Closed World Data Exchange

ANDRÉ HERNICH, Humboldt-Universität zu Berlin, Germany
LEONID LIBKIN, University of Edinburgh, United Kingdom
NICOLE SCHWEIKARDT, Goethe-Universität Frankfurt am Main, Germany

Data exchange deals with *translating* data structured in some source format into data structured in some target format, given a specification of the relationship between the source and the target and possibly constraints on the target, and *answering queries* over the target in a way that is semantically consistent with the information in the source. Theoretical foundations of data exchange have been actively explored recently. It was also noticed that the standard semantics for query answering in data exchange may lead to counter-intuitive or anomalous answers.

In the present paper, we explain that this behavior is due to the fact that solutions can contain "invented" information (i.e., information that is not related to the source instance), and that the presence of incomplete information in target instances has been ignored. In particular, proper query evaluation techniques for databases with nulls have not been used, and the distinction between closed and open world semantics has not been made.

We present a concept of solutions, called CWA-solutions, that is based on the closed world assumption. For data exchange settings without constraints on the target, the space of CWA-solutions has two extreme points: the canonical universal solution (the "maximal" CWA-solution) and the core of the universal solutions (the "minimal" CWA-solution), both of them well studied in data exchange. In the presence of constraints on the target, the core of the universal solutions is still the "minimal" CWA-solution, but there may be no unique "maximal" CWA-solution. We show how to define the semantics of query answering taking into account incomplete information, and show that some of the well-known anomalies go away with the new semantics. The paper also contains results on the complexity of query answering, upper approximations to queries (maybe-answers), and various extensions.

Categories and Subject Descriptors: H.2.5 [**Database Management**]: Heterogeneous Databases—*data translation*; H.2.4 [**Database Management**]: Systems—*relational databases*; *rule-based databases*; *query processing*; H.2.8 [**Database Management**]: Database Applications

General Terms: Theory, Databases

Additional Key Words and Phrases: closed world assumption, null values, certain answers, maybe answers, canonical universal solution, core, the chase

ACM Reference Format:
ACM Trans. Datab. Syst. V, N, Article A (January YYYY), 40 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

This is a preliminary release of an article accepted by ACM Transactions on Database Systems. The definitive version is currently in production at ACM and, when released, will supersede this version.
This article combines and extends significantly the conference papers [Libkin 2006], presented at the *25th ACM Symposium on Principles of Database Systems (PODS 2006)*, and [Hernich and Schweikardt 2007], presented at the *26th ACM Symposium on Principles of Database Systems (PODS 2007)*.

Authors' addresses: A. Hernich, Humboldt-Universität zu Berlin, Institut für Informatik, Unter den Linden 6, 10099 Berlin, Germany, e-mail: hernich@informatik.hu-berlin.de; L. Libkin, School of Informatics, University of Edinburgh, Crichton St., Edinburgh EH8 9LE, UK, e-mail: libkin@inf.ed.ac.uk; N. Schweikardt, Goethe-Universität, Institut für Informatik, Postfach 11 19 32, 60054 Frankfurt (Main), Germany, e-mail: schweika@informatik.uni-frankfurt.de.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

1.1. Data Exchange

Data exchange deals with translating data structured in some source format into data structured in some target format, given a specification of the relationship between the source and the target, and possibly constraints on the target. Data exchange is an old database problem (see, e.g., Shu et al. [1977]). In the past few years it received renewed attention through the development of commercial strength systems like Clio [Miller et al. 2001; Haas et al. 2005], and through the influential papers [Fagin et al. 2005a; Fagin et al. 2005b] which laid the theoretical foundations for data exchange. A good starting point into this area are the surveys [Kolaitis 2005; Barceló 2009; Hernich and Schweikardt 2010] and a very recent book [Arenas et al. 2010].

We focus on *relational* data exchange, as described by Fagin et al. [2005a]. The basic task in relational data exchange is: Given two disjoint (relational database) schemas σ and τ, a finite set Σ_{st} of constraints specifying the relationship between the source and the target, a finite set Σ_{t} of constraints on the target, and a (relational database) instance S over σ, compute an instance T over τ such that the joint instance $S \cup T$ that consists of all relations in S and T satisfies all the constraints in Σ_{st} and Σ_{t}. We call $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ a *data exchange setting*, S a *source instance* for D, and T a *solution* for S under D. Moreover, Σ_{st} is the set of *source-to-target dependencies*, and Σ_{t} the set of *target dependencies* of D. As in [Fagin et al. 2005a; Fagin et al. 2005b], we consider only data exchange settings $(\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, where Σ_{st} and Σ_{t} are finite sets of *tuple-generating dependencies (tgds)* and *equality-generating dependencies (egds)*.

Example 1.1. Consider a schema $\sigma = \{\mathit{Submission}, \mathit{PC}\}$, where relation *Submission* is supposed to store tuples $(\mathit{id}, \mathit{title})$ providing IDs and titles of papers submitted to a conference, and relation *PC* is supposed to store tuples $(\mathit{name}, \mathit{affiliation}, \mathit{paper\_id})$ providing information about PC members and the conference submissions they are assigned to. For example, let S be an instance over σ with[1]

$$\begin{aligned} \mathit{Submission}^S &= \{(1, t_1), (2, t_2)\}, \\ \mathit{PC}^S &= \{(n_1, a_1, 1), (n_1, a_1', 1), (n_2, a_2, 2), (n_3, a_3, 3)\}. \end{aligned}$$

Assume we want to translate instances over σ into instances over the schema $\tau = \{\mathit{Paper}, \mathit{PC}', \mathit{Assign}\}$, where *Paper* stores tuples $(\mathit{id},\mathit{title},\mathit{keywords})$ providing information about the submissions, PC' stores tuples $(\mathit{id}, \mathit{name})$ providing information about the PC members along with an ID, which is required to be a key, and *Assign* contains tuples $(\mathit{paper\_id}, \mathit{pc\_id})$ assigning submissions to PC members by their IDs. This translation can be described by a data exchange setting $D_{\text{conf}} = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, where Σ_{st} consists of the two tgds

$$\begin{aligned} d_1 &= \forall x_1 \forall x_2 \big(\mathit{Submission}(x_1, x_2) \to \exists z\, \mathit{Paper}(x_1, x_2, z)\big), \\ d_2 &= \forall x_1 \forall x_2 \forall y \big(\mathit{PC}(x_1, y, x_2) \to \exists z\, (\mathit{PC}'(z, x_1) \land \mathit{Assign}(x_2, z))\big), \end{aligned}$$

and Σ_{t} consists of the tgd, respectively egd,

$$\begin{aligned} d_3 &= \forall x_1 \forall x_2 \big(\mathit{Assign}(x_1, x_2) \to \exists z\, \exists z'\, \exists z''\, (\mathit{Paper}(x_1, z, z') \land \mathit{PC}'(x_2, z''))\big), \\ d_4 &= \forall x_1 \forall x_2 \forall x_3 \big(\mathit{PC}'(x_1, x_2) \land \mathit{PC}'(x_1, x_3) \to x_2 = x_3\big). \end{aligned}$$

[1] For a relation symbol R and an instance I, we write R^I for the interpretation of R in I.

One possible solution for S under D_{conf} is the instance T over τ with

$$\begin{aligned} \mathit{Paper}^T &= \{(1, t_1, \bot_1), (2, t_2, \bot_2), (3, \bot_3, \bot_4)\},\\ \mathit{PC}'^T &= \{(\bot_5, n_1), (\bot_6, n_2), (\bot_7, n_3)\},\\ \mathit{Assign}^T &= \{(1, \bot_5), (2, \bot_6), (3, \bot_7)\}.\end{aligned}$$

Here, the elements $\bot_1, \bot_2, \ldots, \bot_7$ are called *labeled nulls*, or just *nulls*, in contrast to the other elements in T that are provided by S, which are called *constants*. A null serves as a placeholder for a concrete (constant) value. □

Given a data exchange setting D and a source instance S for D, there may be many different solutions for S under D. Two natural questions are therefore: Which solution should we compute? And how can we answer queries posed against the target schema in a way that is semantically consistent with the information contained in S? Concerning the first question, Fagin et al. [2005a] introduced the notion of *universal solutions*, and argued that these should be the preferred solutions to materialize in data exchange. Furthermore, [Fagin et al. 2005a; Fagin et al. 2005b] identified two particular important universal solutions: the *canonical universal solution* (*canonical solution*, for short), and the *core of the universal solutions* (*core solution*, for short), which is the unique minimal universal solution up to renaming of nulls. Concerning the second question, Fagin et al. [2005a] adopted the *certain answers semantics* for query answering. Given a data exchange setting D, a source instance S for D, and a query Q over τ, the *certain answers of Q on S with respect to D* are defined as

$$\mathsf{certain}^D(Q, S) \;:=\; \bigcap \{Q(T) \mid T \text{ is a solution for } S \text{ under } D\}. \tag{1.1}$$

It was shown that the canonical solution and the core solution are good for answering positive queries, like unions of conjunctive queries, in the sense that the certain answers of a positive query Q can be computed by evaluating Q over such a solution (and removing all tuples with nulls from the result afterwards).

The above results provided the basis for extensions dealing, for example, with rewritability, query answering, schema composition, schema inversion, algorithmic issues, and other data models; see the papers [Mądry 2005; Arenas et al. 2004; Fagin et al. 2005; Arenas and Libkin 2008; Kolaitis et al. 2006; Fuxman et al. 2006; Gottlob and Nash 2008; Arenas et al. 2009] and the book [Arenas et al. 2010].

1.2. Anomalies of Query Answering in Data Exchange

It has been observed in [Fagin et al. 2005a; Arenas et al. 2004] that, on non-positive queries, the certain answers semantics exhibits counter-intuitive behavior – and sometimes this behavior can be described as anomalous. For instance, consider the data exchange setting $D^* = (\{E\}, \{E'\}, \Sigma_{\text{st}}, \emptyset)$, where Σ_{st} consists of the tgd $\forall x_1 \forall x_2 \big(E(x_1, x_2) \rightarrow E'(x_1, x_2)\big)$. Informally, D^* tells us to “copy” E to E'. Given a source instance S for D^* and a query Q over $\{E'\}$, it therefore seems natural to expect that the answer to Q is simply $Q(S')$, where S' is the “copy” of S over $\{E'\}$, that is, the instance S' with $(E')^{S'} = E^S$. For the simple existential query

$$Q(x) := \exists y \big(E'(x, y) \wedge \neg E'(y, x)\big),$$

and the source instance S^* with $E^{S^*} = \{(a, b)\}$, this means that the set of answers to Q with respect to S^* and D^* is $\{a\}$. But contrary to this expectation, $\mathsf{certain}^{D^*}(Q, S^*)$ is empty.

Two more examples, which point out more severe problems with the certain answers semantics, were pointed out in [Fagin et al. 2005a; Arenas et al. 2004]:

— *Rewritings of first-order queries do not exist, even in copying data exchange settings*. Copying data exchange settings are among the simplest possible data exchange settings. For example, the data exchange setting D^* from above is a copying data exchange setting. In general, a *copying data exchange setting* has the form $D = (\sigma, \tau, \Sigma_{\text{st}}, \emptyset)$, where $\tau = \{R' \mid R \in \sigma\}$, R' is a fresh relation symbol for each $R \in \sigma$, and $\Sigma_{\text{st}} = \{\forall \bar{x}\big(R(\bar{x}) \rightarrow R'(\bar{x})\big) \mid R \in \sigma\}$. So, informally, D just tells us to "copy" the source relations to the target relations. Therefore, as above, given a source instance S for D and a query Q over τ, we would intuitively expect that the answer to Q is simply $Q(S')$, where S' is the copy of S over τ (i.e., the target instance S' with $(R')^{S'} = R^S$ for every $R \in \sigma$). But as shown by Arenas et al. [2004], this does not hold for the certain answers semantics. Even more, they show that there is a copying data exchange setting $D = (\sigma, \tau, \Sigma_{\text{st}}, \emptyset)$ and an *existential* query Q over τ such that Q cannot be rewritten to an FO query Q' over τ such that for all source instances S for D we have $\mathsf{certain}^D(Q, S) = Q'(T)$, where T is the canonical solution or the core solution for S under D. An analogous result for LAV data exchange settings (which have no target dependencies, and contain only tgds with a single atom in the body) and a conjunctive query with one inequality was proved by Fagin et al. [2005a].

— *Too much uniformity in query answers*. Let $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a data exchange setting with $\Sigma_{\text{t}} = \emptyset$, and let Q be a non-trivial Boolean query over τ. Then we expect the answer to Q with respect to D to be true in some source instances for D, and false in others. But it was shown in [Arenas et al. 2004, Proposition 5.4] that either for all source instances S the certain answers of Q on S with respect to D are empty, or for all source instances S the certain answers of $\neg Q$ on S with respect to D are empty. So either Q or $\neg Q$ has a trivial answer (the empty set) that is input-independent.

It is natural to assume that the reason for such anomalies lies in some basic problems with the definition of solutions and query answering semantics. In fact, Fagin et al. [2005b] tried to remedy this partially by introducing a different semantics for answering queries based on universal solutions, but this semantics is prone to the above anomalies as well, cf. the full version of Arenas et al. [2004]. Our goal, therefore, is twofold:

(1) we would like to understand what causes these anomalies, and
(2) we would like to find natural notions of solutions and query answering that do not exhibit such anomalous behavior.

1.3. Reasons for Query Answering Anomalies

In our opinion, there are essentially two reasons for the anomalies of query answering in data exchange described above.

First, the basic notions in data exchange rely on the *open world assumption (OWA)* [Reiter 1978; 1984; Imielinski and Lipski, Jr. 1984; van der Meyden 1998]. The OWA is a general principle for dealing with "negative information" (i.e., information that is not supplied, e.g., by an instance, or by a data exchange setting and a source instance). Intuitively, under the OWA, facts that are not explicitly stated to be true or false, are not known to be true or false, and can therefore be either true or false. In data exchange, the OWA amounts to the following property: Given a data exchange setting $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, a source instance S for D, a solution T for S under D, and an instance T' over τ with $T \subseteq T'$, if T' satisfies all dependencies in Σ_{t}, then T' is a solution for S under D (since neither D nor S tells us that the tuples which are in T' and not in T are not part of any solution). This alone causes most of the anomalies mentioned above. For example, Proposition 5.4 in [Arenas et al. 2004] depends entirely on this property.

Second, although solutions may well contain nulls, the definition of the certain answers semantics applies a query Q to solutions as if solutions were instances without nulls. More precisely, Q is applied as if nulls were ordinary database values. But the dangers of treating nulls this way are very well known since the seminal paper by Imielinski and Lipski, Jr. [1984] (see also Abiteboul et al. [1995] and Date and Darwen [1997] for examples of anomalous behavior of the null-values semantics of SQL). Applying Q to an instance as if nulls were constants, distinct from constants that may occur elsewhere in the instance, is known as a *naive-tables*, or just *naive* semantics [Abiteboul et al. 1995; Imielinski and Lipski, Jr. 1984]. This semantics applies to positive queries, but is insufficient for properly representing answers to non-positive queries. Therefore, it should not be surprising that outside the class of positive queries, the certain answers semantics as defined in Eq. (1.1) exhibits anomalous behavior.

1.4. Main Contributions

To overcome the problems described above, we propose new query answering semantics that are based on the *closed world assumption (CWA)* [Reiter 1978; 1984; Imielinski and Lipski, Jr. 1984; van der Meyden 1998], and employ techniques for answering queries on instances with nulls [Lipski, Jr. 1979; Imielinski and Lipski, Jr. 1984; van der Meyden 1998]. Intuitively, the CWA here ensures that query answers depend only on data “moved” from the source instance to the target using the tgds and egds of the data exchange setting. This is the reason why, in our view, the CWA should be the preferable (although not exclusively so) assumption in data exchange. Also, it ensures that some of the anomalies described above do not arise for the new semantics. To obtain the new semantics, we restrict the set of solutions to those solutions that are “valid” under the CWA, and, depending on the semantics, we basically take the certain answers or the maybe answers (in the sense of Lipski, Jr. [1979]) with respect to the restricted set.[2] We have to be careful, though, since in general the restricted set of solutions will contain instances with nulls. Altogether, we obtain four different semantics.

The key step is to formalize an appropriate notion of *CWA-solution* that corresponds to solutions that are “valid” under the CWA. The main idea is that each fact in a CWA-solution must be directly justified by the source instance and the tgds and egds of the data exchange setting. Here, a fact is a simple statement expressible by a Boolean conjunctive query. More precisely, we have the following three informal requirements for CWA-solutions:

(1) Every atom[3] present in a CWA-solution must be “justified” by the source instance S and the tgds and egds of the data exchange setting D. Informally, an atom is justified if it can be inferred from S using the tgds and egds of D.
(2) Justifications for atoms should not be overused. That is, justifications for atoms do not justify more atoms than necessary. This requirement actually prevents excessive use of nulls.
(3) Each fact true in a CWA-solution logically follows from the source instance S and the tgds and egds of the data exchange setting D. That is, CWA-solutions should not “invent” new facts compared to what can be inferred from S using the tgds and egds in D.

[2] There are also more advanced forms of approximating answers that have been proposed in [Buneman et al. 1991; Buneman et al. 1991; Gunter 1992; Libkin 1998], but in this paper we concentrate on the two basic semantics mentioned above.

[3] Here we view instances as sets of atoms of the form $R(\bar{a})$. If an instance I contains an atom $R(\bar{a})$, this means that $\bar{a} \in R^I$. If I does not contain $R(\bar{a})$, this means that $\bar{a} \notin R^I$.

We characterize CWA-solutions as particular universal solutions, with the core solution being the unique minimal CWA-solution (up to renaming of nulls). In particular, CWA-solutions exist if and only if universal solutions exist. We also identify restricted kinds of data exchange settings, where a unique "maximal" CWA-solution is guaranteed to exist. In particular, the canonical solution is the unique "maximal" CWA-solution under data exchange settings without target dependencies. For general settings, however, such "maximal" CWA-solutions may not exist.

The existence of minimal and maximal CWA-solutions allows us to obtain simple characterizations of the four semantics under data exchange settings without target dependencies (and slight extensions thereof), namely as the certain answers or the maybe answers over the canonical solution or the core solution, depending on the semantics. Thus, for such data exchange settings, the problem of query answering is reduced to the well-studied problem of query answering over instances with nulls, while these instances are the canonical solution or the core solution, which we know well how to construct. For general data exchange settings, however, only two of the four semantics can be characterized in such a way.

We also address the problem of computing CWA-solutions. We obtain a data exchange setting $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, where $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$ consists entirely of tgds such that it is undecidable whether a given source instance S for D has a CWA-solution under D. As a consequence, the corresponding problem for universal solutions is undecidable, too. This also strengthens a corresponding result of Deutsch et al. [2008] on the existence of universal models. When restricting attention to the well-known class of *weakly acyclic* data exchange settings, however, known tractability results for universal solutions carry over to CWA-solutions. In particular, for such data exchange settings, CWA-solutions can be computed with polynomial time data complexity.

Finally, we consider the problem of query answering with respect to the four different query semantics. Considering *weakly acyclic* settings, we show that evaluating *unions of conjunctive queries* under two of the semantics is possible with polynomial time data complexity and can be PTIME-hard. Going beyond unions of conjunctive queries, we obtain that evaluation of (Boolean) *first-order queries* under the four semantics has co-NP (resp., NP) data complexity, provided that the underlying data exchange setting is *richly acyclic* (an acyclicity notion that is slightly more restrictive than the usual notion of *weak acyclicity*). Furthermore, there exist conjunctive queries with just one inequality, for which evaluating the query is co-NP-hard (resp., NP-hard).

1.5. Practical Aspects

Data exchange is an area where systems work was ahead of theoretical investigation: data exchange systems existed for a while (and are being worked on), with theoretical foundations arriving a few years later. In fact the main goal of early theoretical papers on data exchange was to offer insights into the semantics of query answering, and to justify – or suggest changes to – algorithms implemented in real-life systems. Our investigation follows this trend, and below we offer some comments on the practical aspects of the theoretical results shown here.

Our characterizations of the space of CWA-solutions further confirms the crucial role that the canonical solution and the core solution play in data exchange. In fact, we show that in many cases queries can be answered using these solutions, even when incompleteness of these solutions is properly taken into account. The results confirm the usual trade-off between these two solutions: while the canonical solution is probably slightly more natural for query answering under the CWA in settings without target constrains, it does not necessarily play a similar role when target constraints are present. All in all, the results suggest that a reasonable balance between these two standard solutions should be used (in line, for example, with a recent investigation of Mecca et al. [2009], which reached similar conclusions by analyzing time/space requirements for building these solutions).

Once a query Q is issued, certain answers to Q should be computed and given to the user. Note that the issue of non-rewritability goes away with the closed-world semantics; instead, one has to use techniques for computing certain answers to queries over database instances with incomplete information. Such instances, e.g., the core solution and the canonical solution, are naive tables. If certain answers are not sufficient for the user, maybe-answers should be computed to provide an upper approximation. No new materialization of the target is required for this purpose.

1.6. Outline

The paper is structured as follows: Section 2 fixes basic notation and definitions that are used throughout the paper. In Section 3, we introduce and illustrate the concept of CWA-solutions; we also identify some basic properties of CWA-solutions. The complexity of computing CWA-solutions is studied in Section 4. Section 5 presents the new CWA-solution-based query answering semantics, and argues that some of the anomalies mentioned above do not arise for the new semantics. Furthermore, Section 5 contains a comparison of the CWA-solution-based semantics with other semantics for query answering which appeared after the conference versions [Libkin 2006; Hernich and Schweikardt 2007] of this article. Section 6 studies the complexity of query answering under the CWA-solution-based semantics. Finally, Section 7 concludes the paper.

2. PRELIMINARIES

This section presents basic definitions that are used throughout the paper.

2.1. Database Instances

A *schema* σ is a *finite* set of relation symbols, each associated with an arity, and an *instance* I over σ assigns to each relation symbol $R \in \sigma$ a *finite* relation R^I of the same arity as R. We often identify I with the set $\{R(\bar{u}) \mid R \in \sigma,\ \bar{u} \in R^I\}$ of *atoms* $R(\bar{u})$ of I. The set of all values that occur in I is denoted by $\mathrm{dom}(I)$. We assume that each such value comes from one of the following two disjoint infinite sets: the set Const of *constant values*, and the set Null of *null values* (*nulls*, for short). Constants are typically denoted by lowercase letters $a, b, c, \ldots$, and nulls by $\perp$, possibly with sub/superscripts. We let $\mathrm{const}(I) := \mathrm{dom}(I) \cap \mathsf{Const}$ and $\mathrm{null}(I) := \mathrm{dom}(I) \cap \mathsf{Null}$. We also let $\mathsf{Dom} := \mathsf{Const} \cup \mathsf{Null}$.

