Answering Non-Monotonic Queries in Relational Data Exchange

André Hernich
Goethe-Universität, Frankfurt am Main, Germany
hernich@informatik.uni-frankfurt.de

ABSTRACT

Relational data exchange deals with translating a relational database instance over some *source schema* into a relational database instance over some *target schema*, according to a *schema mapping* that specifies the relationship between the source data and the target data. Various semantics for answering queries against the target schema exist, each of them suitable for a certain class of queries, and with respect to certain schema mappings. However, for each of these semantics, there are examples that show that it leads to counter-intuitive answers, or that it does not respect logical equivalence of schema mappings.

In this article, we study query answering semantics for deductive databases in the context of relational data exchange. Furthermore, we propose a new semantics, called *GCWA*-answers semantics*, which seems to be well-suited with respect to a number of schema mappings, including schema mappings defined by st-tgds and egds. We show that the GCWA*-answers semantics coincides with the classical certain answers semantics on monotonic queries, and we further explore the data complexity of computing the GCWA*-answers to non-monotonic queries. In particular, we identify a class of schema mappings for which the GCWA*-answers to universal queries can be computed from the core of the universal solutions in polynomial time (data complexity).

Categories and Subject Descriptors

H.2.5 [**Heterogeneous Databases**]: Data translation; H.2.4 [**Systems**]: Relational databases, Rule-based databases, Query processing; D.2.12 [**Interoperability**]: Data mapping

General Terms

Algorithms, Languages, Theory

Keywords

closed world assumption, deductive database, certain answers, core

ICDT 2010, March 22–25, 2010, Lausanne, Switzerland.

1. INTRODUCTION

Data exchange deals with translating data structured in some "old" format into data structured in some "new" format, according to a specification of the relationship between the source data and the target data. Here we consider the case of *relational* data, that is, the source data and the target data are relational. Formally, we are given a *schema mapping* $M = (\sigma, \tau, \Sigma)$ (where σ and τ are schemas, called the *source schema* and the *target schema*, respectively, and Σ is a finite set of logical constraints over σ and τ) and an instance S over σ (called *source instance* for M), and the task is to compute a *solution* for S under M, that is, an instance over τ such that all the constraints in Σ are satisfied. If a solution exists at all, one would like to compute a solution that reflects the source instance as good as possible.

The theoretical foundations of data exchange were laid in two seminal articles by Fagin, Kolaitis, Miller, and Popa [10, 11]. These articles introduced the notion of *universal solutions* (solutions, that are at most as general as any other solution) that possess properties that make them desirable for data exchange, and in particular the *core* of the universal solutions which, for schema mappings specified by tgds and egds, is the smallest universal solution (up to isomorphism). The complexity of computing solutions, and the core of the universal solutions have been investigated [10, 11, 22, 16], and extensions of the "classical" data exchange framework to XML data [5], peer data exchange [13], and model management [12, 9] have been considered. See also the survey articles [21, 6, 18].

An important issue in data exchange is how to answer queries against the target schema (see, e.g., [10, 11, 21, 6, 2]). In [10], the *certain answers* have been proposed to answer queries over the target schema. Given a schema mapping $M = (\sigma, \tau, \Sigma)$, a source instance S for M, and a query q over τ, the set of the *certain answers to q on M and S* consists of all tuples $\bar{t}$ such that $\bar{t}$ is an answer to q on every solution T for S under M. While the certain answers semantics has been widely adopted as the "right" semantics to answer queries preserved under homomorphisms like unions of conjunctive queries, Datalog queries, or the Datalog$^{\mathbf{C}(\neq)}$ queries in [4], it was noted that it may lead to counter-intuitive answers for queries with negations [10, 3, 23]. It was also noted in [3, 23] that the same problems occur for the universal solution-based semantics of [11]. Some of these problems were solved by the CWA-solution-based semantics in [23, 19], and the mixed world-based semantics in [24]. These semantics are defined in terms of new solution concepts (CWA-solutions in [23, 19], and mixed world solutions

in [24]) based on certain conditions that restrict the way how nulls can enter solutions, and using techniques from [20] (see also [28]) to answer queries on individual solutions. Each of these semantics is good for a particular fragment of FO queries with respect to certain schema mappings. However, there are queries on which both semantics lead to answers that intuitively do not seem to be "right". Furthermore, they do not respect logical equivalence of schema mappings. Here, schema mappings $M_1 = (\sigma, \tau, \Sigma_1)$ and $M_2 = (\sigma, \tau, \Sigma_2)$ are logically equivalent if and only if Σ_1 and Σ_2 are logically equivalent, that is, each model of Σ_1 is a model of Σ_2 and vice versa. Since schema mappings are defined by logical formulas, it is desirable that the answer to a query is the same on logically equivalent schema mappings.

The concept of CWA-solutions from [23, 19] is based on the *closed world assumption (CWA)*, originally introduced by Reiter [27] for answering non-monotonic queries on *deductive databases*. A deductive database is a set of *clauses*, where each clause is a universally quantified disjunction of relational atomic formulas and negated relational atomic formulas [14]. A *model* of a deductive database D is an instance without nulls that satisfies all clauses in D, and a query q is usually answered by determining all tuples that are answers to q on each model in a particular subset of the set of all models of D (the subset depends on the semantics). Reiter's CWA defines a particular subset of the set of all models of D, and the corresponding query answering semantics takes only the models in this subset into account. There is a rich literature on semantics for answering non-monotonic queries on deductive databases. The definitions of many of these semantics apply as well, or can be extended, to more general sets of logical sentences, such as sets $\Sigma \cup \{R(\bar{t}) \mid R \in \sigma,\ \bar{t} \in R^S\} \cup \{\neg R(\bar{t}) \mid R \in \sigma,\ \bar{t} \in Const^{\mathrm{ar}(R)} \setminus R^S\}$, where Σ is the set of constraints of a schema mapping $M = (\sigma, \tau, \Sigma)$, and S is a source instance for M. This makes the area of deductive databases attractive for data exchange.

In this article, we translate various semantics for answering non-monotonic queries that originally have been developed for deductive databases, such as the semantics based on Reiter's *CWA* [27], on the *generalized CWA (GCWA)* [26] or on the *extended GCWA (EGCWA)* [29], and the *possible worlds semantics (PWS)* [7], into the framework of data exchange. Furthermore, we study these semantics with respect to the following question: which of these semantics is appropriate for answering non-monotonic queries in data exchange? Unfortunately, none of these semantics seems to be appropriate. The semantics based on Reiter's CWA leads to inconsistent answers already on very simple schema mappings, the GCWA-based semantics seems to be too weak, the EGCWA-based semantics seems to be too strong, and the PWS does not respect logical equivalence of schema mappings.

Inspired by the GCWA-based semantics, we develop the *GCWA*-answers semantics*, which is based on a new concept of solutions, called *GCWA*-solutions*, and which seems to be well-suited for answering non-monotonic queries with respect to a number of schema mappings, including schema mappings defined by st-tgds and egds. GCWA*-solutions are solutions that are unions of inclusion-minimal solutions without nulls. This property makes it particularly easy to work with GCWA*-solutions. The GCWA*-answers to a query are defined as the certain answers with respect to GCWA*-solutions. For a number of schema mappings, including schema mappings defined by st-tgds and egds, we argue that, intuitively, the set of all GCWA*-solutions for a source instance S under such a schema mapping M precisely reflects the information contained in M and S.

It should be emphasized that GCWA*-solutions are solutions without nulls, so that the GCWA*-answers semantics takes into account only solutions without nulls. This is no loss of generality, since, actually, solutions with nulls serve as representations for sets of solutions without nulls [20], so that a query answering semantics should not depend on solutions with nulls. In fact, it can be verified that most of the known certain answers-based query answering semantics in data exchange can be defined without referring to solutions with nulls. One of the advantages of this approach is that we do not run into any trouble that might result from dealing with nulls (they can quickly lead to anomalies, cf., [23]). However, solutions with nulls are still important since, as mentioned above, they serve as finite representations of possibly infinite sets of solutions without nulls.

We study the data complexity of computing the GCWA*-answers to a query. Here, data complexity means that the schema mapping and the query are *fixed* (i.e., they are not part of the input). The main result, and the technically most challenging contribution, of this article is Theorem 6.6. This theorem states that for certain schema mappings M defined by st-tgds, and for all universal queries q (FO queries of the form $\forall \bar{x} \varphi$, where φ is quantifier-free), there is a polynomial time algorithm that, given the core of the universal solutions for some source instance S under M, computes the GCWA*-answers to q on M and S. This also implies that there is a polynomial time algorithm that, given a source instance S for M, outputs the GCWA*-answers to q on M and S. For arbitrary schema mappings defined by st-tgds, the data complexity of computing the GCWA*-answers to universal queries is in co-NP.

Finally, we show that the GCWA*-answers semantics coincides with the certain answers semantics on monotonic queries (Proposition 6.1). Therefore, all tools and techniques developed for computing the certain answers to a query (see, e.g., [10, 25, 3, 21, 23, 8, 4, 6]) can be used to compute the GCWA*-answers to monotonic queries. Furthermore, we show that there is a schema mapping M defined by two LAV tgds, and an existential query q of the form $\exists \bar{x} \varphi$, where φ is a conjunction of relational atomic formulas and a single negated relational atomic formula, such that the problem $\textsc{Eval}(M, q)$ (given a source instance S for M and a tuple $\bar{t}$, is $\bar{t}$ a GCWA*-answer to q on M and S) is co-NP-complete (Proposition 6.2). On the other hand, if we allow existential quantifiers and just one universal quantifier, then $\textsc{Eval}(M, q)$ may be undecidable (Proposition 6.3).

