under existential rules. In *Proc. AAAI*, 6262–6270.

Ceylan, İ. İ., Lukasiewicz, T., Malizia, E. and Vaicenavicius, A. 2019. Explanations for query answers under existential rules. In *Proc. IJCAI*, 1639–1646.

Console, M., Guagliardo, P. and Libkin, L. 2016. Approximations and refinements of certain answers via many-valued logics. In *KR*, 349–358.

Fagin, R., Kolaitis, P. G., Miller, R. J. and Popa, L. 2005. Data exchange: Semantics and query answering. *TCS 336,* 1, 89–124.

Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T. and Schneider, M. 2011. Potassco: The potsdam answer set solving collection. *AI Community 24,* 2, 107–124.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press

Gheerbrant, A. and Libkin, L. 2015. Certain answers over incomplete XML documents: Extending tractability boundary. *Theory of Computing Systems 57,* 4, 892–926.

Gheerbrant, A. and Sirangelo, C. 2019. Best answers over incomplete data: Complexity and first-order rewritings. In *IJCAI*, S. Kraus, Ed., 1704–1710.

Giannotti, F., Pedreschi, D., Saccà, D. and Zaniolo, C. 1991. Non-determinism in deductive databases. In *DOOD*. Springer, 129–146.

Grahne, G. and Onet, A. 2011. On conditional chase termination. In *AMW*.

Grahne, G. and Onet, A. 2018. Anatomy of the chase. *Fundamenta Informaticae 157,* 3, 221–270.

Greco, S., Molinaro, C. and Trubitsyna, I. 2019. Approximation algorithms for querying incomplete databases. *Information Systems 86*, 28–45.

Greco, S., Saccà, D. and Zaniolo, C. 1995. DATALOG queries with stratified negation and choice: From P to d^P. In *Database Theory - ICDT'95, 5th International Conference*, G. Gottlob and M. Y. Vardi, Eds. Lecture Notes in Computer Science, vol. 893. Springer, 82–96.

Greco, S. and Zaniolo, C. 1998. Greedy algorithms in datalog with choice and negation. In *IJCSLP*, 294–309.

Greco, S., Zaniolo, C. and Ganguly, S. 1992. Greedy by choice. In *Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, M. Y. Vardi and P. C. Kanellakis, Eds. ACM Press, 105–113.

Guagliardo, P. and Libkin, L. 2016. Making SQL queries correct on incomplete databases: A feasibility study. In *Proc. Symposium on Principles of Database Systems (PODS)*, 211–223.

Hernich, A. 2011. Answering non-monotonic queries in relational data exchange. *LMCS 7*, 3.

Hernich, A., Libkin, L. and Schweikardt, N. 2011. Closed world data exchange. *TODS 36,* 2, 14:1–14:40.

Imielinski, T. and Lipski, W. 1984. Incomplete information in relational databases. *Journal of the ACM 31,* 4, 761–791.

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

Libkin, L. 2016. SQL's three-valued logic and certain answers. *ACM Transactions Database Systems 41,* 1:1–1:28.

Libkin, L. 2018. Certain answers meet zero-one laws. In *PODS*, J. V. den Bussche and M. Arenas, Eds. ACM, 195–207.

Libkin, L. and Sirangelo, C. 2011. Data exchange and schema mappings in open and closed worlds. *JCSS 77,* 3, 542–571.

Lukasiewicz, T., Malizia, E. and Molinaro, C. 2020. Explanations for inconsistency-tolerant query answering under existential rules. In *Proc. AAAI*, 2909–2916.

Lukasiewicz, T., Malizia, E. and Molinaro, C. 2022. Explanations for negative query answers under inconsistency-tolerant semantics. In *Proc. IJCAI*, 2705–2711.

Marnette, B. 2009. Generalized schema-mappings: From termination to tractability. In *PODS*, 13–22.

Saccà, D. and Zaniolo, C. 1990. Stable models and non-determinism in logic programs with negation. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, April 2–4, 1990, Nashville, Tennessee, USA*, D. J. Rosenkrantz and Y. Sagiv, Eds. ACM Press, 205–217.

https://doi.org/10.1017/S1471068423000339 Published online by Cambridge University Press