The usual operations and notations for sets naturally carry over to instances. In particular, given instances I and J, the union of I and J, denoted by $I \cup J$, is the instance consisting of all atoms of I and all atoms of J; and I is contained in J, written $I \subseteq J$, if every atom of I is contained in J.

Given instances I and J, a *homomorphism* from I to J is a mapping $h\colon \mathrm{dom}(I) \to \mathrm{dom}(J)$ such that h is the identity on $\mathrm{const}(I)$, and for each atom $R(\bar{u})$ in I the atom $R(h(\bar{u}))$ is in J. [4] Here, for $\bar{u} = (u_1, \ldots, u_k)$ we let $h(\bar{u}) := (h(u_1), \ldots, h(u_k))$. An isomorphism from I and J is an injective homomorphism h such that h^{-1} is a homomorphism from J to I. Usually, we identify instances which are the same up to isomorphism. We say that I is *contained* in J if I is isomorphic to an instance $K \subseteq J$.

2.2. Queries and Dependencies

A *first-order query* (FO *query*, for short) over a schema σ is a first-order formula φ over the vocabulary $\sigma \cup \{c \mid c \in \mathsf{Const}\}$ (i.e., FO queries may contain constants), together

[4] Note the difference between this definition of homomorphism, and the one used in Libkin [2006]. Here, a homomorphism can map a null to a constant or to a null, but a homomorphism according to the definition in Libkin [2006] maps nulls always to nulls (and not to constants).

with a tuple $\bar{x} = (x_1, \ldots, x_k)$ that lists the free variables of φ. We denote such a query by $\varphi(\bar{x})$. FO queries are evaluated using the *active domain semantics* (see, e.g., Abiteboul et al. [1995]). That is, quantifiers range over the values that occur in the instance or the query. A *conjunctive query* over σ is an FO query over σ built entirely from relational atomic formulas over σ, using conjunction and existential quantification. *Unions of conjunctive queries* are disjunctions of conjunctive queries. CQ and UCQ respectively denote the set of conjunctive queries and the set of unions of conjunctive queries.

Given $\varphi(x_1, \ldots, x_k) = \bigwedge_{i=1}^{n} R_i(\bar{y}_i)$ and $\bar{u} = (u_1, \ldots, u_k) \in \mathsf{Dom}^k$, we will often write $\varphi(\bar{u})$ for the set of all atoms $R_i(\bar{v}_i)$, where $\bar{v}_i$ is obtained from $\bar{y}_i$ by replacing each variable x_i with u_i.

A *tuple generating dependency* (*tgd*, for short) is a formula of the form

$$\forall\bar{x}\forall\bar{y}\big(\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z})\big), \tag{2.1}$$

where $\varphi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ are conjunctions of relational atomic formulas. *Full tgds* are tgds of the form (2.1), where $\bar{z}$ is empty. An *equality generating dependency* (*egd*, for short) is a formula of the form

$$\forall\bar{x}\big(\varphi(\bar{x}) \to x_i = x_j\big), \tag{2.2}$$

where φ is a conjunction of relational atomic formulas, $\bar{x} = (x_1, \ldots, x_k)$ for some $k \geq 1$, and $i, j \in \{1, \ldots, k\}$. In the sequel, we omit the universal quantifiers in front of tgds and egds, and just write $\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z})$ for (2.1), and $\varphi(\bar{x}) \to x_i = x_j$ for (2.2).

2.3. Data Exchange Settings and Solutions

A *data exchange setting* $(\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ consists of disjoint schemas σ and τ, called *source schema* and *target schema*, respectively, a finite set Σ_{st} of *source-to-target dependencies*, and a finite set Σ_{t} of *target dependencies*. A data exchange setting $(\sigma, \tau, \Sigma_{\mathrm{st}}, \emptyset)$ without target dependencies is also denoted by $(\sigma, \tau, \Sigma_{\mathrm{st}})$. As source-to-target dependencies we will use *source-to-target tgds* (or *s-t tgds*), which are tgds of the form (2.1), where φ is a conjunction of relational atomic formulas over σ and ψ is a conjunction of relational atomic formulas over τ. As target dependencies we will use tgds over τ (called *target tgds*) together with egds over τ. Whenever we talk about a data exchange setting, we mean a data exchange setting that consists of this kind of source-to-target dependencies and target dependencies.

Let $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ be a data exchange setting. A *source instance* for D is an instance over σ, and a *target instance* for D is an instance over τ. We assume that source instances contain only constants (i.e., no nulls); target instances, on the other hand, may also contain nulls. A *solution* for S under D is a target instance T for D such that $S \cup T$ satisfies all s-t tgds in Σ_{st}, written $S \cup T \models \Sigma_{\mathrm{st}}$, and T satisfies all target tgds and egds in Σ_{t}, written $T \models \Sigma_{\mathrm{t}}$.

2.4. Universal Solutions, the Canonical Solution, and the Core

Universal solutions were introduced by Fagin et al. [2005a] as a formalization of "most general solutions". Let D be a data exchange setting, and let S be a source instance for D. A *universal solution* T for S under D is a solution for S under D such that for every solution T' for S under D there is a homomorphism from T to T'. For computing universal solutions, the well-known *chase procedure* [Beeri and Vardi 1984] can be used; see [Fagin et al. 2005a, Section 3.1].

Here we give only the most essential definitions regarding the chase, a detailed exposition can be found in Fagin et al. [2005a]. A tgd $\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z})$ can be *applied* to an instance I (with tuples $\bar{u}, \bar{v}$) if $I \models \varphi(\bar{u}, \bar{v})$ and for every tuple $\bar{w}$ we have $I \not\models \psi(\bar{u}, \bar{w})$. The result of this application is an instance J obtained from I by adding the atoms of $\psi(\bar{u}, \bar{w})$, where $\bar{w}$ is a tuple of pairwise distinct fresh nulls. An egd $\varphi(\bar{x}) \to x_i = x_j$ can be *applied* to I (with a tuple $\bar{u} = (u_1, \ldots, u_{|\bar{x}|})$) if $I \models \varphi(\bar{u})$ and $u_i \neq u_j$. If at least one of u_i and u_j, say u_i, is a null, then the result of this application is an instance J obtained from I by replacing every occurrence of u_i with u_j; if both u_i and u_j are constants, the application is said to *fail*. A *chase sequence of I with* Σ is a (finite or infinite) sequence $C = (I_0, I_1, \ldots)$ of instances such that $I_0 = I$, and each instance I_{i+1} is the result of *applying* a tgd or an egd in Σ to I_i. If C is finite, its *result* is the last instance in C. C is *complete* if it is finite, and no tgd and no egd can be (successfully) applied to its result. C is *successful* if it is complete and its result satisfies Σ. Finally, C is *failing* if it is finite and its result does not satisfy Σ. If $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ is a data exchange setting, S is a source instance for D, and $S \cup T$ is the result of any successful chase sequence of S with $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$, then T is a universal solution for S under D [Fagin et al. 2005a].

Two universal solutions play a special role in data exchange: the *canonical solution* [Fagin et al. 2005a], and the *core solution* [Fagin et al. 2005b].

We first recall the definition of the *canonical solution*, for data exchange settings without target dependencies. The following definition of the canonical solution is from Arenas et al. [2004]. Let $D = (\sigma, \tau, \Sigma_{\mathrm{st}})$ be a data exchange setting without target dependencies, and let S be a source instance for D. For each s-t tgd in Σ_{st} of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and for each pair of tuples $\bar{u}, \bar{v}$ such that $S \models \varphi(\bar{u}, \bar{v})$, let $\bar{\bot}$ be a tuple of pairwise distinct fresh nulls so that $|\bar{\bot}| = |\bar{z}|$, and add the atoms of $\psi(\bar{u}, \bar{\bot})$ to the target. (We recall that ψ is a conjunction of atomic formulas.) The result is the *canonical solution* for S under D, denoted by $\textsc{CanSol}_D(S)$. Notice that $S \cup \textsc{CanSol}_D(S)$ is the result of the *oblivious chase* (see, e.g., Calì et al. [2008]) of S with Σ_{st}.

For example, given the data exchange setting $D = (\{E\}, \{R\}, \Sigma_{\mathrm{st}})$, where $\Sigma_{\mathrm{st}} = \{E(x, y) \to \exists z\, R(x, z)\}$, and the source instance $S = \{E(a, b_1), E(a, b_2)\}$, we have $\textsc{CanSol}_D(S) = \{R(a, \bot_1), R(a, \bot_2)\}$.

Next we give the definition of the *core solution*. A *core* of an instance I is an instance $J \subseteq I$ such that there is a homomorphism from I to J, but there is no homomorphism from J to any instance $K \subsetneq J$. Some basic properties of cores are:

Theorem 2.1 ([Hell and Nešetřil 1992; Fagin et al. 2005b]).

(*1*) *Every instance has a core.*
(*2*) *If I_1 and I_2 are homomorphically equivalent instances, J_1 is a core of I_1, and J_2 is a core of I_2, then J_1 and J_2 are isomorphic. In particular, any two cores of an instance are isomorphic.*
(*3*) *If J is a core of I, then there is a homomorphism from I to J that is the identity on* $\mathrm{dom}(J)$. *In particular, J is a homomorphic image of I.*

By Theorem 2.1, we can speak of *the* core of an instance. In Fagin et al. [2005b] it was shown that, if universal solutions for S under D exist, then there is a universal solution for S under D, denoted by $\textsc{Core}_D(S)$, that is isomorphic to the core of *every* universal solution for S under D. In the previous example, both $\{R(a, \bot_1)\}$ and $\{R(a, \bot_2)\}$ are cores of $\textsc{CanSol}_D(S)$; of course they are isomorphic so we can say that $\{R(a, \bot)\}$ is the core of $\textsc{CanSol}_D(S)$, or equivalently, of the universal solutions for S.

2.5. Instances with Incomplete Information

We review the most important definitions from [Imielinski and Lipski, Jr. 1984; Abiteboul et al. 1995] on instances with incomplete information. Instances with nulls are *instances with incomplete information*. Nulls are treated as “unknown” (as opposed to “nonexistent”) values [Zaniolo 1984]: for each null, we know that there is a constant that can be substituted for that null, but we do not know which constant. Thus, an instance I with nulls represents a number of *complete* instances (without nulls) obtained from I by assigning constants to the nulls in I.

More precisely, let a *valuation of* I be a mapping $v\colon \mathrm{null}(I) \to \mathsf{Const}$. Given an instance I with incomplete information and a valuation v of I, let $v(I)$ be the instance obtained from I by replacing, for every $\bot \in \mathrm{null}(I)$, each occurrence of $\bot$ in I by $v(\bot)$. We then define the set of all instances represented by I as

$$\mathit{Rep}(I) := \{v(I) \mid v \text{ is a valuation of } I\}.$$

Note that $\mathit{Rep}(I)$ is a potentially infinite object. For example, if $I = \{R(\bot)\}$, then $\mathit{Rep}(I) = \{\{R(c)\} \mid c \in \mathsf{Const}\}$. For a set Σ of constraints over I's schema, we set

$$\mathit{Rep}_\Sigma(I) := \{\hat{I} \in \mathit{Rep}(I) \mid \hat{I} \models \Sigma\}.$$

In order to evaluate a query Q on an instance I with incomplete information (where Q comes from a language that works on instances without nulls, e.g., an FO query), one normally considers the set $\{Q(\hat{I}) \mid \hat{I} \in \mathit{Rep}(I)\}$ as the result of Q on I, or $\{Q(\hat{I}) \mid \hat{I} \in \mathit{Rep}_\Sigma(I)\}$ in the context of a set Σ of constraints on I. To represent this set (even for an FO query Q), one needs rather complicated *conditional tables* [Imielinski and Lipski, Jr. 1984]. Instead of exact representation, one often prefers to use lower and upper approximations, namely the *certain* answers and the *maybe* answers, defined as follows:

— *the certain answers of* Q *on* I *w.r.t.* Σ: $\Box_\Sigma Q(I) := \bigcap \{Q(\hat{I}) \mid \hat{I} \in \mathit{Rep}_\Sigma(I)\}$
— *the maybe answers of* Q *on* I *w.r.t.* Σ: $\Diamond_\Sigma Q(I) := \bigcup \{Q(\hat{I}) \mid \hat{I} \in \mathit{Rep}_\Sigma(I)\}$

That is, the certain answers $\Box_\Sigma Q(I)$ contain tuples that are present in the answer to Q no matter which values are assigned to the nulls in I. The maybe answers $\Diamond_\Sigma Q(I)$ contain tuples present in at least one answer to Q for some assignment of values to the nulls in I. Notice that $\Box_\Sigma Q(I)$ is a finite object (since it is contained in $Q(v(I))$ for every valuation v of I with $v(I) \models \Sigma$), but $\Diamond_\Sigma Q(I)$ may well be infinite, and thus some finite representation of it needs to be found.

Note that these certain and maybe answers should not be confused with certain and maybe answers that arise in data exchange: here our only source of incompleteness is nulls in instances, while in data exchange the main source of incompleteness is the existence of multiple target instances.

3. CWA-SOLUTIONS

In this section, we formalize the requirements for CWA-solutions presented in the introduction, and identify basic properties of CWA-solutions.

Let us recall the requirements for CWA-solutions presented in Section 1:

(1) Each atom in a CWA-solution must be justified by the source instance and the tgds and egds of the data exchange setting.
(2) Justifications for atoms should not be overused. That is, justification for atoms do not justify more atoms than necessary.
(3) Each fact true in a CWA-solution logically follows from the source instance S and the tgds and egds of the data exchange setting D. That is, CWA-solutions should not "invent" new facts compared to what can be inferred from S using the tgds and egds in D.

In Section 3.1 and Section 3.2, we formalize the first two requirements by defining *CWA-presolutions* (which intuitively satisfy these requirements). Section 3.1 deals with data exchange settings without target dependencies. It serves as a warm-up for

Section 3.2, where we deal with the more involved case of data exchange settings with target dependencies. CWA-solutions are introduced in Section 3.3.

3.1. CWA-Presolutions for Data Exchange Settings Without Target Dependencies

We aim to define *CWA-presolutions*, a formalization of solutions that satisfy the first two requirements for CWA-solutions, for data exchange settings without target dependencies.

Let $D = (\sigma, \tau, \Sigma_{\mathrm{st}})$ be a data exchange setting without target dependencies, and let S be a source instance for D.

Informally, an atom in a solution for S under D is justified if it can be obtained from S by applying an s-t tgd in Σ_{st}. Let a *justification for an atom with respect to S and D* consist of

— an s-t tgd $d \in \Sigma_{\mathrm{st}}$ of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and
— tuples $\bar{u}$, $\bar{v}$ over Dom such that $S \models \varphi(\bar{u}, \bar{v})$.

We denote such a justification by $(d, \bar{u}, \bar{v})$. It tells us that $\varphi(\bar{u}, \bar{v})$ is satisfied in S, so that any solution T for S under D must satisfy $\psi(\bar{u}, \bar{w})$ for some tuple $\bar{w}$ over Dom. So, intuitively, $(d, \bar{u}, \bar{v})$ can be used to *justify* the atoms of $\psi(\bar{u}, \bar{w})$ for any tuple $\bar{w}$ over Dom of the appropriate length. Let $\mathcal{J}_{D,S}$ be the set of all justifications for atoms with respect to S and D.

Let T be a solution for S under D. For T to be a CWA-presolution, we require each atom of T to be assigned to some justification in $\mathcal{J}_{D,S}$ (requirement 1). More precisely, for each atom $A \in T$ there must be a justification $(d, \bar{u}, \bar{v}) \in \mathcal{J}_{D,S}$ with d of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and an assignment $\bar{w}$ for the variables in $\bar{z}$ such that T satisfies $\psi(\bar{u}, \bar{w})$, and A is one of the atoms of $\psi(\bar{u}, \bar{w})$. Concerning requirement 2, each $(d, \bar{u}, \bar{v}) \in \mathcal{J}_{D,S}$ must be associated to the atoms of $\psi(\bar{u}, \bar{w})$ for *at most one* tuple $\bar{w}$. So, in fact, for each justification $j = (d, \bar{u}, \bar{v}) \in \mathcal{J}_{D,S}$ with d of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ there must be *exactly one* tuple $\bar{w}_j$ so that j justifies the atoms of $\psi(\bar{u}, \bar{w}_j)$. In other words, T is the union of $\psi(\bar{u}, \bar{w}_j)$ as $j = (d, \bar{u}, \bar{v})$ ranges over all justifications in $\mathcal{J}_{D,S}$ and $d = \varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$.

This leads to the following definition of a CWA-presolution. Let $\alpha\colon \mathcal{J}^*_{D,S} \to \mathsf{Dom}$, where $\mathcal{J}^*_{D,S}$ is the set of all pairs (j, z) consisting of a justification $j = (d, \bar{u}, \bar{v}) \in \mathcal{J}_{D,S}$ with d of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and a variable z in $\bar{z}$. For each $j = (d, \bar{u}, \bar{v}) \in \mathcal{J}_{D,S}$ with $d = \varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, let $\mathcal{A}_\alpha(j) := \psi(\bar{u}, \bar{w}_j)$, where $\bar{w}_j$ is obtained from $\bar{z}$ by replacing each variable z in $\bar{z}$ with $\alpha(j, z)$.

Definition 3.1 (*CWA-presolution*). A *CWA-presolution* for S under D is a target instance T for D such that there is a mapping $\alpha\colon \mathcal{J}^*_{D,S} \to \mathsf{Dom}$ with $T = T_{D,\alpha}(S)$, where $T_{D,\alpha}(S) := \bigcup_{j \in \mathcal{J}_{D,S}} \mathcal{A}_\alpha(j)$.

Example 3.2. Recall the data exchange setting D_{conf} and the source instance S from Example 1.1. Let D'_{conf} be D_{conf} without the target dependencies. Then

$$\begin{aligned}\textsc{Core}_{D'_{\mathrm{conf}}}(S) = \big\{&\textit{Paper}(1, t_1, \bot_1), \textit{Paper}(2, t_2, \bot_2), \textit{PC}'(\bot_3, n_1), \textit{PC}'(\bot_4, n_2),\\ &\textit{PC}'(\bot_5, n_3), \textit{Assign}(1, \bot_3), \textit{Assign}(2, \bot_4), \textit{Assign}(3, \bot_5)\big\}\end{aligned}$$

is a CWA-presolution for S under D'_{conf}. It is “generated” by a mapping α with $\alpha(d_1, (i, t_i), (), z) = \bot_i$ for each $i \in \{1, 2\}$, and $\alpha(d_2, (n_i, i), (a), z) = \bot_{i+2}$ for each $i \in \{1, 2, 3\}$ and values a. That is, $T_{D'_{\mathrm{conf}},\alpha}(S) = \textsc{Core}_{D'_{\mathrm{conf}}}(S)$.

It is also easy to see that the target instance

$$T = \{\textit{Paper}(1, t_1, \bot_1), \textit{Paper}(2, t_2, \bot_2), \textit{PC}'(\bot_3, n_1), \textit{PC}'(\bot_3, n_2), \textit{PC}'(\bot_5, n_3), \\ \textit{Assign}(1, \bot_3), \textit{Assign}(2, \bot_3), \textit{Assign}(3, \bot_5)\},$$

which assigns the same ID $\bot_3$ to PC members n_1 and n_2, is a CWA-presolution for S under D'_{conf}. □

Notice that every CWA-presolution for S under D is a solution for S under D. Moreover, as an immediate consequence of the definitions, we obtain:

PROPOSITION 3.3. *Every CWA-presolution for S under D is a homomorphic image of* $\textsc{CanSol}_D(S)$. *More precisely, for every injective $\alpha\colon \mathcal{J}^*_{D,S} \to$ Null and every $\alpha'\colon \mathcal{J}^*_{D,S} \to$ Dom, the mapping $h\colon \operatorname{dom}(T_{D,\alpha}(S)) \to \operatorname{dom}(T_{D,\alpha'}(S))$ with*

$$h(u) := \begin{cases} \alpha'(j,z), & \text{if } u = \alpha(j,z) \text{ for some } (j,z) \in \mathcal{J}^*_{D,S}, \\ u, & \text{otherwise} \end{cases}$$

is a homomorphism from $T_{D,\alpha}(S)$ to $T_{D,\alpha'}(S)$ with $h(T_{D,\alpha}(S)) = T_{D,\alpha'}(S)$.

In particular, $\textsc{CanSol}_D(S)$ itself is a CWA-presolution for S under D.

3.2. CWA-Presolutions for Data Exchange Settings With Target Dependencies

We now extend the definitions to deal with data exchange settings containing target dependencies. Notice that we indeed have to do so, because a CWA-presolution for S under the reduced data exchange setting $D' = (\sigma, \tau, \Sigma_{\text{st}})$ without target dependencies is not necessarily a solution for S under D. For example, we have shown in Example 3.2 that $\textsc{Core}_{D'_{\text{conf}}}(S)$ is a CWA-solution for S under D'_{conf}, but $\textsc{Core}_{D'_{\text{conf}}}(S)$ is no solution for S under D_{conf}, since it does not satisfy tgd d_3 (cf., Example 1.1).

Let $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a data exchange setting with target dependencies, and let S be a source instance for D.

The idea for the definition of CWA-presolutions is as follows. First of all, each atom in such a solution must be justified (requirement 1). Informally, an atom in a solution for S under D is justified if it can be obtained from S using the tgds in Σ_{st} and Σ_{t}: either the atom can be obtained as in Section 3.1 by applying an s-t tgd to S, or it can be obtained by applying a target tgd to already justified atoms. Note that we do not take into account egds here; these will be incorporated later. We have to be careful, though, to avoid "circular justifications": we do not want a tgd $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ and tuples $\bar{u}, \bar{v}$ to justify the atoms in $\psi(\bar{u}, \bar{w})$, while another tgd, applied to the atoms in $\psi(\bar{u}, \bar{w})$, justifies the atoms in $\varphi(\bar{u}, \bar{v})$. Thus, we require that the atom can be obtained by a sequence of applications of s-t tgds and target tgds, where each target tgd is applied to atoms obtained by an earlier application of a tgd in the sequence. As in Section 3.1, to satisfy requirement 2, each tgd $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ should be applied at most once for each pair $\bar{u}, \bar{v}$ of tuples giving values to $\bar{x}, \bar{y}$, respectively.

To formalize this accordingly, we employ a suitably "controlled" version of the chase procedure, which we call *α-chase*.