This article is organized as follows. In Section 2, we fix basic definitions and mention basic results that are used throughout this article. Section 3 explains the problems with known query answering semantics in data exchange in more detail. In Section 4, we then study the query answering semantics for deductive databases in the context of data exchange. The new GCWA*-answers semantics is introduced and illustrated in Section 5, and the data complexity of answering queries using the GCWA*-answers semantics is investigated in Section 6. Most of the proofs are deferred to the full version of this article.

2. PRELIMINARIES

A *schema* σ is a finite set of relation symbols R, each with an associated arity $\mathrm{ar}(R)$. An *instance* I over σ assigns to each $R \in \sigma$ a finite relation R^I of arity $\mathrm{ar}(R)$. $\mathrm{dom}(I)$ denotes the set of all elements that occur in the tuples of the relations R^I, for each $R \in \sigma$. We assume a fixed infinite set *Dom*, whose elements are called *values*, such that $\mathrm{dom}(I) \subseteq Dom$ for all instances I. An *atom* is an expression of the form $R(\bar{t})$, where R is a relation symbol, and $\bar{t} \in Dom^{\mathrm{ar}(R)}$. We often identify an instance I with the set of all atoms $R(\bar{t})$ with $\bar{t} \in R^I$, that is, $I = \{R(\bar{t}) \mid R \in \sigma, \bar{t} \in R^I\}$. An instance I is $\subseteq$-*minimal* in a set $\mathcal{I}$ of instances if $I \in \mathcal{I}$, and there is no $I' \in \mathcal{I}$ with $I' \subsetneq I$.

As usual in data exchange, *Dom* is the union of two disjoint infinite sets *Const* and *Null*, where the elements in *Const* are called *constants*, and the elements in *Null* are called *(labelled) nulls*. Nulls are just placeholders, or variables, for constants. Throughout this article, lower case letters $a, b, c, \ldots$ from the start of the alphabet denote constants, and the symbol $\perp$, possibly with sub/superscripts, denotes nulls. For an instance I, let $\mathrm{const}(I) := \mathrm{dom}(I) \cap Const$ and $\mathrm{nulls}(I) := \mathrm{dom}(I) \cap Null$. An instance is called *ground* if it contains no nulls, and an atom $R(\bar{t})$ is called ground if $\bar{t}$ contains no nulls.

Let $f\colon X \to Y$, where X and Y are arbitrary sets. For a tuple $\bar{t} = (t_1, \ldots, t_n) \in X^n$, we let $f(\bar{t}) := (f(t_1), \ldots, f(t_n))$; for an atom $A = R(\bar{t})$, we let $f(A) := R(f(\bar{t}))$; and for an instance I, we let $f(I) := \{f(A) \mid A \in I\}$. f is called *legal* for an instance I if and only if $\mathrm{dom}(I) \subseteq X$, and $f(c) = c$ for all $c \in \mathrm{const}(I)$. The set of all functions that are legal for I is denoted by *legal*(I). For a tuple $\bar{t} = (t_1, \ldots, t_k)$, we sloppily write $f\colon \bar{t} \to Y$ for a function $f\colon X \to Y$ with $X = \{t_1, \ldots, t_k\}$. Given $\bar{t} = (t_1, \ldots, t_k)$, $\bar{u} = (u_1, \ldots, u_l)$, and an element v, we also write $v \in \bar{t}$ if $v \in \{t_1, \ldots, t_k\}$, and we let $\bar{t} \cap \bar{u}$ be the intersection of $\{t_1, \ldots, t_k\}$ and $\{u_1, \ldots, u_l\}$.

Let I and J be instances. A *homomorphism* from I to J is a function $h\colon \mathrm{dom}(I) \to \mathrm{dom}(J)$ such that $h \in$ *legal*(I) and $h(I) \subseteq J$. If $h(I) = J$, then J is called a *homomorphic image* of I. I and J are *homomorphically equivalent* if and only if there is a homomorphism from I to J, and a homomorphism from J to I. An *isomorphism* from I to J is a homomorphism h from I to J such that h is bijective, and h^{-1} is a homomorphism from J to I. We say that I and J are *isomorphic*, and write $I \cong J$, if and only if there is an isomorphism from I to J. A *core* of I is a subinstance C of I such that there is a homomorphism from I to C, but there is no homomorphism from I to any proper subinstance of C. It is known [17] that if I and J are homomorphically equivalent, C is a core of I, and C' is a core of J, then $C \cong C'$. In particular, any two cores of I are isomorphic, so that we can speak of *the* core of I, which is denoted by $\mathrm{Core}(I)$.

2.1 Queries

As usual [1], a k-ary query over a schema σ is a mapping from instances over σ to Dom^k that is C-generic for some finite set $C \subseteq Const$ (i.e., the query is invariant under renamings of values in $Dom \setminus C$). In the context of queries defined by logical formulas, we will often use the words *formula* and *query* as synonyms. Whenever we speak of a first-order (FO) formula over a schema σ, we mean a FO formula over the vocabulary that consists of all relation symbols in σ, and all constants in *Const*. Each constant in *Const* is interpreted by itself. A $L_{\infty\omega}$ formula over a schema σ is built from atomic FO formulas over σ, negations, existential quantifications, universal quantifications, *infinitary disjunctions* $\bigvee \Phi$, where Φ is an arbitrary set of $L_{\infty\omega}$ formulas over σ, and *infinitary conjunctions* $\bigwedge \Phi$, where Φ is an arbitrary set of $L_{\infty\omega}$ formulas over σ. The semantics of infinitary disjunctions and infinitary conjunctions is the obvious one: for an assignment α of the variables that occur in the formulas in Φ we have $I \models \bigvee \Phi(\alpha)$ if and only if there is some $\varphi \in \Phi$ with $I \models \varphi(\alpha)$, and $I \models \bigwedge \Phi(\alpha)$ if and only if for all $\varphi \in \Phi$, $I \models \varphi(\alpha)$.

Queries are evaluated on an instance using the active domain semantics [1] (i.e., when evaluating a query q on I, then the quantifiers in q range over the constants in I and q). A query $q(\bar{x})$ over a schema σ is *monotonic* if $q(I) \subseteq q(J)$ for all instances I, J over σ with $I \subseteq J$. It is easy to see that all queries preserved under homomorphisms are monotonic. Here, a query $q(\bar{x})$ over σ is *preserved under homomorphisms* if and only if for all instances I, J over σ, all homomorphisms h from I to J, and all tuples $\bar{t} \in q(I)$, we have $h(\bar{t}) \in q(J)$. For example, conjunctive queries, unions of conjunctive queries, Datalog queries, and the Datalog$^{\mathbf{C}(\neq)}$ queries of [4] are preserved under homomorphisms. *Unions of conjunctive queries with inequalities* (see, e.g., [10] for a definition) are an example of monotonic queries that are not preserved under homomorphisms.

2.2 Data exchange

A *schema mapping* is a triple $M = (\sigma, \tau, \Sigma)$, where σ and τ are disjoint schemas, called the *source schema* and the *target schema*, and Σ is a finite set of constraints in some logical formalism over $\sigma \cup \tau$ [10]. To introduce and to study query answering semantics in a general setting, we assume that for all schema mappings $M = (\sigma, \tau, \Sigma)$ considered in this article, Σ consists of $L_{\infty\omega}$ sentences over $\sigma \cup \tau$ (that are C-generic for some finite $C \subseteq Const$). For *algorithmic results*, however, we restrict attention to schema mappings $M = (\sigma, \tau, \Sigma)$, where Σ consists of *source-to-target tuple generating dependencies (st-tgds)* and *equality generating dependencies (egds)*, which have been prominently considered in data exchange. Here, an *st-tgd* is a FO sentence of the form

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

where φ is a conjunction of relational atomic FO formulas over σ with free variables $\bar{x}\bar{y}$, and ψ is a conjunction of relational atomic FO formulas over τ with free variables $\bar{x}\bar{z}$. A *full st-tgd* is a st-tgd without existentially quantified variables $\bar{z}$, and a *LAV tgd* is a st-tgd with a single atomic formula in φ. An *egd* is a FO sentence of the form

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

where φ is a conjunction of relational atomic FO formulas over τ with free variables $\bar{x}$, and x_i, x_j are variables in $\bar{x}$.

Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping. A *source instance* for M is a *ground* instance over σ, and a *target instance* for M is an instance over τ. Given a source instance S for M, a *solution* for S under M is a target instance T for M such that $S \cup T \models \Sigma$, that is, $S \cup T$ satisfies all the constraints in Σ.

A *universal solution* for S under M is a solution T for S under M such that for all solutions T' for S under M there is a homomorphism from T to T'. Note that all universal solutions for S under M are homomorphically equivalent. In particular, there is a target instance C that is isomorphic

to the cores of all universal solutions for S under M. We denote this instance by $\mathrm{Core}(M,S)$. In a number of cases, $\mathrm{Core}(M,S)$ is a solution for S under M. For example, if Σ contains only st-tgds, then $\mathrm{Core}(M,S)$ is a solution for S under M [11], which can be computed in polynomial time:

THEOREM 2.1 ([11]). *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of st-tgds. Then there is a polynomial time algorithm that, given a source instance S for M, outputs $\mathrm{Core}(M,S)$.*

Besides $\mathrm{Core}(M,S)$, the *canonical universal solution* for S under M, denoted by $\mathrm{CanSol}(M,S)$, plays an important role in data exchange. In the following, we give the definition of $\mathrm{CanSol}(M,S)$ from [3] for the case that Σ contains only st-tgds. Let $\mathcal{J}$ be the set of all triples $(\theta, \bar{a}, \bar{b})$ such that θ is a st-tgd in Σ of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, and $S \models \varphi(\bar{a},\bar{b})$. Starting from an empty target instance for M, $\mathrm{CanSol}(M,S)$ is created by adding atoms for each element in $\mathcal{J}$ as follows. For each $j = (\theta, \bar{a}, \bar{b}) \in \mathcal{J}$, where $\theta = \forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, let $\bar{\perp}_j$ be a $|\bar{z}|$-tuple of pairwise distinct nulls such that for all $j' \in \mathcal{J}$ with $j' \neq j$, the set of nulls in $\bar{\perp}_j$ is disjoint from the set of nulls in $\bar{\perp}_{j'}$, and add the atoms in $\psi(\bar{a}, \bar{\perp}_j)$ to the target instance.