A *potential justification* for an atom with respect to S and D consists of

— a tgd d in $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and
— tuples $\bar{u}, \bar{v}$ over Dom of length $\bar{x}$ and $\bar{y}$, respectively,

and is denoted by $(d, \bar{u}, \bar{v})$.[5] Intuitively, $(d, \bar{u}, \bar{v})$ can be used to *justify* the atoms of $\psi(\bar{u}, \bar{w})$ for any tuple $\bar{w}$ over Dom of the appropriate length, provided the atoms of $\varphi(\bar{u}, \bar{v})$ are already justified. Let $\mathcal{J}_D$ be the set of all potential justifications for atoms with respect to S and D.

Now, as in Section 3.1, we assign values to the existentially quantified variables of tgds for each justification by a mapping $\alpha\colon \mathcal{J}_D^* \to \mathsf{Dom}$, where $\mathcal{J}_D^*$ is the set of all pairs (j, z) consisting of a justification $j = (d, \bar{u}, \bar{v}) \in \mathcal{J}_D$ and an existentially quantified variable z in d. For each $j = (d, \bar{u}, \bar{v}) \in \mathcal{J}_D$ with d of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, let $\mathcal{A}_\alpha(j) := \psi(\bar{u}, \bar{w}_j)$, where $\bar{w}_j$ is obtained from $\bar{z}$ by replacing each variable z in $\bar{z}$ by the value $\alpha(j, z)$.

Definition 3.4 (α*-chase sequence*). Let $\alpha\colon \mathcal{J}_D^* \to \mathsf{Dom}$.

(1) An α*-chase sequence* of S with D is a (finite or infinite) sequence $I_0, I_1, I_2, \ldots$ of instances such that $I_0 = S$, and each I_{i+1} is obtained via an α*-application* of a tgd in $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$ to I_i as follows:

Let d be a tgd in $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$ of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, and let $\bar{u}, \bar{v}$ be tuples over Dom. We say that d *can be* α*-applied to* I_i *with* $(\bar{u}, \bar{v})$ if $I_i \models \varphi(\bar{u}, \bar{v})$, and I_i does not contain all the atoms in $\mathcal{A}_\alpha(j)$ yet. The *result* of α-applying d to I_i with $(\bar{u}, \bar{v})$ is the instance $I_i \cup \mathcal{A}_\alpha(j)$.

(2) The *result* of a finite α-chase sequence $C = (I_0, \ldots, I_m)$ of I with D is I_m.

(3) An α-chase sequence C of I with D is *successful* if C is finite, and no tgd d in $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$ can be α-applied to the result of C for no pair $(\bar{u}, \bar{v})$ of tuples over Dom of the appropriate length.

Note that the definition of an α-chase sequence given here differs from the one in Hernich and Schweikardt [2007]. In particular, here we do not consider egds. It is also important to note that, unlike the notion of the chase [Beeri and Vardi 1984] which is used in data exchange to compute universal solutions primarily, an α-chase sequence is used to show that all the atoms in a given solution are justified according to requirement 1 and 2.

Example 3.5. Recall the data exchange setting D_{conf} and the source instance S from Example 1.1. Let $\alpha'\colon \mathcal{J}_D^* \to \mathsf{Dom}$ be an extension of the mapping α from Example 3.2 such that $\alpha'(d_3, (i, \bot_{i+2}), (), z'') = n_i$ for each $i \in \{1, 2, 3\}$, $\alpha'(d_3, (i, \bot_{i+2}), (), z) = t_i$ and $\alpha'(d_3, (i, \bot_{i+2}), (), z') = \bot_i$ for each $i \in \{1, 2\}$, $\alpha'(d_3, (3, \bot_5), (), z) = \bot_6$, and $\alpha'(d_3, (3, \bot_5), (), z') = \bot_7$ Then there is a successful α'-chase sequence of S with D_{conf} whose result is $S \cup \textsc{Core}_{D_{\mathrm{conf}}}(S)$, where

$$\begin{aligned}\textsc{Core}_{D_{\mathrm{conf}}}(S) = \{\, & \mathit{Paper}(1, t_1, \bot_1), \mathit{Paper}(2, t_2, \bot_2), \mathit{Paper}(3, \bot_6, \bot_7),\\ & \mathit{PC}'(\bot_3, n_1), \mathit{PC}'(\bot_4, n_2), \mathit{PC}'(\bot_5, n_3),\\ & \mathit{Assign}(1, \bot_3), \mathit{Assign}(2, \bot_4), \mathit{Assign}(3, \bot_5)\,\}.\end{aligned}$$

Indeed, we can α'-apply the s-t tgds in the first five steps to generate the core of S under the reduced data exchange setting D'_{conf} from Example 3.2. That is, we could first α'-apply d_1 with $\big((i, t_1), ()\big)$ resulting in the atom $\mathit{Paper}(1, t_1, \bot_1)$. Similarly, we could apply the remaining s-t tgds. Finally, note that only one target tgd can be α'-applied to $\textsc{Core}_{D'_{\mathrm{conf}}}(S)$, namely d_3 with $\big((3, \bot_5), ()\big)$, resulting in the atoms $\mathit{Paper}(3, \bot_6, \bot_7)$ and $\mathit{PC}'(\bot_5, n_3)$. □

[5] Note that potential justifications differ from justifications defined in Section 3.1 in that $\varphi(\bar{u}, \bar{v})$ does not need to be satisfied.

The following lemma summarizes some basic properties of α-chase sequences. The straightforward and easy proof is left to the reader.

PROPOSITION 3.6. *Let* $\alpha\colon \mathcal{J}_D^* \to \mathsf{Dom}$.

(*1*) *A successful* α*-chase sequence of* S *with* D *exists if and only if there is no infinite* α*-chase sequence of* S *with* D*.*
(*2*) *If* C_1, C_2 *are successful* α*-chase sequences of* S *with* D*, then* C_1 *and* C_2 *have the same result.*

We are now ready to give the definition of CWA-presolution.

Definition 3.7 (*CWA-presolution*). A *CWA-presolution for* S *under* D is a solution T for D such that $S \cup T$ is the result of a successful α-chase sequence of S with D for some $\alpha\colon \mathcal{J}_D^* \to \mathsf{Dom}$.

In Example 3.5, $\mathrm{CORE}_{D_{\text{conf}}}(S)$ is a CWA-presolution for S under D_{conf} since it is a solution for S under D_{conf}, and, as shown in Example 3.5, $S \cup \mathrm{CORE}_{D_{\text{conf}}}(S)$ is the result of a successful α'-chase sequence of S with D.

Note that by requiring a CWA-presolution to be a solution we insist that α is chosen in such a way that the egds are not violated. Note also that CWA-presolutions as defined in Section 3.1 and CWA-presolutions as defined here coincide with respect to data exchange settings without target dependencies. An equivalent definition of CWA-presolutions in terms of a *game* can be found in Hernich and Schweikardt [2010].

3.3. CWA-Solutions

CWA-presolutions can generate certain facts in the target which, intuitively, do not follow from the source instance and the tgds and egds of the data exchange setting. In Example 3.2, the CWA-presolution T tells us that the PC members n_1 and n_2 are assigned to the same ID. However, the fact that those PC members have the same ID intuitively does not follow from S and the s-t tgds in D'_{conf}. The third requirement for CWA-solutions ensures that such “invented” facts do not occur in a CWA-solution.

Let $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a data exchange setting, and let S be a source instance for D. A *fact* F (over D’s target schema τ) is a Boolean conjunctive query $\exists \bar{z}\, \varphi(\bar{z})$ over τ. For example, if D is as in Example 1.1, then the fact

$$\exists z \big(\boldsymbol{PC}'(z, n_1) \wedge \boldsymbol{PC}'(z, n_2) \big) \tag{3.1}$$

tells us that the PC members n_1 and n_2 have the same ID. A fact $F = \exists \bar{z}\, \varphi(\bar{z})$ is true in a target instance T if it evaluates to true on T, i.e., if there is a tuple $\bar{v}$ of values (constants or nulls) such that T satisfies $\varphi(\bar{v})$. A CWA-solution for S under D is then a CWA-presolution in which every true fact follows from S and the tgds and egds of D.

Definition 3.8 (*CWA-solution*). A *CWA-solution* for S under D is a CWA-presolution T for S under D such that each fact true in T is also true in every solution for S under D. The set of all CWA-solutions for S under D is denoted by $[\![S]\!]^D_{\text{CWA}}$.

We almost immediately obtain the following characterization of CWA-solutions.

THEOREM 3.9. *Let* D *be a data exchange setting, and let* S *be a source instance for* D*. Then for every target instance* T *for* D*, the following are equivalent:*

(*1*) T *is a CWA-solution for* S *under* D*.*
(*2*) T *is both a universal solution and a CWA-presolution for* S *under* D*.*

PROOF. *1* $\Longrightarrow$ *2:* Let T be a CWA-solution for S under D. Then T is a CWA-presolution for S under D. In order to prove 2, it suffices therefore to show that T is a universal solution for S under D.

Let $\bot_1, \dots, \bot_k$ be an enumeration of all the nulls in T (without repetition). Consider the fact $F_T = \exists \bar{z}\, \varphi_T(\bar{z})$, where $\bar{z} = (z_1, \dots, z_k)$, and $\varphi_T(\bar{z})$ is the conjunction of all atoms $R(\bar{u})$ that can be obtained from an atom $R(\bar{t}) \in T$ by replacing each null $\bot_i$ in $\bar{t}$ with z_i. Clearly, F_T is true in T. Since T is a CWA-solution for S, F_T is true in every solution T' for S under D.

Now let T' be an arbitrary solution for S under D. Since F_T is true in T', there is a tuple $\bar{v} = (v_1, \dots, v_k)$ of values such that T' satisfies $\varphi_T(\bar{v})$. In other words, the mapping $h\colon \mathrm{dom}(T) \to \mathrm{dom}(T')$ given by $h(\bot_i) = v_i$ for all $i \in \{1, \dots, k\}$, and $h(a) = a$ for all $a \in \mathrm{const}(T)$ is a homomorphism from T to T'. It follows that for every solution T' for S under D there is a homomorphism from T to T', which shows that T is a universal solution for S under D.

2 $\Longrightarrow$ 1: Suppose T is a universal solution for S under D, and that T is a CWA-presolution for S under D. Since T is already a CWA-presolution for S under D, it remains to show that every fact that is true in T, is true in every solution T' for S under D. Let $F = \exists \bar{z}\, \varphi(\bar{z})$ be a fact that is true in T, and let T' be an arbitrary solution for S under D. In particular, there is a tuple $\bar{v}$ such that $\varphi(\bar{v})$ is true in T. Moreover, since T is a universal solution for S under D, there is a homomorphism h from T to T'. Since φ is preserved under homomorphisms, $\varphi(h(\bar{v}))$ is true in T'. Thus, F is true in T'. □

Theorem 3.9 will be very useful throughout this paper. For example, it can be applied to prove that certain solutions are CWA-solutions or not:

Example 3.10. Recall the data exchange setting D_{conf} and the source instance S from Example 1.1. As pointed out at the end of Section 3.2, $\textsc{Core}_{D_{\text{conf}}}(S)$ is a CWA-presolution for S under D_{conf}. Since $\textsc{Core}_{D_{\text{conf}}}(S)$ is also a universal solution for S under D_{conf}, it follows from Theorem 3.9 that $\textsc{Core}_{D_{\text{conf}}}(S)$ is a CWA-solution for S under D_{conf}.

Moreover, in Example 3.2, T is a CWA-presolution for S under the reduced data exchange setting D'_{conf}. However, it is *no* CWA-solution for S under D'_{conf}, since there is no homomorphism from T to $\textsc{Core}_{D'_{\text{conf}}}(S)$. One can see this also by observing that the fact (3.1) is true in T, but not in $\textsc{Core}_{D'_{\text{conf}}}(S)$.

Finally, let T be the union of two isomorphic copies T_1, T_2 of $\textsc{CanSol}_{D'_{\text{conf}}}(S)$ with $\mathrm{null}(T_1) \cap \mathrm{null}(T_2) = \emptyset$. Then, T is a universal solution for S under D'_{conf}, but by Proposition 3.3, no CWA-presolution for S under D'_{conf}. □

By Theorem 3.9, CWA-solutions are particular universal solutions. The following theorem states that the minimal universal solution – the core solution – is one of those CWA-solutions. In particular, it shows that the core solution is the unique minimal CWA-solution.

THEOREM 3.11. *Let D be a data exchange setting, and let S be a source instance for D such that $\textsc{Core}_D(S)$ exists. Then,*

(*1*) $\textsc{Core}_D(S)$ *is a CWA-solution for S under D.*
(*2*) *If T is a CWA-solution for S under D, then T contains* $\textsc{Core}_D(S)$.

PROOF. It suffices to prove 1, since this immediately implies 2 using Theorem 2.1(3) and Theorem 3.9. Since $\textsc{Core}_D(S)$ is a universal solution for S under D, all that is left to show is that $\textsc{Core}_D(S)$ is a CWA-*pre*solution for S under D.

As a first step, we inductively construct partial mappings $\alpha_i\colon \mathcal{J}^*_D \to \mathsf{Dom}$ and sequences $C_i = (I_0, I_1, \dots, I_i)$ such that C_i is an α'_i-chase sequence of S with D for every extension α'_i of α_i (i.e., α'_i coincides with α_i on all elements in $\mathcal{J}^*_D$ for which α_i is defined) and $I_i \subseteq S \cup \textsc{Core}_D(S)$. We let α_0 be undefined on all elements in $\mathcal{J}^*_D$, and $C_0 := (I_0)$, where $I_0 := S$.

Assume that α_i is a partial mapping from $\mathcal{J}^*_D$ to Dom, and $C_i = (I_0, \dots, I_i)$ is an α'_i-chase sequence of S with D for every extension α'_i of α_i such that $I_i \subseteq S \cup \textsc{Core}_D(S)$. If I_i satisfies all tgds and egds in $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$, then the construction stops. Otherwise, there is some $d \in \Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ that is not satisfied in I_i. Note that if d is an egd, then, since $I_i \subseteq S \cup \textsc{Core}_D(S)$ and d is over the target schema of D, $\textsc{Core}_D(S)$ does not satisfy d. But this is impossible, hence d is a tgd.

Say, d has the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ with $\bar{z} = (z_1, \dots, z_k)$. Then there are tuples $\bar{u}, \bar{v}$ with $I_i \models \varphi(\bar{u}, \bar{v})$, and $I_i \not\models \psi(\bar{u}, \bar{w})$ for every tuple $\bar{w}$. Since $I_i \subseteq S \cup \textsc{Core}_D(S)$ and $S \cup \textsc{Core}_D(S) \models d$, there is some $\bar{w}$ with $\textsc{Core}_D(S) \models \psi(\bar{u}, \bar{w})$. Assuming $\bar{w} = (w_1, \dots, w_k)$, we define $\alpha_{i+1} \colon \mathcal{J}^*_D \to \text{Dom}$ by

$$\alpha_{i+1}(j) := \begin{cases} w_i, & \text{if } j = (d, \bar{u}, \bar{v}, z_i) \\ \alpha_i(j), & \text{if } j \neq (d, \bar{u}, \bar{v}, z_i) \text{ and } \alpha_i(j) \text{ is defined,} \\ \text{undefined}, & \text{otherwise,} \end{cases}$$

and we let $C_{i+1} := (I_0, I_1, \dots, I_i, I_{i+1})$, where $I_{i+1} := I_i \cup \psi(\bar{u}, \bar{w})$. Clearly, C_{i+1} is an α'_{i+1}-chase sequence of S with D for every extension α'_{i+1} of α_{i+1}. Furthermore, the choice of $\bar{w}$ guarantees $I_{i+1} \subseteq S \cup \textsc{Core}_D(S)$.

Since $\textsc{Core}_D(S)$ is finite, and each α_i-application of a tgd produces at least one new atom, we have $I_i \models \Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ for some $i \leq |\textsc{Core}_D(S)|$. It is easy to extend α_i to a total mapping $\alpha \colon \mathcal{J}^*_D \to \text{Dom}$ such that C_i is a *successful* α-chase sequence of S with D.[6] Thus, the target instance T with $I_i = S \cup T$ is a CWA-presolution for S under D. By construction, we have $T \subseteq \textsc{Core}_D(S)$. On the other hand, T cannot be a proper subinstance of $\textsc{Core}_D(S)$: otherwise, there would be a homomorphism from $\textsc{Core}_D(S)$ to a proper subinstance of $\textsc{Core}_D(S)$, namely T (since $\textsc{Core}_D(S)$ is a universal solution for S under D), contradicting that $\textsc{Core}_D(S)$ is a core. So, we have $T = \textsc{Core}_D(S)$, which proves that $\textsc{Core}_D(S)$ is a CWA-presolution for S under D. □

Together with Theorem 3.9, this immediately yields:

COROLLARY 3.12. *For every data exchange setting D, and every source instance S for D, the following statements are equivalent:*

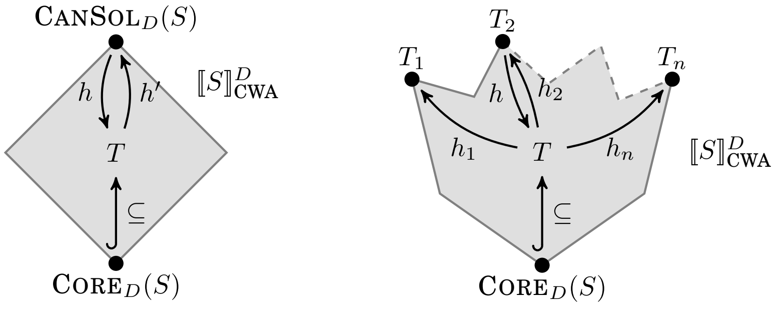
(*1*) *There exists a CWA-solution for S under D.*
(*2*) *There exists a universal solution for S under D.*
(*3*) $\textsc{Core}_D(S)$ *exists.*

In some cases, we even have a CWA-solution T that is *maximal* in the sense that for every CWA-solution T' for S under D there is a homomorphism h from T to T' with $h(T) = T'$. This is true, for instance, if we restrict attention to data exchange settings without target dependencies. In this case, Proposition 3.3 and Theorem 3.9 immediately yield:

PROPOSITION 3.13. *Let D be a data exchange setting without target dependencies, and let S be a source instance for D. Then,* $\textsc{CanSol}_D(S)$ *is the unique maximal CWA-solution for S under D.*

Proposition 3.13 can be extended to a slightly larger class of data exchange settings, namely to data exchange settings $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, where Σ_{t} consists of egds, or all

[6] If α_i is undefined for $(j, z) \in \mathcal{J}^*_D$, where $j = (d, \bar{u}, \bar{v})$, $d = \varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ and $I_i \models \varphi(\bar{u}, \bar{v})$, we pick a tuple $\bar{w}$ such that $I_i \models \psi(\bar{u}, \bar{w})$, and let $\alpha(j, z)$ be the value assigned to z by $\bar{w}$; for all other $(j, z) \in \mathcal{J}^*_D$ for which α_i is undefined, we can define $\alpha(j, z)$ arbitrarily.



(a) no target dependencies (b) with target dependencies

Fig. 1. A representation of the set $[\![S]\!]^D_{\mathrm{CWA}}$ of all CWA-solutions for S under D if D has no target dependencies (a), and if D has target dependencies (b).

tgds in $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$ are full. Here, one first has to extend the definition of $\textsc{CanSol}_D(S)$ to such data exchange settings. The idea is to let $\textsc{CanSol}_D(S)$ be the result of chasing $\textsc{CanSol}_{(\sigma,\tau,\Sigma_{\mathrm{st}})}(S)$ with the dependencies in Σ_{t} (using the standard chase). Details can be found in Appendix A.

Definition 3.14 (*extended canonical solution*). Let $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ be a data exchange setting such that Σ_{t} consists of egds, or all tgds in $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$ are full. Let S be a source instance for D. If there is a CWA-solution for S under D, let $\textsc{CanSol}_D(S)$ be the unique CWA-solution T for S under D such that every CWA-solution T for S under D is a homomorphic image of $\textsc{CanSol}_D(S)$. Otherwise, let $\textsc{CanSol}_D(S)$ be undefined.

Combining Proposition 3.3, Proposition 3.13, and Theorem 3.11, we obtain the following description of CWA-solutions in the case of data exchange settings that adhere to the restriction of Definition 3.14:

THEOREM 3.15. *Let D be a data exchange setting such that Σ_t consists of egds, or all tgds in $\Sigma_{st} \cup \Sigma_t$ are full, and let S be a source instance for D. Then a target instance T for D is a CWA-solution for S under D if and only if the following three conditions are satisfied:*

(*1*) *T is a homomorphic image of* $\textsc{CanSol}_D(S)$*,*
(*2*) *there is a homomorphism from T to* $\textsc{CanSol}_D(S)$*, and*
(*3*) *T contains* $\textsc{Core}_D(S)$*.*

Thus, in this case, the space of CWA-solutions contains two unique extreme points (see Figure 1(a)): the core solution, which is the minimal CWA-solution in the sense that it is contained in all other CWA-solutions, and the canonical solution, which is the maximal CWA-solution in the sense that it has every CWA-solution as a homomorphic image.

Remark 3.16. While for every data exchange setting D without target dependencies, and for every source instance S for D, the minimal CWA-solution $\textsc{Core}_D(S)$ is contained in every CWA-solution for S under D, there can be CWA-solutions for S under D that are not contained in the maximal CWA-solution $\textsc{CanSol}_D(S)$. For example, if $D = (\{R\}, \{R'\}, \Sigma_{\mathrm{st}})$, where $\Sigma_{\mathrm{st}} = \{R(x,y) \rightarrow \exists z_1 \exists z_2\, R'(x, z_1, z_2)\}$, and $S = \{R(a,b), R(a,c)\}$, then $\textsc{CanSol}_D(S) = \{R'(a, \bot_1, \bot_2), R'(a, \bot_3, \bot_4)\}$, but $\{R'(a, \bot_1, \bot), R'(a, \bot_3, \bot)\}$ is a CWA-solution for S under D that is not contained in $\textsc{CanSol}_D(S)$. □

In general, however, there may be no CWA-solution that is maximal in the above sense. Based on the data exchange setting from Remark 3.16, it is not hard to construct a data exchange setting D and a source instance S for D where this is the case. In fact, one can construct D in such a way that for every positive integer n there is a source instance S for D, and 2^n distinct CWA-solutions $T_1, \dots, T_{2^n}$ for S under D such that for every CWA-solution T for S under D there is a homomorphism h from exactly one T_i to T with $h(T_i) = T$. Details can be found in Appendix A. Thus, in general, the set of all CWA-solutions is as shown in Figure 1(b).