2.3 Instances with incomplete information

Instances that may contain nulls are also called *naive tables* [1], or *V-tables* [20]. We call an instance naive table to emphasize that it may contain nulls. Naive tables serve as finite representations of certain incomplete instances, where an *incomplete instance* $\mathcal{I}$ over a schema σ is a set of ground instances over σ [1]. The instances in $\mathcal{I}$ are called the *possible worlds* of $\mathcal{I}$.

A naive table T contains incomplete information in the sense that each null $\perp$ in T represents an *unknown* (rather than nonexistent) constant. The *possible worlds* of (the incomplete instance represented by) T are all instances obtained from T by replacing each null by a constant. Formally, let a *valuation* of T be a function $v\colon \mathrm{dom}(T) \rightarrow \mathit{Const}$ with $v \in \mathit{legal}(T)$. Then the set of all possible worlds of T is

$$\mathit{poss}(T) \;:=\; \{v(T) \mid v \text{ is a valuation of } T\}.$$

Given an incomplete instance $\mathcal{I}$ over a schema σ and a query $q(\bar{x})$ over σ, there are several ways to evaluate q on $\mathcal{I}$. A common way is to take the *certain answers to q on $\mathcal{I}$*, which are defined by

$$\mathit{cert}(q,\mathcal{I}) \;:=\; \bigcap\{q(I) \mid I \in \mathcal{I}\}.$$

3. QUERY ANSWERING SEMANTICS IN DATA EXCHANGE

As already mentioned in the introduction, the certain answers semantics from [10] has been widely adopted as the "right" semantics for answering queries preserved under homomorphisms in data exchange. We also mentioned that it can lead to counter-intuitive answers on non-monotonic queries, and that the same problems occur for the universal solution-based semantics from [11] (see, e.g., [10, 3, 23, 19]). For example, consider the schema mapping $M^* = (\{E\}, \{E'\}, \Sigma)$, where Σ consists of the st-tgd

$$\forall x_1\forall x_2(E(x_1,x_2) \rightarrow E'(x_1,x_2))$$

that just tells us to "copy" E to E'. Thus, given the source instance S^* for M^* with $E^{S^*} = \{(a,a)\}$, we intuitively expect that the set of answers to the query

$$q^*(x) \;:=\; \forall y(E'(x,y) \rightarrow E'(y,x))$$

is simply the set of answers to q^* on the "copy" T_{S^*} of S^* over $\{E'\}$ with $(E')^{T_{S^*}} = \{(a,a)\}$, that is, $\{a\}$. However, contrary to this expectation, the certain answers to q^* on M^* and S^* are empty.

Some of these problems were solved by the CWA-solution-based semantics in [23, 19]. The notion of CWA-solutions has been introduced for the case of schema mappings defined by st-tgds [23], and was extended to the more general case of schema mappings defined by st-tgds, t-tgds and egds in [19] (see, e.g., [10] for a definition of t-tgds). Here we consider the case of schema mappings defined by st-tgds. Given a schema mapping $M = (\sigma, \tau, \Sigma)$, where Σ consists of st-tgds, and a source instance S for M, a *CWA-solution* for S under M can be characterized as a universal solution T for S under M that is a homomorphic image of $\mathrm{CanSol}(M,S)$ [23]. Now a query q over τ can be answered by

$$\mathit{cert}_{\mathrm{CWA}}(q,M,S) \;:=\; \mathit{cert}(q,\mathcal{I}),$$

where $\mathcal{I}$ is the set of all target instances $\hat{T}$ for M for which there is a CWA-solution T for S under M with $\hat{T} \in \mathit{poss}(T)$. This yields the certain answers semantics from [23].

One particular setting, where the CWA-solution-based semantics leads to "good" answers is the setting of schema mappings defined by *full* st-tgds. The following example shows that, in general, $\mathit{cert}_{\mathrm{CWA}}$ can lead to answers that are inconsistent with M and S:

EXAMPLE 3.1. Let $M = (\{E\}, \{F,G\}, \Sigma)$ be the schema mapping, where E, F, G are binary relation symbols, and Σ consists of the st-tgd

$$\theta \;:=\; \forall x_1\forall x_2\big(E(x_1,x_2) \rightarrow \exists z(F(x_1,z) \wedge G(z,x_2))\big).$$

Let S be the source instance for M with $E^S = \{(a,b)\}$. Then $T = \mathrm{CanSol}(M,S)$ with $F^T = \{(a,\perp)\}$ and $G^T = \{(\perp,b)\}$ is the unique CWA-solution for S under M, up to isomorphism. For the query

$$q(x) \;:=\; \exists z\big(F(x,z) \wedge \forall z'(F(x,z') \rightarrow z' = z)\big),$$

it follows therefore that $\mathit{cert}_{\mathrm{CWA}}(q,M,S) = \{a\}$. That is, $\mathit{cert}_{\mathrm{CWA}}(q,M,S)$ excludes the possibility that there is more than one value z with $F(a,z)$. However, this is inconsistent with θ and S, which, taking the usual semantics of existential quantification, tell us that there are *one or more* z with $F(a,z)$ and $G(z,b)$. In particular, they explicitly state that it is possible that there is more than one z with $F(a,z)$.

The example can be modified to show that the mixed world-based semantics in [24] can lead to answers that are inconsistent with M and S.

The next example shows that $\mathit{cert}_{\mathrm{CWA}}$ does not respect logical equivalence of schema mappings. That is, there are logically equivalent schema mappings $M_1 = (\sigma, \tau, \Sigma_1)$ and $M_2 = (\sigma, \tau, \Sigma_2)$, an instance S over σ and a query q over τ such that $\mathit{cert}_{\mathrm{CWA}}(q,M_1,S) \neq \mathit{cert}_{\mathrm{CWA}}(q,M_2,S)$:

EXAMPLE 3.2. Let $M_1 = (\sigma, \tau, \Sigma_1)$ and $M_2 = (\sigma, \tau, \Sigma_2)$ be schema mappings, where σ contains a unary relation symbol P, τ contains a binary relation symbol E, and

$$\begin{aligned}\Sigma_1 \;&:=\; \{\forall x(P(x) \rightarrow E(x,x))\},\\ \Sigma_2 \;&:=\; \Sigma_1 \cup \{\forall x(P(x) \rightarrow \exists z E(x,z))\}.\end{aligned}$$

Then M_1 and M_2 are logically equivalent.

Let S be an instance over σ with $P^S = \{a\}$. Furthermore, let T_1 and T_2 be instances over τ with $E^{T_1} = \{(a, a)\}$ and $E^{T_2} = \{(a, a), (a, \bot)\}$. Note that T_1 is the unique CWA-solution for S under M_1, and that T_2 is a CWA-solution for S under M_2. Thus, for the query

$$q(x) \ := \ \exists z\big(E(x, z) \wedge \forall z'(E(x, z') \rightarrow z' = z)\big),$$

we obtain different answers $cert_{\mathrm{CWA}}(q, M_1, S) = \{a\}$ and $cert_{\mathrm{CWA}}(q, M_2, S) = \emptyset$ to the same query q on logically equivalent schema mappings M_1 and M_2.

Using Example 3.2, one can show that the mixed world-based semantics in [24] do not respect logical equivalence of schema mappings either.

4. DEDUCTIVE DATABASES AND DATA EXCHANGE

In this section, we study query answering semantics for deductive databases in the framework of data exchange. In particular, we focus on the question of which of these semantics leads to an appropriate semantics for answering non-monotonic queries in data exchange.

A *deductive database* [14] over a schema σ is a set of FO sentences, called *clauses*, of the form

$$\forall \bar{x}\big(\neg R_1(\bar{y}_1) \vee \cdots \vee \neg R_m(\bar{y}_m) \vee R'_1(\bar{z}_1) \vee \cdots \vee R'_n(\bar{z}_n)\big),$$

where m and n are nonnegative integers with $m + n \geq 1$, $R_1, \ldots, R_m, R'_1, \ldots, R'_n$ are relation symbols in σ, and $\bar{y}_1, \ldots, \bar{y}_m, \bar{z}_1, \ldots, \bar{z}_n$ are tuples containing elements of *Const* and $\bar{x}$. A *model* of a deductive database D over σ is a *ground* instance I over σ with $I \models D$ (i.e., I satisfies all clauses in D), and a query q is usually answered by $cert(q, \mathcal{I})$, where $\mathcal{I}$ is a set of models of D that depends on the particular query answering semantics.

Various semantics for answering non-monotonic queries on deductive databases exist. These semantics are based on certain assumptions on what negative data can be inferred from a deductive database. For the purposes of data exchange, the most interesting of these assumptions seem to be the *closed world assumption (CWA)* [27], the *generalized closed world assumption (GCWA)* [26], the *extended generalized closed world assumption (EGCWA)* [29], and also the assumption underlying the *possible worlds semantics (PWS)* [7]. What makes these semantics interesting for data exchange is that the definitions of these semantics apply as well, or can be extended from sets of clauses, to more general sets of logical sentences, such as sets

$$\begin{aligned} D_{M,S} \ := \ & \Sigma \cup \{R(\bar{t}) \mid R \in \sigma, \bar{t} \in R^S\} \\ & \cup \{\neg R(\bar{t}) \mid R \in \sigma, \bar{t} \in Const^{\mathrm{ar}(R)} \setminus R^S\}, \end{aligned}$$

where Σ is the set of constraints of a schema mapping $M = (\sigma, \tau, \Sigma)$, and S is a source instance for M. Note that, if Σ consists only of full st-tgds, then $D_{M,S}$ is logically equivalent to a deductive database, since any full st-tgd of the form $\forall \bar{x}(R_1(\bar{y}_1) \wedge \cdots \wedge R_m(\bar{y}_m) \rightarrow R'(\bar{z}))$ is logically equivalent to the clause $\forall \bar{x}(\neg R_1(\bar{y}_1) \vee \cdots \vee \neg R_m(\bar{y}_m) \vee R'(\bar{z}))$.

In the following, we study the CWA, the GCWA, the EGCWA and the PWS in more detail in the context of data exchange.

4.1 The Closed World Assumption (CWA)

The *closed world assumption (CWA)*, first formalized by Reiter [27], assumes that every ground atom that is not implied by a database is false. This is a common assumption for relational databases.

Reiter formalized the CWA, and defined a query answering semantics for deductive databases based on the CWA, as follows. For a deductive database D and a formula φ, we write $D \models \varphi$ if and only if for all instances I with $I \models D$, we have $I \models \varphi$. Given a deductive database D over a schema σ, Reiter defines the set

$$\overline{D} \ := \ \{\neg R(\bar{t}) \mid R \in \sigma, \bar{t} \in Const^{\mathrm{ar}(R)}, D \not\models R(\bar{t})\},$$

which contains negations of all ground atoms $R(\bar{t})$ that are assumed to be false under the CWA. The models of $D \cup \overline{D}$ are called the *CWA-models* of D, and a query q over σ is answered by $cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of all CWA-models of D.

Translated into the data exchange framework, we obtain:

Definition 4.1. (RCWA-solution, RCWA-answers) Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let q be a query over τ.

1. A *RCWA-solution* for S under M is a ground target instance T for M such that $S \cup T$ is a model of $D_{M,S} \cup \overline{D_{M,S}}$.

2. We call $cert_{\mathrm{RCWA}}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of the RCWA-solutions for S under M, the *RCWA-answers to q on M and S*.

It is not hard to see that $cert_{\mathrm{RCWA}}$ coincides with $cert_{\mathrm{CWA}}$ on schema mappings defined by full st-tgds.

Proposition 4.2. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of full st-tgds, let S be a source instance for M, and let q be a query over τ. Then we have $cert_{RCWA}(q, M, S) = cert_{CWA}(q, M, S)$.*

However, for schema mappings that contain non-full st-tgds, $cert_{\mathrm{RCWA}}$ may lead to answers that are inconsistent with M and S. This is illustrated by the following example, which is based on Example 8 in [27].

Example 4.3. Let $M = (\{P\}, \{E\}, \Sigma)$, where Σ consists of the st-tgd $\forall x(P(x) \rightarrow \exists z E(x, z))$, and let S be the source instance for M with $P^S = \{a\}$. There is no RCWA-solution for S under M, because for all $b, c \in Const$,

$$D_{M,S} \ = \ \{P(a), \forall x(P(x) \rightarrow \exists z E(x, z))\} \ \not\models \ E(b, c),$$

and therefore,

$$\begin{aligned} \overline{D_{M,S}} \ = \ & \{\neg P(b) \mid b \in Const, b \neq a\} \cup \\ & \{\neg E(b, c) \mid b, c \in Const\}. \end{aligned}$$

For $q(x) := \exists z E(x, z)$, we thus have $cert_{\mathrm{RCWA}}(q, M, S) = \emptyset$. In other words, $cert_{\mathrm{RCWA}}(q, M, S)$ tells us that there is no value z with $E(a, z)$. This is clearly inconsistent with M and S, since M and S tell us that there is a value z with $E(a, z)$. Thus, intuitively, the set of answers should be $\{a\}$.

4.2 The Generalized Closed World Assumption (GCWA)

Minker [26] extended the CWA to the *generalized closed world assumption (GCWA)* as follows. Given a deductive database D over a schema σ, he first defines[1]

$$\overline{\overline{D}} := \{\neg R(\bar{t}) \mid R \in \sigma, \bar{t} \in Const^{\mathrm{ar}(R)}, \text{ and for all } \subseteq\text{-minimal models } I \text{ of } D, \bar{t} \notin R^I\},$$

which, as in the case of the CWA, contains negations of all ground atoms that are assumed to be false under the GCWA. The models of $D \cup \overline{\overline{D}}$ are called *GCWA-models* of D, and a query $q(\bar{x})$ over σ is answered by $cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of all GCWA-models of D.

The intuition behind the above definitions is that each ground atom in some $\subseteq$-minimal model of D is in some sense an atom that D "speaks" about. For ground atoms that do not occur in any $\subseteq$-minimal model of D, this means that they are merely "invented", and can therefore safely be assumed to be false.

Translated into the data exchange framework, we obtain:

DEFINITION 4.4. (GCWA-solution, GCWA-answers) Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let q be a query over τ.

1. A *GCWA-solution* for S under M is a ground target instance T for M such that $S \cup T$ is a model of $D_{M,S} \cup \overline{\overline{D_{M,S}}}$.
2. We call $cert_{\mathrm{GCWA}}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of the GCWA-solutions for S under M, the *GCWA-answers to q on M and S*.

Similar to the RCWA-answers semantics, it can be shown that $cert_{\mathrm{GCWA}}$ coincides with $cert_{\mathrm{CWA}}$ on schema mappings defined by full st-tgds. Moreover, $cert_{\mathrm{GCWA}}$ leads to "good" answers to the query in Example 4.3:

EXAMPLE 4.5. Recall the schema mapping M, the source instance S, and the query q from Example 4.3. We now have

$$\overline{\overline{D_{M,S}}} = \{\neg P(b) \mid b \in Const, b \neq a\} \cup \{\neg E(b,c) \mid b, c \in Const, b \neq a\}.$$

because each atom of the form $E(a, c)$ is true in some $\subseteq$-minimal model of $D_{M,S}$, and each atom of the form $E(b, c)$ with $b \neq a$ is false in all $\subseteq$-minimal models of $D_{M,S}$. Therefore, the GCWA-solutions for S under M are precisely the target instances T for M for which there is a finite nonempty set $B \subseteq Const$ with $T = T_B$, where $E^{T_B} = \{(a, b) \mid b \in B\}$. It follows that $cert_{\mathrm{GCWA}}(q, M, S) = \{a\}$, as desired.

Nevertheless, there are cases where the GCWA is still quite unsatisfactory, as shown by the following example:

EXAMPLE 4.6. Consider a slight extension of the schema mapping from Example 4.3, namely $M = (\{P\}, \{E, F\}, \Sigma)$, where Σ consists of the st-tgd

$$\theta := \forall x\big(P(x) \to \exists z_1 \exists z_2 (E(x, z_1) \wedge F(z_1, z_2))\big).$$

[1]Recall that an instance I is $\subseteq$-minimal in a set $\mathcal{I}$ of instances if $I \in \mathcal{I}$, and there is no $I' \in \mathcal{I}$ with $I' \subseteq I$. In the definition of $\overline{\overline{D}}$, a $\subseteq$-minimal model of D is a model of D that is $\subseteq$-minimal in the set of all models of D.

Let S be the source instance for M with $P^S = \{a\}$. Then,

$$\overline{\overline{D_{M,S}}} = \{\neg P(b) \mid b \in Const, b \neq a\} \cup \{\neg E(b,c) \mid b, c \in Const, b \neq a\}.$$

Note that for all $b, c \in Const$ we have $\neg F(b, c) \notin \overline{\overline{D_{M,S}}}$, since the target instance T for M with $P^T = \{a\}$, $E^T = \{(a, b)\}$ and $F^T = \{(b, c)\}$ is a $\subseteq$-minimal model of $D_{M,S}$. So, the GCWA-solutions for S under M are the target instances T for M for which there is a finite nonempty set $B \subseteq Const$ with the following properties: (1) $E^T = \{(a, b) \mid b \in B\}$, and (2) for at least one $b \in B$ there is some $c \in Const$ with $(b, c) \in F^T$. In particular, the target instance T^* with $E^{T^*} = \{(a, b)\}$ and $F^{T^*} = \{(b, c), (d, e)\}$ is a GCWA-solution for S under M. For the query

$$q := \forall z_1 \forall z_2 (F(z_1, z_2) \to \exists x E(x, z_1))$$

we thus have $cert_{\mathrm{GCWA}}(q, M, S) = \emptyset$.

So, $cert_{\mathrm{GCWA}}(q, M, S)$ tells us that it is possible that there is a tuple (b, c) in F for which (a, b) is not in E. However, θ and S do not "mention" this possibility. In particular, θ and S only tell us that there are one or more pairs $(b, c) \in Const^2$ such that $E(a, b)$ and $F(b, c)$ occur together in a solution. Thus, whenever $E(a, b)$ is present for some $b \in Const$, then $F(b, c)$ should be present for some $c \in Const$. Similarly, whenever $F(b, c)$ is present for some $b, c \in Const$, then $E(a, b)$ should be present.