4. THE COMPLEXITY OF COMPUTING CWA-SOLUTIONS

We now study the complexity of computing CWA-solutions. As is common in data exchange, we deal only with the case that the data exchange setting is *fixed*. That is, we are interested in the complexity of computing CWA-solutions, given a source instance for some data exchange setting D as input, where D is fixed and does not belong to the input. This corresponds to the *data complexity* [Vardi 1982] of computing CWA-solutions. For proving complexity lower bounds, we consider the corresponding decision problem:

Existence-of-CWA-Solutions(D)
Input: a source instance S for D
Question: Is there a CWA-solution for S under D?

We will also consider the analogous problems Existence-of-Solutions(D) and Existence-of-Universal-Solutions(D), which ask for the existence of a *solution*, or *universal solution*, respectively.

Section 4.1 presents some tractable cases, whereas Section 4.2 shows that the Existence-of-CWA-Solutions problem is undecidable in general.

4.1. Tractable Cases

In the case of data exchange settings $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, where Σ_{t} consists of egds only, there are polynomial time algorithms for computing minimal CWA-solutions as well as maximal CWA-solutions. First, it follows from results in [Fagin et al. 2005a; Arenas et al. 2004] that there is a polynomial time algorithm that, given a source instance S for D, computes the (extended) canonical solution for S under D: All that has to be done is to compute the canonical solution CanSol$_{(\sigma,\tau,\Sigma_{\mathrm{st}})}(S)$ under the data exchange setting $(\sigma, \tau, \Sigma_{\mathrm{st}})$, and to chase CanSol$_{(\sigma,\tau,\Sigma_{\mathrm{st}})}(S)$ with the egds in Σ_{t}. Furthermore, the blocks algorithm by Fagin et al. [2005b] is a polynomial time algorithm that, given a source instance S for D, computes the core solution for S under D provided it exists.

In the case that the set of target dependencies may contain target tgds, known tractability results for universal solutions carry over to CWA-solutions. A fairly broad class of data exchange settings for which universal solutions can be computed in polynomial time (data complexity) is the class of data exchange settings whose set of target dependencies is the union of a set of egds, and a *weakly acyclic set of tgds*:

Definition 4.1 (*[Fagin et al. 2005a; Deutsch and Tannen 2003]*). The *dependency graph* of a set Σ of tgds over τ is the following directed graph. The vertices are all pairs (R, i), called *positions*, where $R \in \tau$ and $i \in \{1, \dots, r\}$, with r being the arity of R. (A variable x is said to appear at position (R, i) in some conjunction φ of relational atomic formulas if φ contains a conjunct $R(t_1, \dots, t_r)$ with $t_i = x$.) For every tgd $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ in Σ, every variable x in $\bar{x}$, and every position (R, i) at which x appears in φ, there is

— a *copying* edge from (R, i) to every position at which x appears in ψ, and

— an *existential* edge from (R, i) to every position at which some variable from $\bar{z}$ appears in ψ.

Σ is called *weakly acyclic* if no cycle in the dependency graph of Σ contains an existential edge. A data exchange setting $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ is called *weakly acyclic* if Σ_{t} is the union of a set of egds, and a weakly acyclic set of tgds.

If $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ is weakly acyclic, there is a polynomial time algorithm based on the chase that, given a source instance S for D, computes a universal solution for S under D if there is one, and outputs that there is no such universal solution otherwise [Fagin et al. 2005a]. The algorithm computes an arbitrary complete chase sequence C of S with $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$; if C is successful and its result is $S \cup T$, it outputs T, otherwise no universal solution for S under D exists. Weak acyclicity here ensures that the length of C is polynomial in the size of S, and that S has no solution under D if C is failing [Fagin et al. 2005a]. Extensions of weak acyclicity which still guarantee these properties have been studied in [Deutsch et al. 2008; Lausen et al. 2009; Marnette 2009]. By Corollary 3.12, the above algorithm immediately yields an algorithm for Existence-of-CWA-Solutions(D). Notice that Proposition 3.1 in Kolaitis et al. [2006] implies a matching PTIME-completeness lower bound.

In some cases the universal solution computed by the algorithm even is a CWA-solution. For example, this is true if Σ_{t} contains no egds: If $S \cup T$ is the result of any successful chase sequence of S with $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$, it is easy to verify that T is a CWA-solution for S under D. On the other hand, it is not hard to find examples where Σ_{t} contains egds so that $S \cup T$ is the result of a successful chase sequence of S with $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$, but T is no CWA-solution; see Appendix B. Nevertheless, for every weakly acyclic data exchange setting, a CWA-solution, namely the core solution, can be computed in polynomial time:

Theorem 4.2 ([Gottlob and Nash 2008]). *For every weakly acyclic data exchange setting D, there is a polynomial time algorithm that, given a source instance S for D, decides whether* Core$_D(S)$ *exists, and if so, computes* Core$_D(S)$.

4.2. An Undecidable Case

We now show that, for a particular data exchange setting D, the problem Existence-of-CWA-Solutions(D) is undecidable.

Theorem 4.3. *There is a data exchange setting D_{HALT} such that* Existence-of-CWA-Solutions(D_{HALT}) *is undecidable.*

Proof. We first present the data exchange setting $D_{\mathrm{HALT}} = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, and then show the undecidability of Existence-of-CWA-Solutions(D_{HALT}) by a reduction from the following variant of the halting problem for Turing machines:

Halt

Input: a deterministic Turing machine $M = (Q, \Sigma, \delta, q_0, Q_F)$ with one tape which is infinite only to the right; here, Q is the set of states, Σ is the tape alphabet, $\delta\colon (Q \setminus Q_F) \times \Sigma \to Q \times \Sigma \times \{\mathrm{L}, \mathrm{R}\}$ is the transition function (a total function), $q_0 \in Q$ is the start state, and $Q_F \subseteq Q$ is the set of final states.

Question: Does M halt on the empty input?

The source schema σ of D_{HALT} contains a 5-ary relation symbol Δ to encode the graph of δ (i.e., $\Delta(q, s, q', s', d)$ corresponds to $\delta(q, s) = (q', s', d)$), and a unary relation symbol $\boldsymbol{Q}_0$ to encode the start state q_0.

Table I. Relation symbols in the target schema τ and their intended meaning

Symbol	Arity	Meaning
Δ'	5	copy of Δ
$Succ_t$	2	successor relation on time points (steps) of the computation
$Succ_p$	3	successor relation on tape positions, for each step
State	3	the state and the tape position for each step
Ins	3	the inscriptions of the relevant tape cells in each step
End	2	the position of the last relevant tape cell in each step
$Copy_L$	3	used to compute the inscriptions of tape cells to the *left* of a given tape position, based on the corresponding positions from the previous step
$Copy_R$	3	used to compute the inscriptions of tape cells to the *right* of a given tape position, based on the corresponding positions from the previous step

Table I below contains the relation symbols of the target schema τ and their intended meaning. For example, $\mathit{Succ}_p(t, p, p')$ means that tape position p' is the successor of tape position p in step t; $\mathit{State}(t, q, p)$ indicates that in step t, the machine is in state q and reads the tape cell at position p; $\mathit{Ins}(t, p, s)$ means that in step t, the tape cell at position p contains the symbol s; and $\mathit{End}(t, p)$ says that p is the last position in the linear order induced by the successor relation $\mathit{Succ}_p(t, \cdot, \cdot)$.

The set of s-t tgds Σ_{st} of D_{HALT} consists of a tgd

$$\Delta(q, s, q', s', d) \to \Delta'(q, s, q', s', d)$$

to copy Δ to Δ', and another tgd

$$\boldsymbol{Q}_0(q) \to \boldsymbol{State}(0, q, 1) \wedge \boldsymbol{Ins}(0, 1, \sqcap) \wedge \boldsymbol{Ins}(0, 2, \sqcap) \wedge \boldsymbol{Succ_p}(0, 1, 2) \wedge \boldsymbol{End}(0, 2)$$

to initialize the “start configuration” (here, $\sqcap$ is the blank symbol that is assumed to be in Σ): Thus, the latter tgd tells us that if q is the start state of M, then in step 0 of the computation of M on the empty input, M is in state q and reads the tape cell at position 1, which contains the blank symbol; the tape cell following the tape cell at position 1 is the tape cell at position 2, which, at the same time, is the last relevant tape cell and contains the blank symbol.

The set Σ_{t} of target dependencies of D_{HALT} consists of target tgds for simulating the Turing machine as follows. There are two tgds that simulate a transition – one for a transition where the tape head moves to the left, and one for a transition where the tape head moves to the right:

$$\begin{aligned}&\boldsymbol{State}(t, q, p) \wedge \boldsymbol{Ins}(t, p, s) \wedge \boldsymbol{Succ_p}(t, p', p) \wedge \Delta'(q, s, q', s', \mathrm{L}) \;\to \\ &\qquad \exists t'\big(\boldsymbol{Succ_t}(t, t') \wedge \boldsymbol{State}(t', q', p') \wedge \boldsymbol{Ins}(t', p, s') \wedge \boldsymbol{Copy_L}(t', t, p) \wedge \boldsymbol{Copy_R}(t', t, p)\big),\end{aligned} \tag{4.1}$$

$$\begin{aligned}&\boldsymbol{State}(t, q, p) \wedge \boldsymbol{Ins}(t, p, s) \wedge \boldsymbol{Succ_p}(t, p, p') \wedge \Delta'(q, s, q', s', \mathrm{R}) \;\to \\ &\qquad \exists t'\big(\boldsymbol{Succ_t}(t, t') \wedge \boldsymbol{State}(t', q', p') \wedge \boldsymbol{Ins}(t', p, s') \wedge \boldsymbol{Copy_L}(t', t, p) \wedge \boldsymbol{Copy_R}(t', t, p)\big).\end{aligned} \tag{4.2}$$

There are two tgds that copy the “successor relation” on the tape positions and the inscriptions of all unmodified tape cells:

$$\begin{aligned}&\boldsymbol{Copy_L}(t', t, p) \wedge \boldsymbol{Succ_p}(t, p', p) \wedge \boldsymbol{Ins}(t, p', s) \;\to \\ &\qquad\qquad \boldsymbol{Copy_L}(t', t, p') \wedge \boldsymbol{Succ_p}(t', p', p) \wedge \boldsymbol{Ins}(t', p', s),\end{aligned} \tag{4.3}$$

$$\begin{aligned}&\boldsymbol{Copy_R}(t', t, p) \wedge \boldsymbol{Succ_p}(t, p, p') \wedge \boldsymbol{Ins}(t, p', s) \;\to \\ &\qquad\qquad \boldsymbol{Copy_R}(t', t, p') \wedge \boldsymbol{Succ_p}(t', p, p') \wedge \boldsymbol{Ins}(t', p', s).\end{aligned} \tag{4.4}$$

Finally, there is a tgd that adds a new tape cell to the end of the tape and marks it the last relevant tape cell:

$$\boldsymbol{Succ}_t(t,t') \wedge \boldsymbol{End}(t,p) \;\rightarrow\; \exists p'\big(\boldsymbol{Succ}_p(t',p,p') \wedge \boldsymbol{Ins}(t',p',\sqcap) \wedge \boldsymbol{End}(t',p')\big). \tag{4.5}$$

This finishes the description of the data exchange setting D_{HALT}.

The reduction from HALT to EXISTENCE-OF-CWA-SOLUTIONS(D_{HALT}) is carried out as follows. Given a Turing machine $M = (Q, \Sigma, \delta, q_0, Q_F)$ for HALT, we create the source instance

$$S_M \;:=\; \{\Delta(q,s,q',s',d) \mid \delta(q,s) = (q',s',d)\} \;\cup\; \{\boldsymbol{Q}_0(q_0)\}.$$

It remains to show that M halts on the empty input if and only if there is a CWA-solution for S_M under D_{HALT}.

Let us first fix some basic notation on Turing machine computations. Recall that a computation of M on the empty input is a sequence $C_0, C_1, \dots, C_n$ of configurations of M, where C_0 is the start configuration of M on the empty input, and for $i < n$, C_{i+1} is the successor configuration of C_i. We represent each configuration C_i by a triple (q_i, p_i, x_i), where $q_i \in Q$ is the state, $p_i \geq 1$ is the head position, and $x_i = x_{i,1}x_{i,2}\cdots x_{i,l_i}$ is the inscription of the tape at positions 1 to l_i in step i of the computation. We can assume without loss of generality that $l_i = i+2$ (since M can visit at most i tape cells in i steps, all positions at positions greater than i are blanks; and we add 2 to simplify the presentation). In particular, $x_0 = \sqcap\sqcap$.

We are now ready to prove that M halts on the empty input if and only if there is a CWA-solution for S_M under D_{HALT}.

("Only if" direction) Suppose that there is a halting computation $C_0, C_1, \dots, C_n$ of M on the empty input, where $C_i = (q_i, p_i, x_{i,1}\cdots x_{i,i+2})$ for every $i \leq n$. Let $X_0, X_1, \dots, X_n$ be pairwise distinct values such that $X_0 = 0$, and $X_1, \dots, X_n$ are nulls. Moreover, let $Y_1, Y_2, \dots, Y_{n+2}$ be pairwise distinct values such that $Y_1 = 1$, $Y_2 = 2$, and $Y_3, \dots, Y_{n+2}$ are nulls that are distinct from $X_1, \dots, X_n$. Then it is easy to verify that the instance $S_M \cup T$ with

$$\begin{aligned} T \;=\; & \big\{\boldsymbol{Succ}_t(X_i, X_{i+1}) \mid 0 \leq i < n\big\} \\ & \cup \big\{\boldsymbol{State}(X_i, q_i, Y_{p_i}) \mid 0 \leq i \leq n\big\} \\ & \cup \big\{\boldsymbol{Ins}(X_i, Y_j, x_{i,j}) \mid 0 \leq i \leq n,\, 1 \leq j \leq i+2\big\} \\ & \cup \big\{\boldsymbol{Succ}_p(X_i, Y_j, Y_{j+1}) \mid 0 \leq i \leq n,\, 1 \leq j < i+2\big\} \\ & \cup \big\{\boldsymbol{End}(X_i, Y_{i+2}) \mid 0 \leq i \leq n\big\} \\ & \cup \big\{\boldsymbol{Copy}_L(X_i, X_{i-1}, j) \mid 1 \leq i \leq n,\, 1 \leq j \leq p_i\big\} \\ & \cup \big\{\boldsymbol{Copy}_R(X_i, X_{i-1}, j) \mid 1 \leq i \leq n,\, p_i \leq j < i+2\big\} \end{aligned}$$

is the result of a successful α-chase sequence of S_M with D_{HALT} for an injective mapping $\alpha\colon \mathcal{J}^*_{D_{\text{HALT}}} \to \mathsf{Null}$. Thus, T is a CWA-presolution for S_M under D_{HALT}. Since α is injective and $\alpha(\mathcal{J}^*_{D_{\text{HALT}}}) \subseteq \mathsf{Null}$, a straightforward modification of the proof of Theorem 3.3(1) in Fagin et al. [2005a] implies that T is a universal solution for S_M under D_{HALT}. Consequently, T is a CWA-solution for S_M under D_{HALT}.

("If" direction) Let T be a CWA-solution for S_M under D_{HALT}. Then there is a mapping $\alpha\colon \mathcal{J}^*_{D_{\text{HALT}}} \to \mathsf{Dom}$ such that $S_M \cup T$ is the result of a successful α-chase sequence of S_M with D_{HALT}. Recall from Proposition 3.6 that the results of any two successful α-chase sequences of S_M with D_{HALT} are the same.

We want to show that there is a halting computation of M on the empty input. The idea is to use α in order to "unravel" T to a sequence $T_0 \subset T_1 \subset \cdots \subset T_n = T$ of subinstances of T such that T_i encodes the first i steps of a halting computation of M on the empty input. Let T_0 be the canonical solution for S_M under the data exchange setting D_{HALT} without the target dependencies of D_{HALT}:

$$\begin{aligned} T_0 \ = \ & \{\Delta'(q,s,q',s',d) \mid \delta(q,s) = (q',s',d)\} \\ & \cup \{\boldsymbol{State}(0,q_0,1),\ \boldsymbol{Ins}(0,1,\sqcap),\ \boldsymbol{Ins}(0,2,\sqcap),\ \boldsymbol{Succ}_{\boldsymbol{p}}(0,1,2),\ \boldsymbol{End}(0,2)\}. \end{aligned}$$

For every $i \geq 0$ for which T_i does not satisfy (4.1) or (4.2), let T_{i+1} be the result of first α-applying (4.1) or (4.2) to T_i, then (4.3) and (4.4) until these are satisfied (which happens after a finite number of such applications), and finally, (4.5). If (4.1) and (4.2) are satisfied in T_i, let $T_{i+1} := T_i$. Since T is finite, $T_0 \subseteq T_1 \subseteq \cdots$, and $T_i = T_{i+1}$ implies $T_{i+1} = T_{i+2}$ for every $i \geq 0$, there is an integer n such that $T_n = T_{n'}$ for all $n' \geq n$.

We show by induction on i that for every $i \in \{0,1,\dots,n\}$ there are

(1) a computation $C_0,\dots,C_i$ of M on the empty input, where for every $j \leq i$, the configuration C_j has the form $(q_j, p_j, x_{j,1} \cdots x_{j,j+2})$,
(2) pairwise distinct $X_0, X_1, \dots, X_i \in \mathsf{Dom}$ with $X_0 = 0$, where $X_1,\dots,X_i$ are nulls, and
(3) pairwise distinct $Y_1, Y_2, \dots, Y_{i+2} \in \mathsf{Dom}$, where $Y_1 = 1$, $Y_2 = 2$, and $Y_3,\dots,Y_{i+2}$ are nulls distinct from $X_1,\dots,X_i$,[7]

such that

$$\begin{aligned} T_i \ = \ & \big\{\boldsymbol{Succ}_{\boldsymbol{t}}(X_j, X_{j+1}) \mid 0 \leq j < i\big\} \\ & \cup \big\{\boldsymbol{State}(X_j, q_j, Y_{p_j}) \mid 0 \leq j \leq i\big\} \\ & \cup \big\{\boldsymbol{Ins}(X_j, Y_k, x_{j,k}) \mid 0 \leq j \leq i,\ 1 \leq k \leq j+2\big\} \\ & \cup \big\{\boldsymbol{Succ}_{\boldsymbol{p}}(X_j, Y_k, Y_{k+1}) \mid 0 \leq j \leq i,\ 1 \leq k < j+2\big\} \\ & \cup \big\{\boldsymbol{End}(X_j, Y_{j+2}) \mid 0 \leq j \leq i\big\} \\ & \cup \big\{\boldsymbol{Copy}_{\boldsymbol{L}}(X_j, X_{j-1}, k) \mid 1 \leq j \leq i,\ 1 \leq k \leq p_j\big\} \\ & \cup \big\{\boldsymbol{Copy}_{\boldsymbol{R}}(X_j, X_{j-1}, k) \mid 1 \leq j \leq i,\ p_j \leq k < j+2\big\}. \end{aligned} \tag{4.6}$$

For $i = 0$, we let C_0 be the start configuration $(q_0, 1, \sqcap\sqcap)$ of M on the empty input, $X_0 := 0$, $Y_1 := 1$, and $Y_2 := 2$. Then 1–3 are satisfied, and T_0 has the required form (4.6).

Suppose now that for some $i < n$, there are

- a computation $C_0,\dots,C_i$ of M on the empty input, where each C_j has the form $(q_j, p_j, x_{j,1} \cdots x_{j,j+2})$,
- pairwise distinct $X_0, X_1, \dots, X_i \in \mathsf{Dom}$ with $X_0 = 0$, where $X_1,\dots,X_i$ are nulls, and
- pairwise distinct $Y_1, Y_2, \dots, Y_{i+2} \in \mathsf{Dom}$, where $Y_1 = 1$, $Y_2 = 2$ and $Y_3,\dots,Y_{i+2}$ are nulls,

such that T_i has the form (4.6). Recall that the first step in obtaining T_{i+1} from T_i is an α-application of (4.1) or (4.2) to T_i. In the following, we consider the case that (4.1) is α-applied; the other case is analogous.

Using that T_i has the form (4.6), it is easy to verify that after α-applying (4.1) to T_i, we obtain the instance

$$\begin{aligned} T'_{i+1} \ := \ T_i \cup \big\{ & \boldsymbol{Succ}_{\boldsymbol{t}}(X_i, X_{i+1}), \boldsymbol{State}(X_{i+1}, q_{i+1}, Y_{p_{i+1}}), \boldsymbol{Ins}(X_{i+1}, Y_{p_i}, x_{i+1,p_i}), \\ & \boldsymbol{Copy}_{\boldsymbol{L}}(X_{i+1}, X_i, p_i), \boldsymbol{Copy}_{\boldsymbol{R}}(X_{i+1}, X_i, p_i)\big\}, \end{aligned} \tag{4.7}$$

[7] For the proof we actually do not need that $X_1,\dots,X_i$ are nulls, and that $Y_3,\dots,Y_{i+2}$ are nulls distinct from $X_1,\dots,X_i$. However, these requirements will be useful later in Remark 4.6.

where $X_{i+1} \in \mathsf{Dom}$, $\delta(q_i, x_{i,p_i}) = (q_{i+1}, x_{i+1,p_i}, \mathrm{L})$, and $p_{i+1} := p_i - 1$.

Let $C_{i+1} := (q_{i+1}, p_{i+1}, x_{i+1,1} \cdots x_{i+1,(i+1)+2})$, where we have $x_{i+1,j} := x_{i,j}$ for all $j \in \{1, \dots, i+2\} \setminus \{p_i\}$, $x_{i+1,(i+1)+2} := \sqcap$, and $x_{i+1,p}$ as determined above by $\delta(q_i, x_{i,p_i})$. Then C_{i+1} is a successor configuration of C_i, and consequently, $C_0, C_1, \dots, C_{i+1}$ is a computation of M on the empty input. This shows that $C_0, C_1, \dots, C_{i+1}$ is as required by 1. Moreover, the following claim shows that $X_0, X_1, \dots, X_{i+1}$ is as required by 2.

Claim (*). The values $X_0, X_1, \dots, X_{i+1}$ are pairwise distinct, with $X_0 = 0$, and $X_1, \dots, X_{i+1}$ being nulls.

Proof. By the induction hypothesis, we already know that $X_0, X_1, \dots, X_i$ are pairwise distinct, $X_0 = 0$, and $X_1, \dots, X_i$ are nulls. Therefore, it remains to prove that X_{i+1} is a null that does not occur in $\{X_0, X_1, \dots, X_i\}$.

Suppose, to the contrary, that $X_{i+1} = X_j$ for some $j \leq i$, or that X_{i+1} is not a null. Pick $\hat{X}_{i+1} \in \mathsf{Null} \setminus \mathrm{dom}(T_i)$. Then it is easy to verify that

$$\begin{aligned} T^* \; := \; & T_i \cup \big\{ \boldsymbol{Succ}_t(X_i, \hat{X}_{i+1}), \boldsymbol{Succ}_t(\hat{X}_{i+1}, \hat{X}_{i+1}) \big\} \\ & \cup \big\{ R(\hat{X}_{i+1}, \bar{u}) \mid R \in \tau \setminus \{\Delta', \boldsymbol{Succ}_t\} \text{ and } \bar{u} \in \big(\mathrm{dom}(T_i) \cup \{\hat{X}_{i+1}\}\big)^{r-1}, \\ & \qquad \text{where } r \text{ is the arity of } R \big\} \end{aligned}$$

is a solution for S_M under D_{HALT}. Since T is a CWA-solution and $T'_{i+1} \subseteq T$, there is a homomorphism h from T'_{i+1} to T^*. Observe that $h(X_0) = X_0$, because $X_0 = 0$ is a constant and h is the identity on constants. Together with $\boldsymbol{Succ}_t^{T_i} = \{(X_j, X_{j+1}) \mid 0 \leq j < i\}$ and the fact that $X_0, X_1, \dots, X_i$ are pairwise distinct, an induction on j shows that $h(X_j) = X_j$ for every $j \leq i$. Moreover, since $\boldsymbol{Succ}_t(X_i, X_{i+1}) \in T'_{i+1}$, and $\boldsymbol{Succ}_t(X_i, \hat{X}_{i+1})$ is the only atom of the form $\boldsymbol{Succ}_t(X_i, \cdot)$ in T^*, we have $h(X_{i+1}) = \hat{X}_{i+1}$.

If $X_{i+1} = X_j$ for some $j \leq i$, this leads to a contradiction: $h(X_{i+1}) = X_j \neq \hat{X}_{i+1} = h(X_{i+1})$. Hence, by the assumption on X_{i+1}, we know that X_{i+1} is not a null, i.e., X_{i+1} is a constant. However, since h is the identity on constants, it is impossible that $h(X_{i+1}) = \hat{X}_{i+1}$ – again, a contradiction. So, the assumption that $X_{i+1} = X_j$ for some $j \leq i$, or that X_{i+1} is not a null must be false. □

Note that the instance T_{i+1} is obtained from T'_{i+1} by α-applying (4.3) and (4.4) until these are satisfied, and finally (4.5). From this, together with the fact that $X_0, X_1, \dots, X_{i+1}$ are pairwise distinct, we conclude that T_{i+1} has the form (4.6), where $Y_{(i+1)+2} \in \mathsf{Dom}$. What remains is to prove that $Y_1, Y_2, \dots, Y_{(i+1)+2}$ are pairwise distinct, and that $Y_3, \dots, Y_{(i+1)+2}$ are nulls distinct from $X_1, \dots, X_{i+1}$. This can be proven in a similar way as Claim (*).

Finally, we show that $C_0, C_1, \dots, C_n$ is a halting computation of M on the empty input. Since $C_0, C_1, \dots, C_n$ is a computation of M on the empty input, we need to show that q_n is a final state. Let β be an assignment for the variables in the body of (4.1) resp. (4.2) with $\beta(t) = X_n$, $\beta(q) = q_n$, and $\beta(s) = x_{n,p_n}$. If β satisfies the body of (4.1), then there are $X_{n+1} \in \mathsf{Dom}$, a state q_{n+1}, and a symbol x_{n+1,p_n} with $\delta(q_n, x_{n,p_n}) = (q_{n+1}, x_{n+1,p_n}, \mathrm{L})$ such that for $p_{n+1} := p_n - 1$, the result of α-applying (4.1) to T_n has the form (4.7) with i replaced by n. By a proof similar to the proof of Claim (*), we obtain $X_{n+1} \notin \{X_0, X_1, \dots, X_n\}$. But this is impossible, because this would imply that $T_{n+1} \supsetneq T_n$; however, n has been chosen such that $T_{n+1} = T_n$. Therefore, β cannot satisfy the body of (4.1). In other words, $\delta(q_n, x_{n,p_n})$ must be undefined. Since δ is a total function on $(Q \setminus Q_F) \times \Sigma$, we conclude that q_n is a final state. The case that β satisfies the body of (4.2) can be handled similarly.

Altogether, the proof of Theorem 4.3 is complete. □

Note that Theorem 4.3 and Corollary 3.12 immediately lead to:

COROLLARY 4.4. *There is a data exchange setting D such that* EXISTENCE-OF-UNIVERSAL-SOLUTIONS(D) *is undecidable.*

Remark 4.5. Even if we would allow infinite CWA-solutions, the EXISTENCE-OF-CWA-SOLUTIONS problem – now asking for infinite CWA-solutions – would be undecidable in general. To see this, let us extend D_{HALT} to a data exchange setting D'_{HALT} by adding unary relation symbols F and F' to the source schema and the target schema, respectively, the s-t tgd $F(q) \rightarrow F'(q)$, and the egd $\mathit{State}(t,q,p) \wedge F'(q) \wedge \Delta'(q',s',q'',s'',d) \rightarrow q = q'$. Furthermore, let $M = (Q,\Sigma,\delta,q_0,Q_F)$ be a Turing machine with $Q \setminus Q_F \neq \emptyset$ that is an admissible input for HALT. Then, for $S'_M := S_M \cup \{F(q) \mid q \in Q_F\}$, it is clear that there is an (infinite) CWA-solution for S'_M whenever M does *not* reach a final state on the empty input, and the latter is the case exactly if M does not halt on the empty input. □

Remark 4.6. As a corollary of the proof of Theorem 4.3 we obtain Theorem 1, Theorem 6 and Theorem 14 of Deutsch et al. [2008]. These results are based on the following terminology. Let Σ be a set of tgds and egds, and let I be an instance. A *model for Σ and I* is a possibly infinite instance J such that there is a homomorphism from I to J, and $J \models \Sigma$. Note that the first condition boils down to $I \subseteq J$ if J contains only constants. A *strong universal model* for Σ and I is a *finite* model for Σ and I such that for every model K for Σ and I there is a homomorphism from J to K. A *weak universal model* for Σ and I is a *finite* model for Σ and I such that for every *finite* model K for Σ and I there is a homomorphism from J to K.

THEOREM 4.7 (DEUTSCH ET AL. [2008]). *It is undecidable, given an instance I and a set Σ of tgds and egds,*

(*1*) *whether there is some complete chase sequence of I with Σ,*
(*2*) *whether all chase sequences of I with Σ terminate (i.e., can be extended to a complete chase sequence of I with Σ),*
(*3*) *whether a strong universal model for Σ and I exists, and*
(*4*) *whether a weak universal model for Σ and I exists.*

This is even true over a fixed schema σ and for $I = \emptyset$.

The first two statements of Theorem 4.7 can be obtained from Theorem 4.3 as follows: Let $D_{\text{HALT}} = (\sigma,\tau,\Sigma_{\text{st}},\Sigma_{\text{t}})$. Given a deterministic Turing machine M, let Σ'_{st} be the set of all s-t tgds of the form $\rightarrow \varphi$ (with empty body), where φ is the conjunction of all atoms in the canonical solution for S_M under $(\sigma,\tau,\Sigma_{\text{st}})$. Then, every chase sequence of $\emptyset$ with $\Sigma'_{\text{st}} \cup \Sigma_{\text{t}}$ is an α-chase sequence of $\emptyset$ with $D'_{\text{HALT}} := (\sigma,\tau,\Sigma'_{\text{st}},\Sigma_{\text{t}})$, where α is an injective function from $\mathcal{J}^*_{D'_{\text{HALT}}}$ to Null. Moreover, every (successful) α-chase sequence of $\emptyset$ with D'_{HALT} can be turned into a (successful) α-chase sequence of S_M with D_{HALT}, and vice versa. Finally, the proof of Theorem 4.3 shows that there is a successful α-chase sequence of S_M with D_{HALT}, where α is an injective function from $\mathcal{J}^*_{D_{\text{HALT}}}$ to Null, if and only if there is a CWA-solution for S under D_{HALT}. Together with Theorem 4.3 and Proposition 3.6(1), this immediately proves the first two statements of Theorem 4.7.

To prove the remaining two statements of Theorem 4.7, note that there is a universal solution for S_M under D_{HALT} if and only if there is a weak universal model for $\Sigma'_{\text{st}} \cup \Sigma_{\text{t}}$ and $\emptyset$, and that every weak universal model for $\Sigma'_{\text{st}} \cup \Sigma_{\text{t}}$ and $\emptyset$ is strongly universal. □

Interestingly, the analogous EXISTENCE-OF-SOLUTIONS problem, asking for a solution instead of a CWA-solution, is trivial for the schema mapping D_{HALT} constructed in the proof of Theorem 4.3. In fact, for every source instance S for $D_{\text{HALT}} =$

$(\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$, the target instance that interprets every relation symbol $R \in \tau$ by $\big(\mathrm{const}(S) \cup \{0,1,2,\sqcap\}\big)^r$, with r being the arity of R, is a solution for S under D_{HALT}.

On the other hand, Kolaitis et al. [2006] have established an undecidability result for the EXISTENCE-OF-SOLUTIONS problem. They exhibit a data exchange setting D_{emb}, and reduce the

EMBEDDING PROBLEM FOR FINITE SEMIGROUPS (EMB)
Input: a finite set A, and a partial function $p\colon A \times A \to A$
Question: Is there a finite set $B \supseteq A$ and a total function $f\colon B \times B \to B$ such that f is associative, and f extends p (i.e., whenever $p(a,b)$ is defined, then $f(a,b) = p(a,b)$)?

which is known to be undecidable [Kolaitis et al. 2006], to EXISTENCE-OF-SOLUTIONS(D_{emb}). D_{emb} has the source schema $\{R\}$ and the target schema $\{R'\}$, where R and R' are ternary relation symbols. The intention is that R encodes the graph of the input function $p\colon A \times A \to A$, whereas R' encodes the graph of a solution $f\colon B \times B \to B$ with respect to EMB. There is one s-t tgd $R(x,y,z) \to R'(x,y,z)$ to ensure that f is an extension of p. Furthermore, there are the target dependencies

$$R'(x,y,z_1) \land R'(x,y,z_2) \;\to\; z_1 = z_2, \tag{4.8}$$

$$R'(x,y,u) \land R'(y,z,v) \land R'(u,z,w) \;\to\; R'(x,v,w), \tag{4.9}$$

$$R'(x_1,x_2,x_3) \land R'(y_1,y_2,y_3) \;\to\; \exists z\, R'(x_i,y_j,z) \text{ for } 1 \le i,j \le 3, \tag{4.10}$$

where (4.8) ensures that f is a function, (4.9) ensures that f is associative, and (4.10) ensures that f is total. The reduction is carried out by encoding an input function $p\colon A \times A \to A$ by the source instance $S_p := \{R(x,y,z) \mid p(x,y) = z\}$, which has a solution under D_{emb} if and only if the desired set B and function f exists. However, the following example shows that this reduction does not establish that EXISTENCE-OF-CWA-SOLUTIONS(D_{emb}) is undecidable.

Example 4.8. The partial function $p\colon \{0,1\}^2 \to \{0,1\}$ with $p(0,1) = 1$ and undefined otherwise is clearly a "yes"-instance for EMB, since $f\colon \{0,1\}^2 \to \{0,1\}$ with $f(x,y) := x + y \bmod 2$ extends p, is associative, and total. However, we show that $S_p = \{R(0,1,1)\}$ is a "no"-instance for EXISTENCE-OF-CWA-SOLUTIONS(D_{emb}), that is, there is no CWA-solution for S_p under D_{emb}.

Assume, to the contrary, that there is a CWA-solution T for S_p under D_{emb}. Since T is finite and satisfies (4.10), there are an integer $k \ge 0$, pairwise distinct values $u_0, u_1, \dots, u_k$ with $u_0 = 0$, and an integer $i \le k$ such that

$$\{R'(u_0,1,u_1),\ R'(u_1,1,u_2),\ \dots,\ R'(u_{k-1},1,u_k),\ R'(u_k,1,u_i)\} \;\subseteq\; T.$$

On the other hand, the target instance $T' := \{R'(u,v,w) \mid u + v = w \bmod k+2\}$ is a solution for S_p under D_{emb}, and Theorem 3.9 implies that there is a homomorphism h from T to T'. In particular,

$$\{R'\big(h(u_0),1,h(u_1)\big),\ R'\big(h(u_1),1,h(u_2)\big),\ \dots,\ R'\big(h(u_k),1,h(u_i)\big)\} \;\subseteq\; T'.$$

Note that we must have $h(u_0) = 0$, since $u_0 = 0$ is a constant, and homomorphisms are the identity on constants. Furthermore, we must have $h(u_i) = i$ for all $i \in \{1,\dots,k\}$: if $i < k$ and $h(u_i) = i$, then $h(u_{i+1}) = i+1$, since T contains the atom $R'(u_i,1,u_{i+1})$, and $R'(i,1,i+1)$ is the only atom of the form $R'(i,1,\cdot)$ in T'. Hence, T' must contain the atom $R'(k,1,i)$, where $i \le k$. But $R'(k,1,k+1)$ is the only atom in T' of the form $R'(k,1,\cdot)$ – a contradiction.

Clearly, there are also "yes"-instances p for EMB such that S_p is a "yes"-instance for EXISTENCE-OF-CWA-SOLUTIONS(D_{emb}). For example, this is true if p is empty or a total associative function. Therefore, even “flipping” the answers, i.e., answering “yes” if S_p is a “no”-instance for EXISTENCE-OF-CWA-SOLUTIONS(D_{emb}), and “no” otherwise, does not yield a reduction from EMB to EXISTENCE-OF-CWA-SOLUTIONS(D_{emb}). □

Even more, we can extend D_{emb} to a data exchange setting D'_{emb} such that EXISTENCE-OF-SOLUTIONS(D'_{emb}) is undecidable, but EXISTENCE-OF-CWA-SOLUTIONS(D'_{emb}) is trivial, which demonstrates once more the difference between EXISTENCE-OF-SOLUTIONS on the one hand and EXISTENCE-OF-CWA-SOLUTIONS as well as EXISTENCE-OF-UNIVERSAL-SOLUTIONS on the other hand.

Example 4.9. Construct D'_{emb} from D_{emb} by adding a new binary target relation symbol E, the s-t tgd $\rightarrow E(0,1)$ without body (which ensures that every solution contains the atom $E(0,1)$), and the target tgd $d := E(x,y) \rightarrow \exists z\, E(y,z)$. Then for every source instance S for D'_{emb}, we have:

— There is a solution for S under D'_{emb} if and only if there is a solution for S under D_{emb}. This is because d is independent of the other tgds and egds of D_{emb}, and d is satisfied in every solution for S under D_{emb} containing $E(0,1)$.
— There is no CWA-solution for S under D'_{emb}. This is enforced by $E(0,1)$ and d.

Hence, the problem EXISTENCE-OF-SOLUTIONS(D'_{emb}) is undecidable, while the problem EXISTENCE-OF-CWA-SOLUTIONS(D'_{emb}) is trivial. □

5. QUERY ANSWERING SEMANTICS

In this section, we introduce the new CWA-solution-based query answering semantics, and argue that some of the anomalies mentioned in Section 1 do not arise for the new semantics. In Section 6, we then study the complexity of query answering under the new semantics.

The new semantics are basically the *certain answers* and the *maybe answers* on CWA-solutions. That is, we take either the intersection (for the certain answers) or the union (for the maybe answers) of the answers to a query $Q(\bar{x})$ on individual CWA-solutions T. Recall, however, that CWA-solutions are in general instances with incomplete information (cf., Section 2.5). Therefore, rather than answering $Q(\bar{x})$ on T using the naive semantics $Q(T)$, we employ techniques for answering queries on instances with incomplete instances and return either the certain answers of Q on T, or the maybe answers of Q on T. Of course, we have to take into account the set Σ_{t} of target dependencies of the corresponding data exchange setting, so that we take the certain answers and the maybe answers with respect to Σ_{t}.

To be more precise, let $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ be a data exchange setting, let S be a source instance for D, and let $Q(\bar{x})$ be a query over τ. Then, using the notions $\Box_{\Sigma_{\mathrm{t}}} Q(T)$ and $\Diamond_{\Sigma_{\mathrm{t}}} Q(T)$ introduced in Section 2.5, we define the following semantics for answering $Q(\bar{x})$:

— The *certain answers semantics on CWA-solutions.* The certain answers of Q on CWA-solutions for S under D, denoted by $\mathsf{certain}^D_\Box(Q,S)$, contains all $|\bar{x}|$-tuples $\bar{t}$ such that for all CWA-solutions T for S under D we have $\bar{t} \in \Box_{\Sigma_{\mathrm{t}}} Q(T)$:

$$\mathsf{certain}^D_\Box(Q,S) \;=\; \bigcap_{T \in [\![S]\!]^D_{\mathrm{CWA}}} \Box_{\Sigma_{\mathrm{t}}} Q(T).$$

— The *potential certain answers semantics on CWA-solutions.* The potential certain answers of Q on CWA-solutions for S under D, denoted by $\mathsf{certain}^D_\Diamond(Q,S)$, contains all $|\bar{x}|$-tuples $\bar{t}$ such that there is a CWA-solution T for S under D with $\bar{t} \in \Box_{\Sigma_t} Q(T)$:

$$\mathsf{certain}^D_\Diamond(Q,S) \;=\; \bigcup_{T \in [\![S]\!]^D_{\mathrm{CWA}}} \Box_{\Sigma_t} Q(T).$$

— The *persistent maybe answers semantics on CWA-solutions*. The persistent maybe answers of Q on CWA-solutions for S under D, denoted by $\mathsf{maybe}^D_\Box(Q,S)$, contains all $|\bar{x}|$-tuples $\bar{t}$ such that for all CWA-solutions T for S under D we have $\bar{t} \in \Diamond_{\Sigma_t} Q(T)$:

$$\mathsf{maybe}^D_\Box(Q,S) \;=\; \bigcap_{T \in [\![S]\!]^D_{\mathrm{CWA}}} \Diamond_{\Sigma_t} Q(T).$$

— The *maybe answers semantics on CWA-solutions*. The maybe answers of Q on CWA-solutions for S under D, denoted by $\mathsf{maybe}^D_\Diamond(Q,S)$, contains all $|\bar{x}|$-tuples $\bar{t}$ such that there is a CWA-solution T for S under D with $\bar{t} \in \Diamond_{\Sigma_t} Q(T)$:

$$\mathsf{maybe}^D_\Diamond(Q,S) \;=\; \bigcup_{T \in [\![S]\!]^D_{\mathrm{CWA}}} \Diamond_{\Sigma_t} Q(T).$$

Note that the above four semantics simply arise from the certain answers semantics and the maybe answers semantics over CWA-solutions by consequential use of proper query evaluation semantics for instances with incomplete information (as indicated in Section 1.4, here we concentrate on the two basic semantics, the certain answers semantics and the maybe answers semantics).

Example 5.1. Recall Example 1.1. Suppose we want to answer the query

$$\begin{aligned} Q(i) \;:=\; & \exists t\, \exists k\, \mathit{Paper}(i,t,k) \\ & \wedge \neg \exists t\, \exists k\, \exists i' \big(\mathit{Paper}(i,t,k) \wedge \mathit{Assign}(i,i') \wedge \mathit{PC}'(i',n_1)\big) \end{aligned}$$

which asks for all IDs i of submissions *not* assigned to PC member n_1. The certain answers of Q on S with respect to D_{conf} are empty, because, as is easy to see, there is a solution for S under D_{conf} in which n_1 has assigned all available submissions. However, looking at D_{conf} and S, it seems natural to expect the answer to contain the entries 2 and 3, since n_1 is not assigned to the submissions with IDs 2 and 3 according to the source instance S, and D_{conf} intuitively does not assign n_1 to those submissions. And indeed, it is not hard to see that under each of the above semantics, the answer to Q on S with respect to D_{conf} is $\{2,3\}$. □

Let us now return to the anomalies mentioned in Section 1. Each of the above semantics leads to the desired result on copying data exchange settings:

Example 5.2. Consider a copying data exchange setting D (cf., Section 1.2) and a source instance S for D. Then the "copy" S' of S over τ (i.e., $(R')^{S'} = R^S$ for all $R \in \sigma$) is the unique CWA-solution for S under D, and therefore,

$$\mathsf{certain}^D_\Box(Q,S) = \mathsf{certain}^D_\Diamond(Q,S) = \mathsf{maybe}^D_\Box(Q,S) = \mathsf{maybe}^D_\Diamond(Q,S) = Q(S'),$$

as intuitively expected. More generally, let D be a data exchange setting defined by full tgds and egds. Then there is at most one CWA-solution for S under D (note that if there is no CWA-solution, then there is no solution at all). If a CWA-solution exists, then this CWA-solution, call it T, intuitively corresponds to the expected result of translating S to the target, so that the answer to a query Q on S with respect to D should be expected to be $Q(T)$; indeed, in this case each of the new semantics yields the answer $Q(T)$. Otherwise, if no CWA-solution exists, then each of the new semantics yields the empty set, as expected. □

The preceeding example demonstrates that the anomalies observed by Arenas et al. [2004] and explained in Section 1.2 disappear with the CWA-solution-based semantics. In addition, it shows that the rewriting of a query Q in a copying data exchange setting is Q itself, as it should be in such a setting. Thus, the CWA-solution-based semantics resolve some of the most unpleasant anomalies of query answering in data exchange.

5.1. Characterizations

While the new semantics seem to be rather diverse, there are simple connections between them, and in some cases they can be characterized in terms of canonical solutions, and core solutions. Theorem 5.3 below states that in order to evaluate a query under the potential certain answers semantics or the persistent maybe answers semantics, it suffices to compute the certain answers or the maybe answers, respectively, on the core solution. A similar characterization for the certain answers semantics and the maybe answers semantics, and with core solution replaced by canonical solution, holds for data exchange settings without target dependencies (and slight extensions thereof), but not in general. The problem of answering queries in data exchange can thus often be reduced to the classical and well studied problem of answering queries in databases with incomplete information [Abiteboul et al. 1995; Abiteboul et al. 1991; Imielinski and Lipski, Jr. 1984].