4.3 Extensions of the GCWA

Various extensions of the GCWA have been proposed. One of these extensions is the *extended GCWA (EGCWA)* by Yahya and Henschen [29], which restricts the set of models of a deductive database D to the $\subseteq$-minimal models of D. So, given a schema mapping $M = (\sigma, \tau, \Sigma)$ and a source instance S for M, a *EGCWA-solution* for S under M can be defined as a ground $\subseteq$-minimal solution for S under M, and given a query $q(\bar{x})$ we can define

$$cert_{\mathrm{EGCWA}}(q, M, S) := cert(q, \mathcal{I}),$$

where $\mathcal{I}$ is the set of all EGCWA-solutions for S under M. Then, for the schema mapping M, the source instance S for M, and the query q in Example 4.5, $cert_{\mathrm{EGCWA}}(q, M, S) = cert_{\mathrm{GCWA}}(q, M, S)$, and for the schema mapping M, the source instance S for M, and the query q in Example 4.6, $cert_{\mathrm{EGCWA}}(q, M, S) \neq \emptyset$, as desired. However, the EGCWA seems to be too strong in the sense that it removes too many solutions from the set of all solutions. We illustrate this by the following example.[2]

EXAMPLE 4.7. Let $M = (\{P\}, \{E\}, \Sigma)$ be a schema mapping, where Σ consists of

$$\theta = \forall x (P(x) \to \exists^{[2,3]} z E(x, z)),$$

where $\exists^{[2,3]} z E(x, z)$ is an abbreviation for "there exist two or three z such that $E(x, z)$". Let S be the source instance for M with $P^S = \{a\}$. Then the $\subseteq$-minimal solutions for S under M have the form $\{E(a, b_1), E(a, b_2)\}$, where b_1, b_2 are distinct constants. Thus, for

$$q(x) := \exists z_1 \exists z_2 \big(E(x, z_1) \wedge E(x, z_2) \wedge \forall z_3 (E(x, z_3) \to (z_3 = z_1 \vee z_3 = z_2))\big),$$

[2]One can also use an argumentation as in Example 3.1, but Example 4.7 seems to make our point a bit more obvious.

we have $cert_{\text{EGCWA}}(q, M, S) \neq \emptyset$. In other words, the answer $cert_{\text{EGCWA}}(q, M, S)$ excludes the possibility that there are three distinct values b_1, b_2, b_3 with $E(a, b_i)$ for each $i \in \{1, 2, 3\}$. But θ and S explicitly mention this possibility. Thus, intuitively, $cert_{\text{EGCWA}}$ is inconsistent with M and S.

To conclude this section, let us consider the *possible worlds semantics (PWS)* by Chan [7]. An obvious translation of the PWS for the case of schema mappings defined by st-tgds is as follows: Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds, and let S be a source instance for M. The definition of a PWS-solution for S under M can be given in terms of *justifications*, as in [23, 19]. Given a target instance T for M and an atom $R(\bar{t}) \in T$, we say that $R(\bar{t})$ is *justified* in T under M and S if and only if there is a st-tgd $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ in Σ, tuples $\bar{a}, \bar{b}$ over $\text{dom}(S)$ with $S \models \varphi(\bar{a}, \bar{b})$, and a tuple $\bar{u}$ over $\text{dom}(T)$ such that $T \models \psi(\bar{a}, \bar{u})$, and $R(\bar{t})$ is one of the atoms in $\psi(\bar{a}, \bar{u})$. A PWS-solution for S under M is then a ground solution T for S under M such that all atoms in T are justified in T under M and S. For a query q over τ, we let

$$cert_{\text{PWS}}(q, M, S) \;:=\; cert(q, \mathcal{I}),$$

where $\mathcal{I}$ is the set of all PWS-solutions for S under M. However, $cert_{\text{PWS}}$ does not respect logical equivalence of schema mappings as can easily be verified using the schema mapping, the source instance and the query in Example 3.2.

5. GCWA*-ANSWERS

In this section, we introduce the GCWA*-answers semantics, and argue that it is well-suited for answering nonmonotonic queries with respect to a number of schema mappings, including schema mappings defined by st-tgds and egds.

As a motivating example, let us consider the schema mapping M and the source instance S for M from Example 4.3. Let $\mathcal{T}$ be the set of all GCWA-solutions for S under M. As shown in Example 4.5, $\mathcal{T}$ consists of all target instances T for M such that there is a nonempty finite set $B \subseteq \mathit{Const}$ with $T = T_B$, where $E^{T_B} = \{(a, b) \mid b \in B\}$. Intuitively, $\mathcal{T}$ precisely reflects the information contained in M and S: taking the usual semantics of existential quantification, M and S just tell us that there is one $b \in \mathit{Const}$ such that $E(a, b)$ holds, or there are two distinct $b_1, b_2 \in \mathit{Const}$ such that $E(a, b_1)$ and $E(a, b_2)$ hold, or there are three distinct $b_1, b_2, b_3 \in \mathit{Const}$ such that $E(a, b_1)$, $E(a, b_2)$ and $E(a, b_3)$ hold, and so on. The case that there are n distinct constants $b_1, \ldots, b_n$ such that $E(a, b_i)$ holds for each $i \in \{1, \ldots, n\}$ is captured by T_B, where $B := \{b_1, \ldots, b_n\}$. Since M and S specify a translation of S from source to target, it seems to be reasonable to assume that the result of the translation is one of the instances in $\mathcal{T}$. Thus, intuitively, the certain answers to a query on $\mathcal{T}$ are based precisely on the information contained in M and S.

Note that each instance in $\mathcal{T}$ is a union of $\subseteq$-minimal solutions for S under M. On the other hand, consider the schema mapping M, the source instance S for M, and the GCWA-solution T^* for S under M from Example 4.6. Then T^* is *not* a union of $\subseteq$-minimal solutions for S under M. However, let $\mathcal{T}$ be the set of all ground target instances for M that are unions of $\subseteq$-minimal solutions for S under M. That is, let $\mathcal{T}$ be the set of all ground target instances T for M such that $E^T = \{(a, b) \mid (b, c) \in F^T \text{ for some } c \in \mathit{Const}\}$ and $F^T \neq \emptyset$. Then, intuitively, $\mathcal{T}$ precisely reflects the information contained in M and S: taking once again the usual semantics of existential quantification, M and S tell us that there is one pair $(b, c) \in \mathit{Const}^2$ such that $E(a, b)$ and $F(b, c)$ hold, or there are two pairs $(b_1, c_1), (b_2, c_2) \in \mathit{Const}^2$ such that $E(a, b_i)$ and $F(b_i, c_i)$ hold for each $i \in \{1, 2\}$, and so on. Note that the certain answers to the query q from Example 4.6 on $\mathcal{T}$ are nonempty, as desired.

The preceding two examples suggest that it might be a good idea to answer queries by the certain answers over all solutions that are unions of ground $\subseteq$-minimal solutions. We call such solutions *GCWA*-solutions*:

Definition 5.1. (GCWA*-solution, GCWA*-answers) Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let q be a query over τ.

1. A *GCWA*-solution* for S under M is a ground target instance T for M such that T is a solution for S under M, and T is a union of $\subseteq$-minimal solutions for S under M.

2. We call $cert_{\text{GCWA}^*}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of the GCWA*-solutions for S under M, the *GCWA*-answers to q on M and S*.

Note that the GCWA*-answers are preserved under logical equivalence.

Let us now argue for a more general class of schema mappings that, intuitively, GCWA*-solutions directly reflect the information in a schema mapping from this class and a source instance. Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of *right-monotonic $L_{\infty\omega}$-st-tgds*, which are $L_{\infty\omega}$-sentences of the form

$$\theta \;:=\; \forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x})),$$

where φ is a $L_{\infty\omega}$-formula over σ, and ψ is a *monotonic* $L_{\infty\omega}$-formula over τ. We assume that the universal quantifiers, and the quantifiers in φ are relativized to the active domain over σ. Moreover, we assume that the quantifiers in ψ are relativized to the active domain over τ. This ensures that for each instance S over σ, and each instance T over τ, $S \cup T \models \theta$ if and only if for all $\bar{a} \in (\text{dom}(S) \cup \text{dom}(\theta))^{|\bar{x}|}$, where $\text{dom}(\theta)$ is the set of all constants in θ, $S \models \varphi(\bar{a})$ implies $T \models \psi(\bar{a})$. Note that right-monotonic $L_{\infty\omega}$-st-tgds in particular capture st-tgds. Now, given a source instance S for M, let

$$\Psi_{M,S} \;:=\; \{\psi(\bar{a}) \mid \text{there exists } \forall \bar{x}(\varphi(\bar{x}) \rightarrow \psi(\bar{x})) \in \Sigma \text{ and } \bar{a} \in \mathit{Const}^{|\bar{x}|} \text{ with } S \models \varphi(\bar{a})\}.$$

Then for each ground target instance T for M, it holds that T is a solution for S under M if and only if T satisfies all sentences in $\Psi_{M,S}$. Since all sentences in $\Psi_{M,S}$ are monotonic, $\Psi_{M,S}$ is logically equivalent to the sentence

$$\psi_{M,S} \;:=\; \bigvee_{T_0 \in \mathcal{T}_0} \bigwedge_{R(\bar{t}) \in T_0} R(\bar{t}),$$

where $\mathcal{T}_0$ is the set of all $\subseteq$-minimal ground solutions for S under M (i.e., for all ground instances T over τ, we have $T \models \psi_{M,S}$ if and only if T satisfies all sentences in $\Psi_{M,S}$). Note that $\psi_{M,S}$ tells us that the target contains one ground $\subseteq$-minimal solution for S under M, or the target contains two distinct ground $\subseteq$-minimal solutions for S under M,

and so on. So, intuitively, the information contained in $\psi_{M,S}$ (and thus in M and S) is directly reflected by the set of all instances T over τ for which there is a set $\mathcal{T}$ with one or more instances in $\mathcal{T}_0$ such that T is the union of all the instances in $\mathcal{T}$. This set corresponds to the set of all GCWA*-solutions for S under M. A similar argumentation shows that, intuitively, the set of GCWA*-solutions directly reflects the information in M and S if M is defined by a set of right-monotonic $L_{\infty\omega}$-st-tgds, and egds. Furthermore, we are not restricted to $L_{\infty\omega}$-st-tgds and egds:

EXAMPLE 5.2. Consider once again the schema mapping M, the source instance S for M, and the query q from Example 4.7. For each ground target instance T for M that is the union of $\subseteq$-minimal solutions for S under M there exists a nonempty finite set $B \subseteq Const$ with $E^T = \{(a,b) \mid b \in B\}$. Due to the constraint θ in M, T is a GCWA*-solution for S under M if and only if $2 \leq |B| \leq 3$. Thus, intuitively, the set of GCWA*-solutions directly reflects the information contained in M and S. It follows that $cert_{\text{GCWA}^*}(q, M, S) = \emptyset$, as desired.

Clearly, the notion of GCWA*-solutions generalizes the notion of EGCWA-solutions in the sense that every EGCWA-solution is a GCWA*-solution. The following proposition implies that the notion of GCWA*-solutions is a restriction of the notion of GCWA-solutions (since $\neg R(\bar{t}) \equiv R(\bar{t}) \rightarrow \bigvee \emptyset$).

PROPOSITION 5.3. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let $D := D_{M,S}$. Consider*

$$D^* := \{R(\bar{t}) \rightarrow \varphi \mid R \in \sigma \cup \tau,\ \bar{t} \in Const^{\text{ar}(R)},\ \varphi \text{ is a monotonic } L_{\infty\omega} \text{ sentence over } \sigma \cup \tau \text{ that is satisfied in every } \subseteq\text{-minimal model } I \text{ of } D \text{ with } \bar{t} \in R^I\}.$$

Then for all ground target instances T for M we have: T is a GCWA-solution for S under M if and only if $S \cup T$ is a model of $D \cup D^*$.*

The following result translates Theorem 5 in [26] from GCWA-solutions to GCWA*-solutions, and shows that for a given schema mapping M and a source instance S for M, the set $D_{M,S} \cup D^*_{M,S}$, where $D^*_{M,S}$ is defined as in Proposition 5.3, is maximally consistent in the sense that the addition of any sentence ψ of the form $R(\bar{t}) \rightarrow \varphi$, where φ is a monotonic $L_{\infty\omega}$ sentence and $D_{M,S} \cup D^*_{M,S} \not\models \psi$, leads to a set of constraints that is inconsistent with $D_{M,S} \cup D^*_{M,S}$.

PROPOSITION 5.4. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a nonempty source instance for M, let $D := D_{M,S}$ and let $D' := D \cup D^*$.*

1. *For all monotonic $L_{\infty\omega}$-sentences φ over $\sigma \cup \tau$, we have $D \models \varphi$ if and only if $D' \models \varphi$.*
2. *For all ground atoms $R(\bar{t})$ over $\sigma \cup \tau$, all monotonic $L_{\infty\omega}$ sentences φ over $\sigma \cup \tau$, and $\psi \in \{\varphi, R(\bar{t}) \rightarrow \varphi\}$ with $D' \not\models \psi$:*
   (a) *$D' \cup \{\psi\}$ has no model, or*
   (b) *there is a monotonic $L_{\infty\omega}$-sentence χ over $\sigma \cup \tau$ such that $D' \cup \{\psi\} \models \chi$, but $D' \not\models \chi$.*

6. THE DATA COMPLEXITY OF COMPUTING GCWA*-ANSWERS

In this section, we study the data complexity of computing GCWA*-answers, where data complexity means that the schema mapping and the query to be evaluated is fixed (i.e., not part of the input). We concentrate on schema mappings defined by st-tgds only. For a schema mapping M and a query language L, we are particularly interested in whether there are algorithms $\mathbb{A}_1, \mathbb{A}_2$ with the following properties:

- $\mathbb{A}_1$ takes a source instance S for M as input and computes a solution T for S under M, and
- $\mathbb{A}_2$ takes T and a query $q \in L$ as input and computes $cert_{\text{GCWA}^*}(q, M, S)$.

The second property is particularly important, since a common assumption in data exchange is that the schema mapping and the source instance are not available once the data translation has been performed, so that the query must be answered based on a materialized solution. Ideally, both $\mathbb{A}_1$, and $\mathbb{A}_2$ for fixed $q \in L$, run in polynomial time.

For proving lower bounds, we consider the problem

EVAL(M, q)	
Input:	a source instance S for M, and a tuple $\bar{t} \in Const^{\lvert\bar{x}\rvert}$
Question:	Is $\bar{t} \in cert_{\text{GCWA}^*}(q, M, S)$?

for schema mappings $M = (\sigma, \tau, \Sigma)$ and queries $q(\bar{x})$ over τ. The complexity of this problem can be seen as a lower bound on the joint complexity of finding T as above, and obtaining $cert_{\text{GCWA}^*}(q, M, S)$ from T. If, for example, EVAL(M, q) is co-NP-complete, then finding T is intractable, or computing $cert_{\text{GCWA}^*}(q, M, S)$ from T is intractable.

We first study the complexity of computing the GCWA*-answers to monotonic queries and existential queries in Section 6.1, and turn to universal queries in Section 6.2.

6.1 Monotonic queries and existential queries

For monotonic queries, all results obtained for the certain answers semantics (see, e.g., [10, 25, 3, 21, 23, 8, 4, 6]) carry over to the GCWA*-answers semantics:

PROPOSITION 6.1. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let q be a monotonic query over τ. Then, $cert_{GCWA^*}(q, M, S)$ equals the set of the certain answers to q on M and S.*

In particular, for computing GCWA*-answers to queries preserved under homomorphisms with respect to a schema mapping $M = (\sigma, \tau, \Sigma)$, where Σ consists of st-tgds, it suffices to compute, given a source instance S for M, an arbitrary universal solution T for S under M such as Core(M, S), which can be done in polynomial time by Theorem 2.1; then for each query $q(\bar{x})$ that is preserved under homomorphisms, the GCWA*-answers to q on M and S coincide with $q(T)_\downarrow = \{\bar{t} \in Const^{\lvert\bar{x}\rvert} \mid \bar{t} \in q(T)\}$, which can be computed from T in time polynomial in $|T|$, for fixed q.

For general monotonic queries, like unions of conjunctive queries with inequalities, the concept of an *ihom-universal model set* [8] may be useful. Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, and let S be a source instance for M. An ihom-universal model set for M and S is a set $\mathcal{T}$ of solutions for

S under M such that for every solution T' for S under M, there is some $T \in \mathcal{T}$ and an *injective* homomorphism from T to T', and no proper subset of $\mathcal{T}$ is a ihom-universal model set for M and S. It was shown in [8] that the certain answers to a monotonic query q on M and S can be computed using an ihom-universal model set $\mathcal{T}$ by computing the intersection of the query results $q(T)_\downarrow$ over all $T \in \mathcal{T}$. However, ihom-universal model sets can get very large: For the simple schema mapping $M = (\{P\}, \{R\}, \Sigma)$, where Σ consists of the st-tgd $P(x) \rightarrow \exists y \exists z R(x, y, z)$, and any source instance S for M with $|P^S| = n$, it is not hard to see that there are $\gg 2^n$ many solutions in every ihom-universal model set for M and S. On the other hand, ihom-universal model sets must in general be large: as shown by Mądry [25], there is a schema mapping M defined by LAV st-tgds, and a Boolean conjunctive query q with just two inequalities such that $\text{EVAL}(M, q)$ is co-NP-hard.

What seems to be more practical is to compute a "small" representation of the set of all relevant solutions such as $\text{Core}(M, S)$, and given a union q of conjunctive queries with inequalities over τ, compute a set of solutions that is sufficient for computing the certain answers to q on M and S. For example, [10] have shown that for a Boolean query q that is the union of conjunctive queries with at most one inequality per disjunct, the certain answers can be computed from an arbitrary universal solution in polynomial time: First, a new solution T is computed from the precomputed universal solution using the *chase procedure* [10]. If the chase fails to compute a solution, then the certain answers to q on M and S are nonempty. Otherwise, the certain answers to q on M and S are $q(T)$.

We now turn to *existential queries*, which are FO queries of the form $\exists \bar{x} \varphi$, where φ is a quantifier-free FO query. A particular class of existential queries are *conjunctive queries with negation* ($\text{CQ}^\neg$ queries, for short), which are queries of the form $\exists \bar{x}(L_1 \wedge \cdots \wedge L_k)$ such that for each $i \in \{1, \ldots, k\}$, we have $L_i \in \{R(\bar{u}), \neg R(\bar{u})\}$, where R is a relation symbol, and $\bar{u}$ is a tuple of elements in *Const* and $\bar{x}$. A simple reduction from the CLIQUE problem [15] shows that $\text{EVAL}(M, q)$ is already co-NP-hard for schema mappings M defined by LAV tgds and $\text{CQ}^\neg$ queries with only one negated atom:

PROPOSITION 6.2. *There exists a schema mapping $M = (\sigma, \tau, \Sigma)$, where Σ consists of two LAV tgds, and a Boolean $CQ^\neg$ query q over τ with one negated atomic query such that* $\text{EVAL}(M, q)$ *is* co-NP*-complete.*

Even more, moving from existential queries to FO queries with an $\exists^* \forall$ quantifier prefix, we can show:

PROPOSITION 6.3. *There exists a schema mapping $M = (\sigma, \tau, \Sigma)$, where Σ consists of two LAV tgds, and a Boolean $\exists^* \forall$ FO query q over τ such that* $\text{EVAL}(M, q)$ *is undecidable.*

6.2 Universal queries

This section presents the technically most challenging result of this article: Theorem 6.6, which states that for certain schema mappings M defined by st-tgds, and for universal queries q, there is a polynomial time algorithm for computing the GCWA*-answers to q from the core of the universal solutions. Here, a *universal query* is a FO query of the form $\forall \bar{y} \varphi(\bar{x}, \bar{y})$, where φ is quantifier-free. Our first result is:

PROPOSITION 6.4. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let q be a universal query over τ. Then,* $\text{EVAL}(M, q)$ *is in* co-NP.