THEOREM 5.3. *Let $D = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be a data exchange setting, let S be a source instance for D, and let Q be a query over τ. Then we have*

(*1*) $\mathsf{certain}^D_\Diamond(Q, S) = \Box_{\Sigma_t} Q(\textsc{Core}_D(S))$,
(*2*) $\mathsf{maybe}^D_\Box(Q, S) = \Diamond_{\Sigma_t} Q(\textsc{Core}_D(S))$.

Moreover, if Σ_t consists of egds, or all tgds in $\Sigma_{st} \cup \Sigma_t$ are full, then

(*3*) $\mathsf{certain}^D_\Box(Q, S) = \Box_{\Sigma_t} Q(T^*)$,
(*4*) $\mathsf{maybe}^D_\Diamond(Q, S) = \Diamond_{\Sigma_t} Q(T^*)$.

For the proof, we need the following result:

PROPOSITION 5.4. *Let $D = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be a data exchange setting, and let S be a source instance for D. Then, for every CWA-solution T for S under D, we have $\mathit{Rep}_{\Sigma_t}(\textsc{Core}_D(S)) \subseteq \mathit{Rep}_{\Sigma_t}(T)$.*

Moreover, if Σ_t consists of egds, or all tgds in $\Sigma_{st} \cup \Sigma_t$ are full, then $\mathit{Rep}_{\Sigma_t}(T) \subseteq \mathit{Rep}_{\Sigma_t}(\textsc{CanSol}_D(S))$.

PROOF. Let T be a CWA-solution for S under D. To prove $\mathit{Rep}_{\Sigma_t}(\textsc{Core}_D(S)) \subseteq \mathit{Rep}_{\Sigma_t}(T)$, let $\hat{T} \in \mathit{Rep}_{\Sigma_t}(\textsc{Core}_D(S))$. Then there is a valuation v of $\textsc{Core}_D(S)$ with $v(\textsc{Core}_D(S)) = \hat{T}$. On the other hand, T is a universal solution for S under D, and by Theorem 2.1(3), there is a homomorphism h from T to $\textsc{Core}_D(S)$ such that $h(T) = \textsc{Core}_D(S)$. It follows that the composition $v' := v \circ h$ of h and v is a valuation of T with $v'(T) = v(\textsc{Core}_D(S)) = \hat{T}$, and consequently, $\hat{T} \in \mathit{Rep}_{\Sigma_t}(T)$

The proof for the other inclusion $\mathit{Rep}_{\Sigma_t}(T) \subseteq \mathit{Rep}_{\Sigma_t}(\textsc{CanSol}_D(S))$ in the case that Σ_t consists of egds, or all tgds in $\Sigma_{st} \cup \Sigma_t$ are full, is analogous: in this case, the (extended) canonical solution $\textsc{CanSol}_D(S)$ exists, and there is a homomorphism h from $\textsc{CanSol}_D(S)$ to T with $h(\textsc{CanSol}_D(S)) = T$. □

We are now ready to give the proof of Theorem 5.3:

PROOF OF THEOREM 5.3. We first prove 1 and 2. By Proposition 5.4, every CWA-solution T for S under D satisfies $\mathit{Rep}_{\Sigma_t}(\textsc{Core}_D(S)) \subseteq \mathit{Rep}_{\Sigma_t}(T)$, hence

$$\Box_{\Sigma_t} Q(T) = \bigcap_{\hat{T} \in \mathit{Rep}_{\Sigma_t}(T)} Q(\hat{T}) \subseteq \bigcap_{\hat{T} \in \mathit{Rep}_{\Sigma_t}(\textsc{Core}_D(S))} Q(\hat{T}) = \Box_{\Sigma_t} Q(\textsc{Core}_D(S)), \tag{5.1}$$

$$\Diamond_{\Sigma_t} Q(T) = \bigcup_{\hat{T} \in \mathit{Rep}_{\Sigma_t}(T)} Q(\hat{T}) \supseteq \bigcup_{\hat{T} \in \mathit{Rep}_{\Sigma_t}(\textsc{Core}_D(S))} Q(\hat{T}) = \Diamond_{\Sigma_t} Q(\textsc{Core}_D(S)). \tag{5.2}$$

Consequently,

$$\begin{aligned} \mathsf{certain}^D_{\Diamond}(Q,S) &= \bigcup_{T \in [\![S]\!]^D_{\mathrm{CWA}}} \Box_{\Sigma_t} Q(T) \overset{(5.1)}{=} \Box_{\Sigma_t} Q(\textsc{Core}_D(S)), \\ \mathsf{maybe}^D_{\Box}(Q,S) &= \bigcap_{T \in [\![S]\!]^D_{\mathrm{CWA}}} \Diamond_{\Sigma_t} Q(T) \overset{(5.2)}{=} \Diamond_{\Sigma_t} Q(\textsc{Core}_D(S)). \end{aligned}$$

The proof for 3 and 4 is analogous; here we use $\mathit{Rep}_{\Sigma_t}(T) \subseteq \mathit{Rep}_{\Sigma_t}(\textsc{CanSol}_D(S))$ as guaranteed by Proposition 5.4. □

Furthermore, one can use Theorem 5.3 to establish the following relationship between the new semantics.

COROLLARY 5.5. *For every data exchange setting $D = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$, every source instance S for D, and every query $Q(\bar{x})$ over τ, we have:*

$$\mathsf{certain}^D_{\Box}(Q,S) \subseteq \mathsf{certain}^D_{\Diamond}(Q,S) \subseteq \mathsf{maybe}^D_{\Box}(Q,S) \subseteq \mathsf{maybe}^D_{\Diamond}(Q,S).$$

PROOF. The first and third inclusion follow directly from the definitions. For proving $\mathsf{certain}^D_{\Diamond}(Q,S) \subseteq \mathsf{maybe}^D_{\Box}(Q,S)$, observe that for all CWA-solutions T for S under D it holds that $\Box_{\Sigma_t} Q(T) \subseteq \Diamond_{\Sigma_t} Q(T)$. Therefore it follows from Theorem 5.3 that $\mathsf{certain}^D_{\Diamond}(Q,S) = \Box_{\Sigma_t} Q(\textsc{Core}(S)) \subseteq \Diamond_{\Sigma_t} Q(\textsc{Core}(S)) = \mathsf{maybe}^D_{\Box}(Q,S)$. □

Note that the certain answers semantics and the certain answers semantics on universal solutions of [Fagin et al. 2005a; Fagin et al. 2005b] both produce subsets of $\mathsf{certain}^D_{\Box}(Q,S)$.

5.2. Discussion and Other Semantics

Although the CWA-solution-based semantics resolve the anomalies of query answering mentioned in Section 1.2, there are scenarios where they still give rise to unintended answers.

Example 5.6. Consider a data exchange setting with the source schema containing a relation *Person* with an attribute *name*, the target schema containing a relation *ParentChild* with attributes *parent* and *child*, and a single tgd

$$\forall x\ (\mathit{Person}(x) \to \exists z\ \mathit{ParentChild}(x,z)).$$

Then, under the CWA, for each person p, a single tuple $(p, \bot)$ will be inserted into the target. Hence, the certain answers to a query $\forall x \exists! z\ \mathit{ParentChild}(x,z)$ stating that each parent has exactly one child will be true, even though this was not intended. □

To remedy this shortcoming, several other semantics for query answering have been proposed [Libkin and Sirangelo 2011; Afrati and Kolaitis 2008; Hernich 2010], which we now briefly discuss.

Motivated by problems such as those described in Example 5.6, Libkin and Sirangelo [2011] propose a combination of the CWA-based approach of this paper with the "classical" OWA-based approach. Rather than insisting that the values at all positions of atoms in the target are unique (since these atoms are justified by exactly one justification), they relax this requirement by letting the user (or more precisely, the developer of the data exchange setting) control which positions of atoms in the target may be considered as *open* (not unique), and which positions are considered as *closed* (unique). If a position of an atom is *open*, then we may instantiate that atom with a different value at that position (so the value at that position is not unique), while positions that are *closed* are not allowed to change (they are unique). Technically, this is achieved by specifying for each position in the head of an s-t tgd whether the corresponding value at that position should be *open*, or *closed*. We should remark that this approach is applicable to data exchange settings without target dependencies, and it was left open how to extend it to more general data exchange settings. Using this "mixed-world approach", we can solve the problem described in Example 5.6 as follows:

Example 5.7. The tgd in that example will become

$$\forall x\ (\textit{Person}(x) \rightarrow \exists z\ \textit{ParentChild}(x^{\text{closed}}, z^{\text{open}})),$$

indicating that while only people from the source are moved to the target as first attributes, the number of children associated with them is not restricted (i.e., is viewed under the OWA).

Afrati and Kolaitis [2008] show a different version of closed-world semantics to be useful for answering *queries with aggregates*. Their main point is that the CWA-solution-based semantics of this paper are too weak in this setting, since the instances in $\textit{Rep}_{\Sigma_t}(T)$ for CWA-solutions T may contain values that do not occur in the source instance. In their semantics, aggregate queries are answered by the certain answers over the endomorphic images of the canonical solution. That semantics, too, was developed for data exchange settings without target dependencies, and the behavior of such semantics with target dependencies was left open.

We should remark that most of the above-mentioned semantics heavily rely on the concrete presentation of the set of s-t tgds specifying the data exchange setting. This is true for the certain answers semantics and the maybe answers semantics on CWA-solutions as well as the semantics of Libkin and Sirangelo [2011] and Afrati and Kolaitis [2008]. The reason is that these semantics are defined in terms of the canonical solution, which is highly sensitive to slight variations of the data exchange setting. For example, consider the two data exchange settings $D_1 = (\{P\}, \{E\}, \{\theta_1\})$ and $D_2 = (\{P\}, \{E\}, \{\theta_2\})$, where θ_1 and θ_2 are respectively defined as $\forall x(P(x) \rightarrow \exists y\, E(x, y))$ and $\forall x(P(x) \rightarrow \exists y \exists z\, (E(x, y) \wedge E(x, z)))$. Although θ_1 and θ_2 are logically equivalent, the canonical solutions of source instances under D_1 and D_2 differ: the source instance $\{P(a)\}$, for example, has the canonical solution $\{E(a, \perp)\}$ under D_1, and the canonical solution $\{E(a, \perp), E(a, \perp')\}$ under D_2. Note that neither the potential certain answers semantics nor the persistent maybe answers semantics have this disadvantage.

To obtain unique answers for logically equivalent[8] data exchange settings, Gottlob et al. [2009] propose to first *normalize* the data exchange setting as described in their paper, and then apply the semantics. Another approach is to use the semantics developed in Hernich [2010]. That semantics does not build directly on the approach

[8] Fagin et al. [2008] considered different notions of equivalence between data exchange settings. Logical equivalence is the strongest such notion. Instead of unique query answers on logically equivalent data exchange settings, one could also require unique query answers on data exchange settings that are equivalent under any of the other notions of equivalence.

presented in this work, but is inspired by the original proposal of the CWA by Reiter [1978], and variants thereof. Furthermore, it can be applied to a broader class of data exchange settings. A discussion including a comparison with the above-mentioned semantics can be found in [Hernich 2010].

6. COMPLEXITY OF QUERY ANSWERING

Finally, we consider the complexity of answering queries under the new semantics. More precisely, given a data exchange setting D, a query $Q(\bar{x})$ over D's target schema, and answer $\in \{\mathsf{certain}_\Box, \mathsf{certain}_\Diamond, \mathsf{maybe}_\Box, \mathsf{maybe}_\Diamond\}$, we are interested in the complexity of the problem

$\textsc{Eval}_{\mathsf{answer}}(D, Q)$
Input: a source instance S for D, and a tuple $\bar{t} \in \mathsf{Dom}^{|\bar{x}|}$
Question: Is $\bar{t} \in \mathsf{answer}^D(Q, S)$?

Thus we deal with the *data complexity* of query answering.

The following definitions will be convenient. Let $\mathcal{D}$ be a class of data exchange settings, let $\mathcal{L}$ be a query language, and let $\mathcal{C}$ be a complexity class. We say that the data complexity of $\mathcal{L}$ with respect to answer and $\mathcal{D}$ is in $\mathcal{C}$ if for every data exchange setting $D \in \mathcal{D}$ and each query $Q \in \mathcal{L}$ over D's target schema, $\textsc{Eval}_{\mathsf{answer}}(D, Q)$ is in $\mathcal{C}$. We also say that the data complexity of $\mathcal{L}$ with respect to answer and $\mathcal{D}$ is $\mathcal{C}$-hard if there is a data exchange setting $D \in \mathcal{D}$ and a query $Q \in \mathcal{L}$ over D's target schema such that $\textsc{Eval}_{\mathsf{answer}}(D, Q)$ is hard for $\mathcal{C}$. Finally, we say that the data complexity of $\mathcal{L}$ with respect to answer and $\mathcal{D}$ is $\mathcal{C}$-complete if the data complexity of $\mathcal{L}$ with respect to answer and $\mathcal{D}$ is in $\mathcal{C}$, and $\mathcal{C}$-hard.

6.1. Complexity of Answering FO Queries

As the next proposition shows, the $\textsc{Eval}$-problem may be undecidable for the two semantics $\mathsf{certain}_\Box$ and $\mathsf{maybe}_\Diamond$, even with respect to weakly acyclic data exchange settings. For a proof see Appendix C.

PROPOSITION 6.1. *There is a weakly acyclic data exchange setting D and a Boolean FO query Q over D's target schema such that* $\textsc{Eval}_{\mathsf{certain}_\Box}(D, \neg Q)$ *and* $\textsc{Eval}_{\mathsf{maybe}_\Diamond}(D, Q)$ *are undecidable.*

We remark that Proposition 6.1 is based entirely on the fact that, given a data exchange setting D, a source instance S for D, a mapping $\alpha\colon \mathcal{J}_D^* \to \mathsf{Dom}$, and a tuple $(j, z) \in \mathcal{J}_D^*$ with $j = (d, \bar{u}, \bar{v})$, the value $\alpha(j, z)$ does not only depend on d, $\bar{u}$ and z, but also on $\bar{v}$. This makes it possible to cascade the creation of nulls even though the data exchange setting is weakly acyclic. The following restriction of weakly acyclic data exchange settings prohibits this.[9]

Definition 6.2 (*richly acyclic data exchange setting*).

- The *extended dependency graph* of a set Σ of tgds is obtained from the dependency graph of Σ (see Definition 4.1) as follows: for every tgd $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ in Σ, every variable y in $\bar{y}$, and every position (R, i) at which y appears in φ, add an existential edge from (R, i) to every position at which some variable from $\bar{z}$ appears in ψ.
- A set Σ of tgds is *richly acyclic* if no cycle in the extended dependency graph of Σ contains an existential edge.

[9] We remark that richly acyclic data exchange settings were considered also by Gottlob and Nash [2008], see the remark below Definition 1 in their paper. Our notion of richly acyclic corresponds to their Definition 1a.

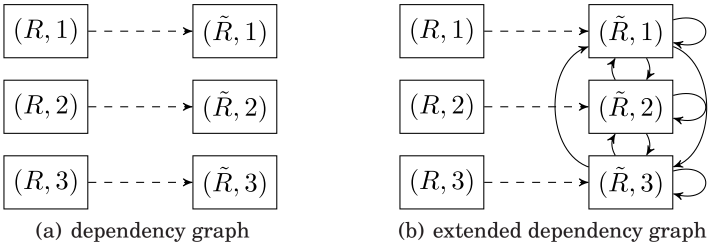


Fig. 2. The dependency graph of Σ (a), and the extended dependency graph of Σ (b). Dashed edges represent copying edges, while solid edges represent existential edges.

— A data exchange setting $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ is called *richly acyclic* if Σ_{t} is the union of a richly acyclic set of tgds, and a set of egds.

Note that every richly acyclic data exchange setting is weakly acyclic, but not vice versa as illustrated by the following example:

Example 6.3. Consider the set Σ consisting of the two tgds

$$R(x,y,z) \to \tilde{R}(x,y,z) \quad \text{and} \quad \tilde{R}(x,y,z) \to \exists x' \exists y' \exists z'\, \tilde{R}(x',y',z'),$$

which occur in the data exchange setting constructed in the proof of Proposition 6.1. The dependency graph of Σ is as shown in Figure 2(a). Since this graph contains no existential edges at all, Σ is weakly acyclic. On the other hand, the extended dependency graph of Σ, shown in Figure 2(b), contains existential edges. Moreover, it contains cycles through existential edges, which implies that Σ is not richly acyclic. □

Furthermore, the EVAL-problem is decidable for FO queries with respect to richly acyclic data exchange settings. More precisely, we have the following:

THEOREM 6.4. *Let* answer $\in \{\text{certain}_\Box, \text{certain}_\Diamond, \text{maybe}_\Box, \text{maybe}_\Diamond\}$. *Then the data complexity of FO with respect to* answer *and the class of richly acyclic data exchange settings as well as the class of weakly acyclic data exchange settings is as follows:*

answer	*richly acyclic settings*	*weakly-acyclic settings*
$\text{certain}_\Box$	co-NP-*complete*	*undecidable*
$\text{certain}_\Diamond$	co-NP-*complete*	co-NP-*complete*
$\text{maybe}_\Box$	NP-*complete*	NP-*complete*
$\text{maybe}_\Diamond$	NP-*complete*	*undecidable*

The upper bounds of Theorem 6.4 follow from Lemma 6.6 below. The lower bounds of Theorem 6.4 follow from Theorem 6.8 below, and Proposition 6.1.

Remark 6.5. Of course it has long been known that the data complexity of computing certain (resp. maybe) answers for FO queries is coNP-complete (NP-complete, respectively) [Abiteboul and Duschka 1998; Abiteboul et al. 1991], in the size of an instance T. However, here we measure the complexity of answering queries on CWA-solutions for S under D, in terms of the size of S. Thus, in the proof of Theorem 6.4, we have to provide hardness examples that, unlike those in [Abiteboul and Duschka 1998; Abiteboul et al. 1991], arise as CWA-solutions like $\text{CANSOL}_D(S)$ or $\text{CORE}_D(S)$ for some fixed data exchange setting D. □

LEMMA 6.6. *Let $D = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be a data exchange setting, and let $Q(\bar{x})$ be an FO query over D's target schema. Then:*

(*1*) *If the data exchange setting D is weakly acyclic, then* $\text{EVAL}_{\mathsf{certain}_\Diamond}(D,Q) \in \text{co-NP}$ *and* $\text{EVAL}_{\mathsf{maybe}_\Box}(D,Q) \in \text{NP}$.
(*2*) *If the data exchange setting D is richly acyclic, then* $\text{EVAL}_{\mathsf{certain}_\Box}(D,Q) \in \text{co-NP}$ *and* $\text{EVAL}_{\mathsf{maybe}_\Diamond}(D,Q) \in \text{NP}$.

PROOF. Let $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, and let $Q(\bar{x})$ be an FO query over τ.

Ad 1: By duality (i.e., for every tuple $\bar{t}$ we have $\bar{t} \in \mathsf{certain}^D_\Diamond(Q,S)$ precisely if $\bar{t} \notin \mathsf{maybe}^D_\Box(\neg Q, S)$), it suffices to show that $\text{EVAL}_{\mathsf{maybe}_\Box}(D,Q) \in \text{NP}$. Let S be a source instance for D, and let $\bar{t} \in \mathsf{Dom}^{|\bar{x}|}$. By Theorem 5.3, we have

$$\mathsf{maybe}^D_\Box(Q,S) \;=\; \Diamond_{\Sigma_\text{t}} Q(\text{CORE}_D(S))$$

if CWA-solutions for S under D exist. Note that if no CWA-solution exists, then by Corollary 3.12, $\text{CORE}_D(S)$ does not exist. In particular, the algorithm guaranteed by Theorem 4.2 will indicate that $\text{CORE}_D(S)$ does not exist, so that we can output $\mathsf{maybe}^D_\Box(Q,S) = \emptyset$. In the following we assume that CWA-solutions for S under D exist.

By Theorem 4.2, we can compute $T_0 := \text{CORE}_D(S)$ in time polynomial in the size of S. Thus, it remains to show that we can nondeterministically check whether $\bar{t} \in \Diamond_{\Sigma_\text{t}} Q(T_0)$ in time polynomial in the size of T_0.

We have $\bar{t} \in \Diamond_{\Sigma_\text{t}} Q(T_0)$ if and only if there is an instance $\hat{T} \in \textit{Rep}_{\Sigma_\text{t}}(T_0)$ such that $\bar{t} \in Q(\hat{T})$. Furthermore, if there is an instance $\hat{T} \in \textit{Rep}_{\Sigma_\text{t}}(T_0)$ with $\bar{t} \in Q(\hat{T})$, then it is easy to see that there is such an instance $\hat{T}$ with $\mathrm{dom}(\hat{T}) \subseteq C \cup f(\mathrm{null}(T_0))$, where C is the set of all constants that occur in T_0, Q, and $\bar{t}$, and $f\colon \mathrm{null}(T_0) \to \mathsf{Const} \setminus C$ is an injective mapping. Thus, we can check $\bar{t} \in \Diamond_{\Sigma_\text{t}} Q(T_0)$ by the following nondeterministic procedure:

(1) "Guess" a valuation $v\colon \mathrm{dom}(T_0) \to C \cup f(\mathrm{null}(T_0))$ of T_0.
(2) Check whether $\hat{T} := v(T_0)$ satisfies all target tgds and egds of Σ. If not, reject the input.
(3) If $\bar{t} \in q(\hat{T})$, accept the input. Otherwise reject it.

Clearly, this procedure runs in time polynomial in the size of T_0, which completes the proof of 1.

Ad 2: Again, by duality (i.e., $\bar{t} \in \mathsf{certain}^D_\Box(Q,S)$ if and only if $\bar{t} \notin \mathsf{maybe}^D_\Diamond(\neg Q, S)$), it suffices to show that $\text{EVAL}_{\mathsf{maybe}_\Diamond}(D,Q) \in \text{NP}$. Let S be a source instance for D, and let $\bar{t} \in \mathsf{Dom}^{|\bar{x}|}$. Then, $\bar{t} \in \mathsf{maybe}^D_\Diamond(Q,S)$ if and only if there are a CWA-solution T for S under D with $\bar{t} \in \Diamond_{\Sigma_\text{t}} Q(T)$. Therefore, the following nondeterministic algorithm decides whether $\bar{t}$ belongs to $\mathsf{maybe}^D_\Diamond(Q,S)$:

(1) Compute $T_0 := \text{CORE}_D(S)$.
(2) Generate a successful α-chase sequence C of S with D, "guessing" the relevant values for $\alpha\colon \mathcal{J}^*_D \to \mathsf{Dom}$ "along the way".[10] Let $S \cup T$ be the result of C.
(3) If T does not satisfy the egds of Σ_t, reject S and $\bar{t}$.
(4) Check whether there is a homomorphism from T to T_0. If not, reject the input.
(5) If $\bar{t} \in \Diamond_{\Sigma_\text{t}} Q(T)$, then accept S and $\bar{t}$. Otherwise reject.

[10] Note that we can restrict attention to mappings $\alpha\colon \mathcal{J}^*_D \to \mathrm{dom}(S) \cup C \cup \mathsf{Null}$, where C is the set of all constants that occur in $\Sigma_{\text{st}} \cup \Sigma_\text{t}$ – all other choices do not lead to CWA-solutions for S under D.

Note that step 2–4 guarantee that T is a CWA-solution for S under D: step 2 and 3 ensure that T is a CWA-presolution for S under D, while step 4 ensures that T is a universal solution for S under D. Thus, the algorithm indeed checks whether $\bar{t} \in \mathsf{maybe}^D_\diamond(Q, S)$. Furthermore, the algorithm runs in polynomial time: By Theorem 4.2, we can accomplish step 1 in polynomial time. By the proof of Theorem 3.9 in Fagin et al. [2005a], step 2 can be accomplished in polynomial time as well. It is also easy to see that steps 3 and 4 can be accomplished in polynomial time. Finally, we have shown in part 1 of the proof that step 5 can be accomplished in polynomial time. □

Remark 6.7. Lemma 6.6 still holds if Q is a query with polynomial time data complexity. Recall that a query Q has polynomial time data complexity if the language $\{enc(I)\#enc(\bar{t}) \mid \bar{t} \in Q(I)\}$ is in PTIME, where *enc*(I) and *enc*$(\bar{t})$ are encodings of an instance I and a tuple $\bar{t}$ over $\mathrm{dom}(I)$, respectively. □

For the hardness results of Theorem 6.4, we can use:

THEOREM 6.8 ([MĄDRY 2005]). *There is a data exchange setting D without target dependencies, and a Boolean conjunctive query Q with two inequalities such that the following problem is* co-NP*-complete: given a source instance S for D, decide whether the certain answers of Q on S with respect to D are non-empty.*

Mądry's proof is formulated for the certain answers semantics of Fagin et al. [2005a], but it is not hard to see that it carries over to the semantics $\mathsf{certain}_\square$ and $\mathsf{certain}_\diamond$. By duality, we then get the hardness results concerning the remaining two semantics $\mathsf{maybe}_\square$ and $\mathsf{maybe}_\diamond$.

6.2. Complexity of Answering Positive Queries

We now turn to the case of positive queries, such as unions of conjunctive queries (UCQ, for short) or Datalog queries. This case was most heavily studied in the context of data exchange [Fagin et al. 2005a; Fagin et al. 2005b; Mądry 2005]. We show that such positive queries can be evaluated in polynomial time under the two certain answer-based semantics $\mathsf{certain}_\square$ and $\mathsf{certain}_\diamond$.

First recall that [Fagin et al. 2005a; Fagin et al. 2005b] and others follow the naive approach to evaluation of queries on target instances. In this approach, the answer to a query $Q(\bar{x})$ on a target instance T is simply the set $Q(T)$ of all tuples $\bar{t}$ over $\mathsf{Const} \cup \mathsf{Null}$ such that $T \models Q(\bar{t})$. So it is assumed that the domain of the database comes from $\mathsf{Const} \cup \mathsf{Null}$, and thus the equality predicate is available on the entire domain; in particular, two nulls are equal if they are just symbolically the same null. This corresponds precisely to query evaluation over *naive tables* [Abiteboul et al. 1995; Imielinski and Lipski, Jr. 1984].

Based on this naive evaluation, [Fagin et al. 2005a; Fagin et al. 2005b] proposed a semantics for evaluating conjunctive queries with respect to weakly acyclic data exchange settings, which happened to coincide with their notion of certain answers. For a target instance T, define $T_\downarrow$ as the instance T from which all tuples containing nulls have been removed. Then the evaluation function for conjunctive queries from [Fagin et al. 2005a; Fagin et al. 2005b] is

$$\mathsf{CQ\_eval}^D(Q, S) \ := \ Q(T)_\downarrow,$$

where D is a weakly acyclic data exchange setting, Q is a conjunctive query over D's target schema, S is a source instance for D, and T is an arbitrary universal solution for S under D. It turns out that this is precisely what two of the semantics we studied here do for the class of unions of conjunctive queries (also known as positive relational algebra queries, i.e., $\{\sigma, \pi, \bowtie, \cup\}$-queries in which selection predicates are positive Boolean combinations of equalities).

Recall that unions of conjunctive queries are preserved under homomorphisms (see, e.g., Chandra and Merlin [1977]), i.e., if Q is a union of conjunctive queries, I, J are instances such that there is a homomorphism h from I to J, and $\bar{t} \in Q(I)$, then $h(\bar{t}) \in Q(J)$.

LEMMA 6.9. *Let $D = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be a data exchange setting, S a source instance for D, and Q a query over τ that is preserved under homomorphisms. Then, for every CWA-solution T for S under D we have*

$$\mathsf{certain}^D_\Box(Q, S) \;=\; \mathsf{certain}^D_\Diamond(Q, S) \;=\; Q(T)_\downarrow.$$

PROOF. We first prove the following intermediate claim:

(*) If T and T' are homomorphically equivalent target instances for D with $T' \models \Sigma_\mathrm{t}$, then $\Box_{\Sigma_\mathrm{t}} Q(T') = Q(T)_\downarrow$.

Let T and T' be target instances for D such that $T' \models \Sigma_\mathrm{t}$. Furthermore, let h be a homomorphism from T' to T, and let h' be a homomorphism from T to T'.

We first show that $\Box_{\Sigma_\mathrm{t}} Q(T') \subseteq Q(T)_\downarrow$. Let $\bar{t} \in \Box_{\Sigma_\mathrm{t}} Q(T')$. Then for all $\hat{T} \in \mathit{Rep}_{\Sigma_\mathrm{t}}(T')$ we have $\bar{t} \in Q(\hat{T})$. This implies that $\bar{t} \in Q(T')$ (here we need that T' satisfies Σ_t), and that $\bar{t}$ consists entirely of constants. Since h is a homomorphism from T' to T, and Q is preserved under homomorphisms, this leads to $h(\bar{t}) \in Q(T)$. Since $\bar{t}$ consists entirely of constants, and h is the identity on constants, we conclude that $\bar{t} \in Q(T)_\downarrow$.

The proof for $Q(T)_\downarrow \subseteq \Box_{\Sigma_\mathrm{t}} Q(T')$ is pretty much the same: If $\bar{t} \in Q(T)_\downarrow$, then we have $\bar{t} \in Q(T)$, and that $\bar{t}$ consists entirely of constants. The remaining part of the proof is then the same, except that T and T' must be interchanged, and h must be replaced with h'.

Now, by Theorem 3.9, every two CWA-solution for S under D are homomorphically equivalent, and satisfy Σ_t. Therefore, for every CWA-solution T for S under D we have

$$\mathsf{certain}_\Box(Q, S) \;=\; \bigcap_{T' \in [\![S]\!]_{\mathrm{CWA}}} \Box_{\Sigma_\mathrm{t}} Q(T') \;\overset{(*)}{=}\; \bigcap_{T' \in [\![S]\!]_{\mathrm{CWA}}} Q(T)_\downarrow \;=\; Q(T)_\downarrow$$

and

$$\mathsf{certain}_\Diamond(Q, S) \;\overset{\text{Thm. 5.3}}{=}\; \Box_{\Sigma_\mathrm{t}} Q(\textsc{Core}(S)) \;\overset{(*)}{=}\; Q(T)_\downarrow. \qquad \Box$$

Remark 6.10. Concerning the *maybe answer*-based semantics, even for quantifier-free conjunctive queries we may have $\mathsf{maybe}^D_\Box(Q, S) \neq \mathsf{maybe}^D_\Diamond(Q, S)$. For example, it is easy to find a data exchange setting D with a single target relation R, and a source instance S for D so that $\textsc{CanSol}_D(S) = \{R(a, \bot_1), R(a, \bot_2)\}$. Then we have $\textsc{Core}_D(S) = \{R(a, \bot)\}$. Thus, if Q is defined such that $Q(x, y, z) := R(x, y) \wedge R(x, z)$, then $\mathsf{maybe}^D_\Box(Q, S) = \Diamond Q(\textsc{Core}(S)) \subsetneq \Diamond Q(\textsc{CanSol}(S)) = \mathsf{maybe}^D_\Diamond(Q, S)$. $\Box$

Now we can show that unions of conjunctive queries can be answered efficiently with respect to the two semantics $\mathsf{certain}_\Box$ and $\mathsf{certain}_\Diamond$, and weakly acyclic data exchange settings.

PROPOSITION 6.11. *Let $D = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be a weakly acyclic data exchange setting, let Q be a union of conjunctive queries over τ, and let* answer *be one of* $\mathsf{certain}_\Box$ *and* $\mathsf{certain}_\Diamond$. *Then there is a polynomial time algorithm that, given a source instance S for D, computes* $\mathsf{answer}^D(Q, S)$.

PROOF. To compute $\mathsf{answer}^D(Q,S)$, we can proceed as follows: First, we generate an arbitrary CWA-solution for S under D. Since D is weakly acyclic, Theorem 4.2 tells us that this is possible in time polynomial in the size of S. By Lemma 6.9, we can now return the set $Q(T)_\downarrow$ as the result of $\mathsf{answer}^D(Q,S)$. (Recall that a fixed UCQ query can be evaluated in time polynomial in the size of the input instance [Abiteboul et al. 1995].) □

Note that Proposition 6.11 is still true if Q is not a UCQ query, but satisfies the following two requirements instead: (1) Q is preserved under homomorphisms, and (2) Q can be evaluated in time polynomial in the size of the input instance. So, for example, if Q is a Datalog query, then Proposition 6.11 remains true.

The complexity bound of Proposition 6.11 is tight in the following sense; see Appendix D for a proof.

PROPOSITION 6.12. *Let* $\mathsf{answer} \in \{\mathsf{certain}_\Box, \mathsf{certain}_\Diamond\}$. *Then the data complexity of unions of conjunctive queries with respect to* answer *and the class of weakly acyclic data exchange settings is* PTIME-*hard.*

A slight extension of unions of conjunctive queries, where each conjunct can contain an inequality $x \neq y$ between variables x and y, increases the complexity of query evaluation. Let D be a weakly acyclic data exchange setting, and let Q be a union of conjunctive queries with at most one inequality per disjunct. Recall from Fagin et al. [2005a] that there is a polynomial time algorithm $\mathbb{A}$ that, given a source instance S for D, computes the certain answers of Q on S with respect to D. It is easy to see that $\mathbb{A}$ can also be used to compute $\mathsf{certain}_\Box^D(Q,S)$ and $\mathsf{certain}_\Diamond^D(Q,S)$, as long as the target dependencies of D consist of egds, or all s-t tgds and target tgds of D are full. However, if we do not impose this restriction on D, then Theorem 6.13 below implies that even if D is richly acyclic, it is unlikely that there is a polynomial time algorithm that computes $\mathsf{certain}_\Box^D(Q,S)$ or $\mathsf{certain}_\Diamond^D(Q,S)$ on input S.

THEOREM 6.13. *There is a richly acyclic data exchange setting* D *and a conjunctive query* Q *with one inequality such that:*

(*1*) $\textsc{Eval}_{\mathsf{certain}_\Box}(D,Q)$ *and* $\textsc{Eval}_{\mathsf{certain}_\Diamond}(D,Q)$ *are* co-NP-*complete.*
(*2*) $\textsc{Eval}_{\mathsf{maybe}_\Box}(D,\neg Q)$ *and* $\textsc{Eval}_{\mathsf{maybe}_\Diamond}(D,\neg Q)$ *are* NP-*complete.*

PROOF. We prove only 1, since 2 follows by duality. Let the source schema of D be $\{R,C,L\}$, and the target schema $\{R',C',L'\}$. Let the source-to-target dependencies and target dependencies of D consist of

— $R(i,j,p) \to R'(i,j,p)$,
— $C(i) \to \exists t\, C'(i,t)$,
— $L(j,p) \to \exists t\, L'(j,p,t)$,

and let the target dependencies Σ_t of D consist of

— $C'(i,1) \to \exists j,p\big(R'(i,j,p) \wedge L'(j,p,1)\big)$,
— $L'(j,p,1) \wedge L'(j,p',1) \to p = p'$.

Finally, let

$$Q \;:=\; \exists i \exists t\, (C'(i,t) \wedge t \neq 1)\,.$$

Note that D is richly acyclic. Hence, $\textsc{Eval}_{\mathsf{certain}_\Box}(D,Q)$ and $\textsc{Eval}_{\mathsf{certain}_\Diamond}(D,Q)$ are in co-NP by Lemma 6.6. To prove co-NP-hardness, we give a reduction from the complement of the NP-complete SAT, the satisfiability problem for propositional formulas in conjunctive normal form (see, e.g., [Papadimitriou 1994]).

The reduction is carried out as follows. On input of a propositional formula

$$\varphi(x_1, x_2, \dots, x_n) \ := \ C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

in conjunctive normal form, we construct the source instance

$$\begin{aligned} S_\varphi \ := \ & \{R(i,j,1) \mid x_j \text{ occurs in } C_i\} \ \cup \ \{R(i,j,0) \mid \neg x_j \text{ occurs in } C_i\} \\ & \cup \{C(i) \mid 1 \le i \le m\} \ \cup \ \{L(j,b) \mid 1 \le j \le n \text{ and } b \in \{0,1\}\}. \end{aligned}$$

Note that there is exactly one CWA-solution for S_φ, denoted by T_φ. It consists of a copy of R, contains for each $i \in \{1,2,\dots,m\}$ an atom $C'(i,\bot_i)$, and for each $j \in \{1,\dots,n\}$ and $b \in \{0,1\}$ an atom $L'(j,b,\bot_{j,b})$, where the nulls $\bot_i, \bot_{j,b}$ introduced for each i, j and b are pairwise distinct. Therefore, $\mathsf{certain}_\Box(Q,S_\varphi) = \mathsf{certain}_\Diamond(Q,S_\varphi) = \Box_{\Sigma_t} Q(T_\varphi)$. We claim that φ is satisfiable iff $\Box_{\Sigma_t} Q(T_\varphi) = \emptyset$.

($\Longrightarrow$) Let $\alpha \colon \{x_1, x_2, \dots, x_n\} \to \{0,1\}$ be a satisfying truth assignment for φ. We extend α to negated variables by $\alpha(\neg x_i) = 1 - \alpha(x_i)$. Then, for every $i \in \{1,\dots,m\}$ there is a literal ℓ in C_i such that $\alpha(\ell) = 1$. Define a valuation v of T_φ such that for each $\bot \in \mathrm{null}(T_\varphi)$,

$$v(\bot) = \begin{cases} 1 & \text{if } \bot = \bot_i \\ \alpha(x_j) & \text{if } \bot = \bot_{j,1} \\ \alpha(\neg x_j) & \text{if } \bot = \bot_{j,0}. \end{cases}$$

Then, $v(T_\varphi)$ satisfies the target dependencies of D, i.e., $v(T_\varphi) \in \boldsymbol{Rep}_{\Sigma_t}(T_\varphi)$, but $v(T_\varphi)$ does not satisfy Q. Therefore, $\Box_{\Sigma_t} Q(T_\varphi) = \bigcap_{\hat{T} \in \boldsymbol{Rep}_{\Sigma_t}(T_\varphi)} Q(\hat{T}) = \emptyset$.

($\Longleftarrow$) Assume now that $\Box_{\Sigma_t} Q(T_\varphi) = \emptyset$. Then $\boldsymbol{Rep}_{\Sigma_t}(T_\varphi)$ contains an instance $\hat{T}$ that does not satisfy Q. Define a truth assignment $\alpha \colon \{x_1,\dots,x_n\} \to \{0,1\}$ such that for each $j \in \{1,\dots,n\}$,

$$\alpha(x_j) = \begin{cases} 1 & \text{if } L'(j,1,1) \in \hat{T} \\ 0 & \text{otherwise}, \end{cases}$$

and extend it to negated variables as above. We claim that α satisfies φ; i.e., for each $i \in \{1,\dots,m\}$ there is a literal ℓ in C_i with $\alpha(\ell) = 1$.

Let $i \in \{1,\dots,m\}$. Then $C(i,1) \in \hat{T}$, because $\hat{T} \not\models Q$. Since $\hat{T}$ satisfies the target dependencies of D, there are $j \in \{1,\dots,n\}$ and $p \in \{0,1\}$ such that $R'(i,j,p)$ and $L'(j,p,1)$ are in $\hat{T}$, and $L'(j,1-p,1) \notin \hat{T}$. If $p = 0$, then we have $R'(i,j,0) \in \hat{T}$ and $L'(j,1,1) \notin \hat{T}$. $R'(i,j,0) \in \hat{T}$ indicates that the literal $\neg x_j$ occurs in C_i, and $L'(j,1,1) \notin \hat{T}$ indicates that $\alpha(\neg x_j) = 1 - \alpha(x_j) = 1$. So, if $p = 0$, then C_i is satisfied under α. It remains therefore to show that C_i is satisfied under α if $p = 1$. If $p = 1$, then $R'(i,j,1)$ and $L'(j,1,1)$ are in $\hat{T}$. $R'(i,j,1) \in \hat{T}$ indicates that the literal x_j occurs in C_i, and $L'(j,1,1) \in \hat{T}$ indicates that $\alpha(x_j) = 1$. Consequently, C_i is satisfied under α. □

Table II summarizes the present section's results on the complexity of the problems $\textsc{Eval}_{\mathsf{certain}_\Box}(D,Q)$ and $\textsc{Eval}_{\mathsf{certain}_\Diamond}(D,Q)$ for various restrictions of the data exchange setting D and the query language from which Q is chosen.

7. CONCLUDING REMARKS

In this paper, we introduced CWA-solutions as a new concept of solutions for data exchange that is based on the closed world assumption. Using CWA-solutions, we then developed new query answering semantics which do not suffer from some of the known anomalies of query answering with respect to the certain answers semantics

Table II. Complexity of $\textsc{Eval}_{\mathsf{certain}_\Box}(D, Q)$ and $\textsc{Eval}_{\mathsf{certain}_\Diamond}(D, Q)$ for certain restrictions of D and Q.

		query language		
		UCQ	UCQ with at most one inequality per disjunct	FO
restriction of data exchange setting	weakly acyclic	PTIME (Prop. 6.11)	co-NP-hard (Thm. 6.13)	undecidable (Prop. 6.1)
	richly acyclic		co-NP-complete (Lem. 6.6 & Thm. 6.13)	co-NP-complete (Lem. 6.6 & Thm. 6.8)
	only s-t tgds, egds		PTIME (Fagin et al. [2005a], Thm. 5.12)	
	only s-t tgds			
	only full tgds, egds			PTIME (folklore)

and the universal solution-based certain answers semantics. These semantics further confirmed the special status of the canonical solution and the core in data exchange. In fact, for the more common certain answers semantics, our results indicate that the canonical solution (or a solution that behaves like the canonical solution with respect to Rep_{Σ_t}) is the preferred solution to materialize.

Nevertheless, there is still much left to do. Extensions and variations on the theme of CWA-solutions appeared already. For example, Libkin and Sirangelo [2011] argued that in some scenarios the closed-world semantics may be too restrictive, and showed how to obtain solutions that combine the OWA and the CWA semantics. It was left open however how to add target constraints to such mixed data exchange settings. In [Afrati and Kolaitis 2008], a different version of closed-world semantics is shown to be useful for answering queries with aggregates. In that semantics, a CWA-solution must be contained in the canonical solution, rather than be its homomorphic image. Again, nothing is known about the behavior of such semantics with target constraints.

The results of Libkin and Sirangelo [2011] indicate that with respect to the composition operation, the closed-world semantics behaves similarly to the open-world semantics. Nothing however is known about the interaction of CWA-solutions and other operators on schema mappings such as inverses [Fagin 2007] and recoveries [Arenas et al. 2009].

We reduced query answering in data exchange to query answering over naive tables, which may be intractable for queries outside of the positive fragment of relational algebra. It would be nice to find ways to overcome this; for example, by finding easily constructible and fairly large subsets of certain answers.

And, finally, data exchange techniques have recently been looked at in the XML context [Arenas and Libkin 2008]. There is no clearly defined concept of a good solution in that case (as the analog of the canonical solution may fail to satisfy schema specifications), nor well-defined techniques for answering queries with incomplete information. Thus defining a proper semantics for solutions and query answering for XML remains open.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENT

We thank Marcelo Arenas, Pablo Barceló, and Ronald Fagin for their comments on earlier versions of this paper, and the anonymous referees for valuable comments and suggestions. The first and the third author were supported by DFG grants SCHW 837/3-1 and SCHW 837/3-2. The second author was supported by EPSRC grants grants G049165 and F028288, and FET-Open Project FoX (grant agreement 233599).

REFERENCES

ABITEBOUL, S. AND DUSCHKA, O. M. 1998. Complexity of answering queries using materialized views. In *Proceedings of the 17th ACM Symposium on Principles of Database Systems (PODS)*. 254–263.

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley.

ABITEBOUL, S., KANELLAKIS, P. C., AND GRAHNE, G. 1991. On the representation and querying of sets of possible worlds. *Theoretical Computer Science 78,* 1, 159–187.

AFRATI, F. N. AND KOLAITIS, P. G. 2008. Answering aggregate queries in data exchange. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*. 129–138.

ARENAS, M., BARCELÓ, P., FAGIN, R., AND LIBKIN, L. 2004. Locally consistent transformations and query answering in data exchange. In *Proceedings of the 23th ACM Symposium on Principles of Database Systems (PODS)*. 229–240.

ARENAS, M., BARCELÓ, P., LIBKIN, L., AND MURLAK, F. 2010. *Relational and XML Data Exchange*. Morgan & Claypool.

ARENAS, M., BARCELÓ, P., AND REUTTER, J. 2009. Query languages for data exchange: Beyond unions of conjunctive queries. In *Proceedings of the 12th International Conference on Database Theory (ICDT)*. 73–83.

ARENAS, M. AND LIBKIN, L. 2008. XML data exchange: Consistency and query answering. *Journal of the ACM 55,* 2, Article 7.

ARENAS, M., PÉREZ, J., AND RIVEROS, C. 2009. The recovery of a schema mapping: Bringing exchanged data back. *ACM Transactions on Database Systems 34,* 4, Article 22.

BARCELÓ, P. 2009. Logical foundations of relational data exchange. *SIGMOD Record 38,* 1, 49–58.