The main result of this section is based on the following notion of *packed st-tgds*:

DEFINITION 6.5. (packed st-tgd)
A st-tgd $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ is *packed* if for all distinct atomic formulas ψ_1, ψ_2 in ψ, there is a variable in $\bar{z}$ that occurs both in ψ_1 and in ψ_2.

I think that the class of schema mappings defined by packed st-tgds is an interesting class of schema mappings. For example, consider a schema mapping M defined by st-tgds $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, where ψ contains at most two atoms with variables from $\bar{z}$. Then M is logically equivalent to a schema mapping defined by packed st-tgds, since every st-tgd θ in M is logically equivalent to a set Θ of packed st-tgds, where the size of Θ is at most the number of atomic formulas in the head of θ. An example of a st-tgd that is *not* packed is $\forall x(P(x) \rightarrow \exists z_1 \exists z_2 \exists z_3(E(x, z_1) \wedge E(z_1, z_2) \wedge E(z_2, z_3)))$.

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

THEOREM 6.6. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let q be a universal query over τ. There is a polynomial time algorithm that, given* $\text{Core}(M, S)$ *for some source instance S for M, outputs $cert_{GCWA^*}(q, M, S)$.*

Note that Theorem 6.6 and Theorem 2.1 immediately imply that for every schema mapping M specified by packed st-tgds, and for every universal query q over M's target schema, there is a polynomial time algorithm that takes a source instance S for M as input, and outputs $cert_{\text{GCWA}^*}(q, M, S)$. Furthermore, an interesting consequence of Theorem 6.6 is as follows. Let M be a schema mapping defined by packed st-tgds, and let S be a source instance for M. Then the same solution for S under M, namely $\text{Core}(M, S)$, can be used to efficiently compute both the certain answers to queries preserved under homomorphisms on M and S, and the GCWA*-answers to universal queries on M and S (if M and the query are not part of the input). If one considers to compute only the certain answers to queries preserved under homomorphisms on M and S, and the GCWA*-answers to universal queries on M and S, it suffices therefore to compute $\text{Core}(M, S)$, which, by Theorem 2.1, is possible in polynomial time if M is not part of the input.

Let us now turn to the proof of Theorem 6.6. Note that Theorem 6.6 follows easily from:

THEOREM 6.7. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let $q(\bar{x})$ be a universal query over τ. Then there is a polynomial time algorithm that, given* $\text{Core}(M, S)$ *for some source instance S for M, and a tuple $\bar{t} \in Const^{|\bar{x}|}$, decides whether $\bar{t} \in cert_{GCWA^*}(q, M, S)$.*

The remainder of this section is devoted to the proof of Theorem 6.7. Let us begin by showing how $\text{Core}(M, S)$ can be used to decide membership in $cert_{\text{GCWA}^*}(q, M, S)$.

6.2.1 GCWA*-Answers and the Core

Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ is a set of packed st-tgds, and let $q(\bar{x})$ be a universal query over τ. Given $\text{Core}(M, S)$ and a tuple $\bar{t} \in Const^{|\bar{x}|}$, how can we decide whether $\bar{t} \in cert_{\text{GCWA}^*}(q, M, S)$?

Observe that $\bar{t} \notin cert_{\text{GCWA}^*}(q, M, S)$ if and only if there is a GCWA*-solution $\tilde{T}$ for S under M with $\tilde{T} \models \neg q(\bar{t})$. By the definition of GCWA*-solution and the fact that Σ consists of st-tgds, the latter is the case precisely if there is a nonempty finite set $\mathcal{T}$ of ground $\subseteq$-minimal solutions for S under M with $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Using the following lemma, we can reformulate this in terms of $\text{Core}(M, S)$:

LEMMA 6.8. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. Then the set of all ground $\subseteq$-minimal solutions for S under M is precisely the set of all $\subseteq$-minimal instances in $poss(\text{Core}(M, S))$.*

Given $\text{Core}(M, S)$ and $\bar{t}$, it remains to decide whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(\text{Core}(M, S))$ such that $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Since q is a universal query, $\neg q$ is logically equivalent to a query of the form $\exists \bar{y}\, \varphi(\bar{x}, \bar{y})$. Before tackling the general case in Section 6.2.3, the following section deals with the case that $\bar{y}$ contains no variable and φ consists of a single atomic FO formula.

6.2.2 Finding Atoms in Minimal Possible Worlds

Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, let $T := \text{Core}(M, S)$ for some source instance S for M, and let $R(\bar{t})$ be an atom over τ. To decide whether there is a $\subseteq$-minimal instance T_0 in $poss(T)$ with $R(\bar{t}) \in T_0$, it suffices to consider only the instances in $min_C(T)$, where C is the set of all constants in $\bar{t}$, and $min_C(T)$ is defined as follows:

DEFINITION 6.9. ($val_C(T)$, $min_C(T)$) Let T be a naive table, and let $C \subseteq Const$.

1. Let $val_C(T)$ be the set of all functions $f\colon \text{dom}(T) \to \text{dom}(T) \cup C$ that are legal for T.
2. Let $min_C(T)$ be the set of all instances $\hat{T}$ for which there is some $f \in val_C(T)$ with $\hat{T} = f(T)$, and there is no $f' \in val_C(T)$ with $f'(T) \subsetneq \hat{T}$.

PROPOSITION 6.10. *Let T be a naive table, let $C \subseteq Const$.*

1. *For each $T_0 \in poss(T)$, the following are equivalent:*
   (a) *T_0 is a $\subseteq$-minimal instance in $poss(T)$.*
   (b) *There is an instance $T_0' \in min_C(T)$ and an injective valuation v of T_0' such that $v(T_0') = T_0$, and $v^{-1}(c) = c$ for all $c \in \text{dom}(T_0) \cap C$.*
2. *If T is a core, then $T \in min_C(T)$.*
3. *Each instance in $min_C(T)$ is a core. Moreover, there is a naive table T' and a function $f \in val_C(T')$ such that $f(T')$ is a core, but $f(T') \notin min_C(T')$.*

The size of $min_C(T)$ can be exponential in the size of T, so that it is not possible to enumerate all instances in $min_C(T)$ in polynomial time, given T and $R(\bar{t})$ as input. To tackle this problem, we take advantage of a nice structural property of T that can be described in terms of *atom blocks*:

DEFINITION 6.11. (atom block) Let T be a naive table.

- The *Gaifman graph of the atoms of T* is the undirected graph whose vertices are the atoms of T, and which has an edge between two atoms $A, A' \in T$ if and only if $A \neq A'$, and there is a null that occurs in A and A'.
- An *atom block* of T is the set of atoms in a connected component of the Gaifman graph of the atoms of T.

The crucial property of T is:

LEMMA 6.12 ([10]). *For every schema mapping $M = (\sigma, \tau, \Sigma)$, where Σ consists of st-tgds, there is a positive integer bs such that if S is a source instance for M, and B is an atom block of $\text{Core}(M, S)$, then $|\text{nulls}(B)| \leq bs$.*

The following naive algorithm now seems to decide if there is an instance $T_0 \in min_C(T)$ with $R(\bar{t}) \in T_0$:

1. Compute the atom blocks of T.
2. Consider the atom blocks B of T in turn, and
3. if there is an instance $B_0 \in min_C(B)$ with $R(\bar{t}) \in B_0$, accept the input; otherwise reject it.

Since $|\text{nulls}(B)| \leq bs$ for each atom block B of T, we only have to consider at most $|val_C(B)| = |\text{dom}(B) \cup C|^{bs}$ mappings in step 3 to find all the instances B_0 in $min_C(B)$. Thus, the whole algorithm runs in polynomial time. However, Example 6.13 below shows that this algorithm is incorrect. In particular, the example exhibits a naive table T that is a core, and an atom block B of T such that there is an atom of some $\subseteq$-minimal instance $B_0 \in poss(B)$ that is not an atom of any $\subseteq$-minimal instance in $poss(T)$. For all sets $C \subseteq Const$, this implies that there is an atom of some instance $B_0 \in min_C(B)$ that is not an atom of any instance in $min_C(T)$.

EXAMPLE 6.13. Let T be the naive table with

$$E^T = \{(a, b), (a, \bot), (b, \bot), (b, \bot'), (b, \bot''), (\bot', \bot'')\},$$

and consider the atom block

$$B = \{E(b, \bot'), E(b, \bot''), E(\bot', \bot'')\}$$

of T. Note that T is a core. It is not hard to see that every $\subseteq$-minimal instance in $poss(B)$ has one of the following forms:

1. $\{E(b, b)\}$,
2. $\{E(b, c), E(c, c)\}$ with $c \in Const \setminus \{b\}$, or
3. $\{E(b, c), E(b, c'), E(c, c')\}$ with $c, c' \in Const \setminus \{b\}$ and $c \neq c'$.

Thus, there is a $\subseteq$-minimal instance in $poss(B)$ of the third form that contains $E(c, a)$ (substitute c' with a).