BEERI, C. AND VARDI, M. Y. 1984. A proof procedure for data dependencies. *Journal of the ACM 31,* 4, 718–741.

BUNEMAN, P., DAVIDSON, S. B., AND WATTERS, A. 1991. A semantics for complex objects and approximate answers. *Journal of Computer and System Sciences 43,* 1, 170–218.

BUNEMAN, P., JUNG, A., AND OHORI, A. 1991. Using powerdomains to generalize relational databases. *Theoretical Computer Science 91,* 1, 23–55.

CALÌ, A., GOTTLOB, G., AND KIFER, M. 2008. Taming the infinite chase: Query answering under expressive relational constraints. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. 70–80.

CHANDRA, A. K. AND MERLIN, P. M. 1977. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th ACM Symposium on Theory of Computing (STOC)*. 77–90.

DATE, C. J. AND DARWEN, H. 1997. *A Guide to the SQL Standard*. Addison-Wesley.

DEUTSCH, A., NASH, A., AND REMMEL, J. 2008. The chase revisited. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*. 149–158.

DEUTSCH, A. AND TANNEN, V. 2003. Reformulation of XML queries and constraints. In *Proceedings of the 9th Conference on Database Theory (ICDT)*. 225–241.

FAGIN, R. 2007. Inverting schema mappings. *ACM Transactions on Database Systems 32,* 4, Article 25.

FAGIN, R., KOLAITIS, P. G., MILLER, R. J., AND POPA, L. 2005a. Data exchange: Semantics and query answering. *Theoretical Computer Science 336,* 1, 89–124.

FAGIN, R., KOLAITIS, P. G., NASH, A., AND POPA, L. 2008. Towards a theory of schema-mapping optimization. In *Proceedings of the 27th Symposium on Principles of Database Systems (PODS)*. 33–42.

FAGIN, R., KOLAITIS, P. G., AND POPA, L. 2005b. Data exchange: Getting to the core. *ACM Transactions on Database Systems 30,* 1, 174–210.

FAGIN, R., KOLAITIS, P. G., POPA, L., AND TAN, W. C. 2005. Composing schema mappings: Second-order dependencies to the rescue. *ACM Transactions on Database Systems 30,* 4, 994–1055.

FUXMAN, A., KOLAITIS, P. G., MILLER, R. J., AND TAN, W. C. 2006. Peer data exchange. *ACM Transactions on Database Systems 31,* 4, 1454–1498.

GOTTLOB, G. AND NASH, A. 2008. Efficient core computation in data exchange. *Journal of the ACM 55,* 2, Article 9.

GOTTLOB, G., PICHLER, R., AND SAVENKOV, V. 2009. Normalization and optimization of schema mappings. *PVLDB 2,* 1, 1102–1113.

GUNTER, C. A. 1992. The mixed powerdomain. *Theoretical Computer Science 103,* 2, 311–334.

HAAS, L. M., HERNÁNDEZ, M. A., HO, H., POPA, L., AND ROTH, M. 2005. Clio grows up: From research prototype to industrial tool. In *Proceedings of the 31th ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 805–810.

HELL, P. AND NEŠETŘIL, J. 1992. The core of a graph. *Discrete Mathematics 109,* 1-3, 117–126.

HERNICH, A. 2010. Answering non-monotonic queries in relational data exchange. In *Proceedings of the 13th International Conference on Database Theory (ICDT)*. 143–154.

HERNICH, A. AND SCHWEIKARDT, N. 2007. CWA-solutions for data exchange settings with target dependencies. In *Proceedings of the 26th ACM Symposium on Principles of Database Systems (PODS)*. 113–122.

HERNICH, A. AND SCHWEIKARDT, N. 2010. Logic and data exchange: Which solutions are "good" solutions? In *Logic and the Foundations of Game and Decision Theory (LOFT 8)*, G. Bonanno, B. Löwe, and W. van der Hoek, Eds. Lecture Notes in Computer Science Series, vol. 6006. Springer-Verlag, 61–85.

IMIELINSKI, T. AND LIPSKI, JR., W. 1984. Incomplete information in relational databases. *Journal of the ACM 31,* 4, 761–791.

KOLAITIS, P. G. 2005. Schema mappings, data exchange, and metadata management. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS)*. 61–75.

KOLAITIS, P. G., PANTTAJA, J., AND TAN, W. C. 2006. The complexity of data exchange. In *Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS)*. 30–39.

LAUSEN, G., MEIER, M., AND SCHMIDT, M. 2009. On chase termination beyond stratification. CoRR, arXiv:0906.4228v2.

LIBKIN, L. 1998. Models of approximation in databases. *Theoretical Computer Science 190,* 2, 167–210.

LIBKIN, L. 2006. Data exchange and incomplete information. In *Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS)*. 60–69.

LIBKIN, L. AND SIRANGELO, C. 2011. Data exchange and schema mappings in open and closed worlds. *Journal of Computer and System Sciences*. To appear. Extended abstract in Proc. PODS 2008.

LIPSKI, JR., W. 1979. On semantic issues connected with incomplete information in databases. *ACM Transactions on Database Systems 4,* 3, 262–296.

MĄDRY, A. 2005. Data exchange: On the complexity of answering queries with inequalities. *Information Processing Letters 94*, 253–257.

MARNETTE, B. 2009. Generalized schema mappings: From termination to tractability. In *Proceedings of the 28th ACM Symposium on Principles of Database Systems (PODS)*. 13–22.

MECCA, G., PAPOTTI, P., AND RAUNICH, S. 2009. Core schema mappings. In *ACM SIGMOD Conference*. 655–668.

MILLER, R. J., HERNÁNDEZ, M. A., HAAS, L. M., YAN, L.-L., HO, C. T. H., FAGIN, R., AND POPA, L. 2001. The Clio project: Managing heterogeneity. *SIGMOD Record 30,* 1, 78–83.

PAPADIMITRIOU, C. H. 1994. *Computational Complexity*. Addison-Wesley.

REITER, R. 1978. On closed world databases. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, 55–76.

REITER, R. 1984. Towards a logical reconstruction of relational database theory. In *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, Eds. Springer-Verlag, 191–233.

SHU, N. C., HOUSEL, B. C., TAYLOR, R. W., GHOSH, S. P., AND LUM, V. Y. 1977. EXPRESS: A data EXtraction, Processing, and REStructuring system. *ACM Transactions on Database Systems 2,* 2, 134–174.

VAN DER MEYDEN, R. 1998. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*. Kluwer, 307–356.

VARDI, M. Y. 1982. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC)*. 137–146.

ZANIOLO, C. 1984. Database relations with null values. *Journal of Computer and System Sciences 28,* 1, 142–166.

Received Month Year; revised Month Year; accepted Month Year

Online Appendix to: Closed World Data Exchange

ANDRÉ HERNICH, Humboldt-Universität zu Berlin, Germany
LEONID LIBKIN, University of Edinburgh, United Kingdom
NICOLE SCHWEIKARDT, Goethe-Universität Frankfurt am Main, Germany

A. MAXIMAL CWA-SOLUTIONS

In this section we justify Definition 3.14 on extended canonical solutions, and we illustrate that for general data exchange settings there are no maximal CWA-solutions in the sense of maximality of the (extended) canonical solution.

We begin by justifying Definition 3.14. Note that if $D = (\sigma, \tau, \Sigma_{\mathrm{st}}, \Sigma_{\mathrm{t}})$ is a data exchange setting where all tgds that occur in $\Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}}$ are full, then every source instance S for D has at most one CWA-solution for S under D. If there is a CWA-solution, say T, we can define $\textsc{CanSol}_D(S) := T$, and if there is none, then we let $\textsc{CanSol}_D(S)$ be undefined. We now turn to the case that Σ_{t} consists of egds:

PROPOSITION A.1. *Let $D = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be a data exchange setting where Σ_t consists of egds, and let S be a source instance for D with at least one CWA-solution under D. Then there is a unique CWA-solution T for S under D such that every CWA-solution for S under D is a homomorphic image of T.*

PROOF. Consider a complete chase sequence $C = (T_0, T_1, \ldots, T_m)$ of the canonical solution $\textsc{CanSol}_{(\sigma,\tau,\Sigma_{\mathrm{st}})}(S)$ with Σ_{t}. Then C is successful. Otherwise, by [Fagin et al. 2005a, Theorem 3.3], there would be no solution for S under D, contradicting the assumption that there is a CWA-solution for S under D. Define $T := T_m$, and note that T is a CWA-solution for S under D. Moreover, T is unique up to isomorphism [Beeri and Vardi 1984]. It remains to prove that every CWA-solution for S under D is a homomorphic image of T.

To this end we show by induction on s that for every $s \in \{0, 1, \ldots, m\}$, every CWA-solution for S under D is a homomorphic image of T_s. For the base case, note that every CWA-solution for S under D is a homomorphic image of $T_0 = \textsc{CanSol}_{(\sigma,\tau,\Sigma_{\mathrm{st}})}(S)$. This follows from Proposition 3.3 together with the fact that Σ_{t} consists of egds, and hence, every CWA-solution for S under D is a CWA-*pre*solution for S under $(\sigma, \tau, \Sigma_{\mathrm{st}})$.

Suppose now that for some $s \in \{0, \ldots, m-1\}$, every CWA-solution for S under D is a homomorphic image of T_s. Then we can argue in precisely the same way as in Case 2 of the proof of Lemma 3.4 in Fagin et al. [2005a]. In particular, let T' be an arbitrary CWA-solution for S under D. By the induction hypothesis, there is a homomorphism h from T_s to T with $h(T_s) = T$. Case 2 of the proof of Lemma 3.4 in Fagin et al. [2005a] shows that h is also a homomorphism from T_{s+1} to T with $h(T_{s+1}) = h(T_s) = T$, which completes the induction step. □

We now show that in general there is no CWA-solution that is maximal in the above sense. The following example exhibits a data exchange setting D' based on the data exchange setting from Remark 3.16, and a source instance S where this is the case. In fact, D' has the property that for every positive integer n there is a source instance S for D', and 2^n distinct CWA-solutions $T_1, \ldots, T_{2^n}$ for S under D' such that for every CWA-solution T for S under D' there is a homomorphism h from exactly one T_i to T with $h(T_i) = T$.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Example A.2. Recall the data exchange setting $D = (\{R\}, \{R'\}, \Sigma_{\text{st}})$ from Remark 3.16, where $R(x, y) \to \exists z_1 \exists z_2\, R'(x, z_1, z_2)$ is the unique tgd in Σ_{st}. Let D' be the data exchange setting obtained from D by adding the target tgd

$$R'(x, x_1, y) \wedge R'(x, x_2, y) \to R''(x, x_1, x_2).$$

Then, given the source instance $S' = \{R(a, b), R(a, c)\}$, both

$$T_1 \;:=\; \{R'(a, \bot_1, \bot_2), R'(a, \bot_3, \bot_4), R''(a, \bot_1, \bot_1), R''(a, \bot_3, \bot_3)\}$$

and

$$\begin{aligned} T_2 \;:=\; \{&R'(a, \bot_1, \bot_2), R'(a, \bot_3, \bot_2),\\ &R''(a, \bot_1, \bot_1), R''(a, \bot_3, \bot_3), R''(a, \bot_1, \bot_3), R''(a, \bot_3, \bot_1)\}\end{aligned}$$

are CWA-solutions for S' under D'.

Note that every CWA-solution for S' under D' is a homomorphic image of T_1 or T_2. Indeed, let T be a CWA-solution for S' under D'. Then there are (not necessarily distinct) nulls $\bot_1', \bot_2', \bot_3', \bot_4'$ such that $(R')^T = \{(a, \bot_1', \bot_2'), (a, \bot_3', \bot_4')\}$. If $\bot_2' \neq \bot_4'$, we have $(R'')^T = \{(a, \bot_1', \bot_1'), (a, \bot_3', \bot_3')\}$, so that $T = h_1(T_1)$, where $h_1(a) = a$, and $h_1(\bot_i) = \bot_i'$ for every $i \in \{1, 2, 3, 4\}$. Furthermore, if $\bot_2' = \bot_4'$, we have $(R'')^T = \{(a, \bot_1', \bot_1'), (a, \bot_3', \bot_3'), (a, \bot_1', \bot_3'), (a, \bot_3', \bot_1')\}$, so that $T = h_2(T_2)$, where $h_2(a) = a$, and $h_2(\bot_i) = \bot_i'$ for every $i \in \{1, 2, 3, 4\}$.

However, T_1 is no homomorphic image of T_2 and vice versa: If T is a CWA-solution for S' under D' such that T_1 is a homomorphic image of T, then T must contain atoms of the form $R'(a, \bot_1', \bot_2')$ and $R'(a, \bot_3', \bot_4')$, and therefore, T must be isomorphic to T_1. In particular, T_1 is no homomorphic image of T_2. On the other hand, if T is a CWA-solution for S' under D' such that T_2 is a homomorphic image of T, then T must contain atoms of the form $R''(a, \bot_1', \bot_2')$ and $R''(a, \bot_2', \bot_1')$. But this implies that T contains $R'(a, \bot_1', \bot')$ and $R'(a, \bot_2', \bot')$, and therefore, T is isomorphic to T_2. In particular, T_2 is no homomorphic image of T_1.

Finally, for $n \geq 1$ consider the source instance $S_n = \{R(i, b), R(i, c) \mid 1 \leq i \leq n\}$. Then for every set $\mathcal{I} \subseteq \{1, \ldots, n\}$, the target instance

$$\begin{aligned} T_{\mathcal{I}} \;:=\; &\bigcup_{i \in \mathcal{I}} \quad \big\{R'(i, \bot_1^i, \bot_2^i), R'(i, \bot_3^i, \bot_4^i), R''(i, \bot_1^i, \bot_1^i), R''(i, \bot_3^i, \bot_3^i)\big\}\\ \cup &\bigcup_{i \in \{1, \ldots, n\} \setminus \mathcal{I}} \Big(\big\{R'(i, \bot_1^i, \bot_2^i), R'(i, \bot_3^i, \bot_2^i)\big\} \cup \big\{R''(i, \bot_k^i, \bot_l^i) \mid k, l \in \{1, 3\}\big\}\Big)\end{aligned}$$

is a CWA-solution for S_n under D'. Arguing as above, it is not hard to see that for every CWA-solution T for S_n under D' there is exactly one $\mathcal{I} \subseteq \{1, \ldots, n\}$ such that T is a homomorphic image of $T_{\mathcal{I}}$. Furthermore, for distinct $\mathcal{I}, \mathcal{I}' \subseteq \{1, \ldots, n\}$, $T_{\mathcal{I}}$ is no homomorphic image of $T_{\mathcal{I}'}$. Thus, there are 2^n "maximal" CWA-solutions for S_n under D'. □

B. REMARK ON COMPUTING CWA-SOLUTIONS VIA THE CHASE

The example below exhibits a weakly acyclic data exchange setting $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ with Σ_{t} containing egds, and a source instance S for D such that there is a successful chase sequence of S with $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ whose result is $S \cup T$, but T is no CWA-solution for S under D.

Example B.1. Consider the data exchange setting $D = (\{P\}, \{E, F\}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, where P is unary, E, F are binary, and $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ consists of

$$\begin{aligned} P(x) &\to \exists z_1 \exists z_2 \big(E(x, z_1) \wedge E(x, z_2)\big) \\ E(y, x) &\to \exists z\, F(x, z) \\ E(y, x) \wedge E(y, x') &\to x = x'. \end{aligned}$$

Furthermore, consider the source instance $S := \{P(a)\}$. Then the instance $S \cup T$, where

$$T \ := \ \{E(a, \bot_1), F(\bot_1, \bot_2), F(\bot_1, \bot_3)\},$$

is the result of some successful chase sequence of S with $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$: first apply the first tgd, then apply the second tgd twice, and finally, apply the egd. But T is no CWA-presolution for S under D, and therefore, T is no CWA-solution for S under D. □

C. UNDECIDABILITY OF FO QUERY ANSWERING UNDER WEAKLY ACYCLIC DATA EXCHANGE SETTINGS AND SEMANTICS CERTAIN$_\Box$/MAYBE$_\Diamond$

The goal of this section is to prove Proposition 6.1 of the main article:

PROPOSITION 6.1. *There are a weakly acyclic data exchange setting D and a Boolean FO query Q over D's target schema such that* $\text{EVAL}_{\mathsf{certain}_\Box}(D, \neg Q)$ *and* $\text{EVAL}_{\mathsf{maybe}_\Diamond}(D, Q)$ *are undecidable.*

PROOF. Consider the data exchange setting $D = (\{R\}, \{\tilde{R}\}, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, where R and $\tilde{R}$ are ternary relation symbols, Σ_{st} consists of the tgd $R(x, y, z) \to \tilde{R}(x, y, z)$, and Σ_{t} consists of the tgd $\tilde{R}(x, y, z) \to \exists x' \exists y' \exists z'\, \tilde{R}(x', y', z')$. Let Q be the conjunction of the target dependencies (4.8)–(4.10) of the data exchange setting D_{emb} introduced in Section 4.2. That is,

$$\begin{aligned} Q \ = \ & \forall x \forall y \forall z \forall z' \big(\tilde{R}(x, y, z) \wedge \tilde{R}(x, y, z') \to z = z'\big) \\ & \wedge \forall x \forall y \forall z \forall u \forall v \forall w \big(\tilde{R}(x, y, u) \wedge \tilde{R}(y, z, v) \wedge \tilde{R}(x, v, w) \to \tilde{R}(u, z, w)\big) \\ & \wedge \forall x_1 \forall x_2 \forall x_3 \forall y_1 \forall y_2 \forall y_3 \big(\tilde{R}(x_1, x_2, x_3) \wedge \tilde{R}(y_1, y_2, y_3) \to \bigwedge_{1 \le i, j \le 3} \exists z \tilde{R}(x_i, y_j, z)\big). \end{aligned}$$

Note that Q is true in a target instance T for D if and only if $\tilde{R}^T$ encodes the graph of an associative (total) function. We show that $\text{EVAL}_{\mathsf{maybe}_\Diamond}(D, Q)$ is undecidable. By duality (i.e., $\mathsf{maybe}^D_\Diamond(Q, S) \neq \emptyset$ if and only if $\mathsf{certain}^D_\Box(\neg Q, S) = \emptyset$), this implies that $\text{EVAL}_{\mathsf{certain}_\Box}(D, \neg Q)$ is undecidable.

To show that $\text{EVAL}_{\mathsf{maybe}_\Diamond}(D, Q)$ is undecidable, we give a reduction from the undecidable embedding problem for finite semigroups (see Section 4.2) to the problem $\text{EVAL}_{\mathsf{maybe}_\Diamond}(D, Q)$: given an associative partial function p, we map p to the source instance $S_p = \{R(x, y, z) \mid p(x, y) = z\}$ and the empty tuple. It remains to show that $\mathsf{maybe}^D_\Diamond(Q, S_p) \neq \emptyset$ exactly if p is a "yes"-instance for the embedding problem for finite semigroups.

If $\mathsf{maybe}^D_\Diamond(Q, S_p) \neq \emptyset$, then there is a CWA-solution T for S_p under D and $\hat{T} \in \boldsymbol{Rep}_{\Sigma_{\text{t}}}(T)$ with $\hat{T} \models Q$. So, $\tilde{R}^{\hat{T}}$ represents the graph of a finite total associative function f that extends p. Consequently, p is a "yes"-instance for the embedding problem for finite semigroups.

On the other hand, suppose that p is a "yes"-instance for the embedding problem for finite semigroups, that is, there is a finite total associative function $f \colon B^2 \to B$ that extends p. Let $(a_1^1, a_2^1, a_3^1), \dots, (a_1^k, a_2^k, a_3^k)$ be an enumeration of all tuples $(a_1, a_2, a_3) \in B^3$ such that $f(a_1, a_2) = a_3$ and $p(a_1, a_2)$ is undefined. Moreover, let $\bot_1^1, \bot_2^1, \bot_3^1, \bot_1^2, \dots, \bot_3^k$

be a sequence of pairwise distinct nulls. Then the instance T with $\tilde{R}^T = R^{S_p} \cup \{(\perp_1^i, \perp_2^i, \perp_3^i) \mid 1 \leq i \leq k\}$ is a CWA-solution for S_p under D: $(\perp_1^1, \perp_2^1, \perp_3^1)$ can be generated from an arbitrary tuple in R^{S_p}, and $(\perp_1^{i+1}, \perp_2^{i+1}, \perp_3^{i+1})$ can be generated from $(\perp_1^i, \perp_2^i, \perp_3^i)$. Finally, consider the instance $\hat{T} = v(T)$, where v maps each $\perp_j^i$ to a_j^i. Then $\hat{T} \models Q$, which implies that $\mathsf{maybe}_\diamond^D(Q, S_p) \neq \emptyset$. □

D. PROOF OF PROPOSITION 6.12

Here we prove Proposition 6.12 from the main body of this paper:

PROPOSITION 6.12. *Let* answer $\in \{\mathsf{certain}_\Box, \mathsf{certain}_\diamond\}$. *Then the data complexity of unions of conjunctive queries with respect to* answer *and the class of weakly acyclic data exchange settings is* PTIME-*hard.*

PROOF. The proof is a slight modification of the proof of [Kolaitis et al. 2006, Proposition 3.1], which shows that there is a data exchange setting D such that, given a source instance for D, it is PTIME-hard to decide whether there is a solution for S under D.

Let $D = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be this setting. Here, σ consists of two ternary relation symbols P, N, and a unary relation symbol V, and τ consists of two ternary relation symbols P', N', and three unary relation symbols V', M', W'. The tgds in Σ_{st} simply copy the source relations to the corresponding primed target relations. Finally, Σ_{t} consists of the following target dependencies:

$$\begin{aligned} d_1 &= W'(u) \wedge W'(v) \rightarrow u = v \\ d_2 &= P'(x, x, x) \rightarrow M'(x) \\ d_3 &= P'(x, y, z) \wedge M'(y) \wedge M'(z) \rightarrow M'(x) \\ d_4 &= N'(x, y, z) \wedge M'(x) \wedge M'(y) \wedge M'(z) \wedge V'(u) \rightarrow W'(u). \end{aligned}$$

Now let $D' := (\sigma, \tau, \Sigma_{\text{st}}, \{d_2, d_3\})$, and let

$$Q := \exists x \exists y \exists z \big(N'(x, y, z) \wedge M'(x) \wedge M'(y) \wedge M'(z)\big).$$

Then, given a source instance S for D (resp. D'), it is easy to see that there is a solution for S under D if and only if $\mathsf{answer}^{D'}(Q, S) = \emptyset$. □