However, there is no $\subseteq$-minimal instance in $poss(T)$ that contains $E(c, a)$: Such an instance must be obtained from T by a valuation v of T with $v(\bot') = c$ and $v(\bot'') = a$, since $E(\bot', \bot'')$ is the only atom in T that could be the preimage of $E(c, a)$—all other atoms either have a or b as their first value. However, let v be a valuation of T with $v(\bot') = c$ and $v(\bot'') = a$, and let $f\colon \text{dom}(T) \to \text{dom}(T)$ be such that $f(a) = a$, $f(b) = b$, $f(\bot') = a$ and $f(\bot) = f(\bot'') = \bot$. Then, for $v' := v \circ f$, we have

$$\begin{aligned} v'(T) &= \{E(a, b), E(b, a), E(a, v(\bot)), E(b, v(\bot))\} \\ &\subsetneq \{E(a, b), E(b, a), E(a, v(\bot)), E(b, v(\bot)), \\ &\qquad E(b, c), E(c, a)\} \\ &= v(T). \end{aligned}$$

Thus, $v(T)$ is not $\subseteq$-minimal in $poss(T)$.

Here we use a different approach: we identify a subset $\mathcal{S}$ of $\min_C(T)$ of size polynomial in the size of T such that it suffices to consider only the instances in $\mathcal{S}$ to decide whether $R(\bar{t})$ occurs in some instance in $\min_C(T)$. The set $\mathcal{S}$ consists of all naive tables T_0 for which there is an atom block B of T with $T_0 \in \mathit{min}_C(T, B)$, where $\mathit{min}_C(T, B)$ is defined below.

DEFINITION 6.14. ($\mathit{minval}_C(T, B)$, $\mathit{min}_C(T, B)$)
Let T be a naive table, let B be an atom block of T, let $\overline{B} := T \setminus B$, and let $C \subseteq \mathit{Const}$.

1. Let $\mathit{val}_C(T, B)$ be the set of all functions $f \in \mathit{val}_C(T)$ with $f(\bot) = \bot$ for all $\bot \in \text{nulls}(\overline{B})$ such that all nulls that occur in $f(B) \setminus \overline{B}$ belong to nulls(B).
2. Let $\mathit{minval}_C(T, B)$ be the set of all $f \in \mathit{val}_C(T, B)$ such that there is no $f' \in \mathit{val}_C(T, B)$ with $f'(T) \subsetneq f(T)$.
3. Let $\mathit{min}_C(T, B) := \{\text{Core}(f(T)) \mid f \in \mathit{minval}_C(T, B)\}$.

Using (a simplified version of) the blocks algorithm from [11] for computing the core of a naive table, where the number of nulls in each atom block is bounded by a constant, in polynomial time, we obtain:

PROPOSITION 6.15. *For each positive integer bs, there is a polynomial time algorithm that, given a naive table T such that the number of nulls in each atom block of T is at most bs, an atom block B of T, and a set $C \subseteq \mathit{Dom}$, outputs $\mathit{min}_C(T, B)$.*

The following Lemma 6.16 tells us that the instances in $\mathit{min}_C(T, B)$ indeed belong to $\mathit{min}_C(T)$. To state and to prove the lemma, the notion of a retraction is convenient. Given an instance I, a *retraction* of I is a homomorphism h from I to I such that $h(u) = u$ for all elements u in the range of h. In particular, for all atoms $A \in h(I)$, we have $A \in I$ and $h(A) = A$. It is known that a core of I is an instance J for which there is a retraction h of I with $h(I) = J$, and there is no retraction of J to a proper subinstance of J ([17]).

LEMMA 6.16. *Let T be a naive table, let B be an atom block of T, let $\overline{B} := T \setminus B$, and let $C \subseteq \mathit{Const}$. Then for all $f \in \mathit{minval}_C(T, B)$ there is a retraction h of $\hat{T} := f(T)$ with*

1. *$h(\bot) = \bot$ for all $\bot \in \text{nulls}(f(B) \setminus \overline{B})$,*
2. *$h(\hat{T})$ is a core of $\hat{T}$, and*
3. *$h(\hat{T}) \in \mathit{min}_C(T)$.*

In particular, $\mathit{min}_C(T, B) \subseteq \mathit{min}_C(T)$.

Clearly, the union of the sets $\mathit{min}_C(T, B)$ over all atom blocks B of T does not cover the whole set of instances in $\mathit{min}_C(T)$. However, Lemma 6.20 below tells us that for each atom A of some instance $T_0 \in \mathit{min}_C(T)$ there is an atom block B of T and an instance $T_B \in \mathit{min}_C(T, B)$ that contains an atom A' isomorphic to A in the following sense:

NOTATION 6.17. We say that two atoms A_1, A_2 are *isomorphic*, and we write $A_1 \cong A_2$, if the instances $\{A_1\}$ and $\{A_2\}$ are isomorphic.

Note that $R(u_1, \ldots, u_r)$ and $R'(u'_1, \ldots, u'_{r'})$ are isomorphic if and only if $R = R'$, $r = r'$, and for all $i, j \in \{1, \ldots, r\}$, $u_i \in \mathit{Const}$ if and only if $u'_i \in \mathit{Const}$, $u_i \in \mathit{Const}$ implies $u_i = u'_i$, and $u_i = u_j$ if and only if $u'_i = u'_j$.

Lemma 6.20 is based on the following notion of a *packed* atom block:

DEFINITION 6.18. (packed atom block) An atom block B of a naive table is called *packed* if for all atoms $A, A' \in B$ with $A \neq A'$, there is a null that occurs in A and A'.

Immediately from the definitions, we obtain:

PROPOSITION 6.19. *If $M = (\sigma, \tau, \Sigma)$ is a schema mapping, where Σ consists of packed st-tgds, and S is a source instance for M, then each atom block of* Core(M, S) *is packed.*

We are now ready to state the main result of the present Section 6.2.2:

LEMMA 6.20. *Let T be a naive table such that T is a core, and each atom block of T is packed. Let $B_1, \ldots, B_n$ be the atom blocks of T, let $C \subseteq \mathit{Const}$, and let $T_0 \in \mathit{min}_C(T)$.*

Then for each $i \in \{1, \ldots, n\}$, there is a $T_i \in \mathit{min}_C(T, B_i)$ and a homomorphism h_i from T_i to T_0 with $h_i(T_i) = T_0$ such that for each atom $A \in T_0$ there is a $j \in \{1, \ldots, n\}$ and an atom $A' \in T_j$ with $h_j(A') = A$ and $A \cong A'$.

A polynomial time algorithm that, given T and $R(\bar{t})$, decides whether $R(\bar{t})$ occurs in some $\subseteq$-minimal instance in $\mathit{poss}(T)$ is now as follows. Let C be the set of all constants in $\bar{t}$. Consider each atom block B of T, and each $T_0 \in \mathit{min}_C(T, B)$ in turn, and accept the input if and only if $R(\bar{t}) \in T_0$ for some T_0. By Proposition 6.15, $\mathit{min}_C(T, B)$ can be computed in polynomial time for each atom block B of T, so the whole procedure runs in polynomial time.

6.2.3 Proof of Theorem 6.7

In this section, we sketch a proof of Theorem 6.7. Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let $q(\bar{x})$ be a universal query over τ. We show that there is a polynomial time algorithm that, given $T := \text{Core}(M, S)$ for some source instance S for M, and a tuple $\bar{t} \in \mathit{Const}^{|\bar{x}|}$, decides whether $\bar{t} \in \mathit{cert}_{\text{GCWA}^*}(q, M, S)$.

As shown in Section 6.2.1, we have $\bar{t} \notin \mathit{cert}_{\text{GCWA}^*}(q, M, S)$ if and only if there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ such that $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Now observe that $\neg q$ is logically equivalent to a query $\bar{q}$ of the form

$$\bar{q}(\bar{x}) \;=\; \bigvee_{i=1}^{m} q_i(\bar{x}),$$

where each q_i is an existential query of the form

$$q_i(\bar{x}) \;=\; \exists \bar{y}_i \bigwedge_{j=1}^{n_i} \varphi_{i,j},$$

and each $\varphi_{i,j}$ is an atomic FO formula or the negation of an atomic FO formula. Indeed, since q is a universal query, we have $\neg q \equiv \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$, where φ is quantifier-free. By transforming φ into "disjunctive normal form", we obtain a query of the form $\exists \bar{y} \bigvee_{i=1}^{m} \bigwedge_{j=1}^{n_i} \varphi_{i,j}$, where each $\varphi_{i,j}$ is an atomic FO formula or the negation of an atomic FO formula. By moving existential quantifiers inwards, we finally obtain $\bar{q}$. It remains therefore to decide whether there is some $i \in \{1, \ldots, m\}$ and a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $\mathit{poss}(T)$ such that $\bigcup \mathcal{T} \models q_i(\bar{t})$.

We are now ready to describe the algorithm. Let bs be a constant as in Lemma 6.12. Then, for each source instance S for M and each atom block B of Core(M, S), we have $|\text{nulls}(B)| \leq bs$. Furthermore, Proposition 6.19 tells us that each atom block of Core(M, S) is packed. Given $T := \text{Core}(M, S)$ for some source instance S for M, and a

tuple $\bar{t} \in Const^{|\bar{x}|}$ as input, the algorithm simply checks for each $i \in \{1, \ldots, m\}$, whether there is a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models q_i(\bar{t})$. If for some $i \in \{1, \ldots, m\}$, there is such a set $\mathcal{T}$, the algorithm rejects the input; otherwise, it accepts the input. By the following lemma, this is possible in polynomial time:

LEMMA 6.21. *Let $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ be a FO query over τ, where $\varphi = \bigwedge_{i=1}^{n} \varphi_i$, and each φ_i is an atomic FO formula or the negation of an atomic FO formula. For each positive integer bs, there is a polynomial time algorithm that decides:*

$\text{CoreEval}_{\tau,bs}$	
Input:	*a naive table T over τ such that T is a core and each atom block of T is packed and contains at most bs nulls; and a tuple $\bar{t} \in Const^{\lvert\bar{x}\rvert}$*
Question:	*Is there a nonempty finite set $\mathcal{T}$ of $\subseteq$-minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models q(\bar{t})$?*

7. CONCLUSION

We studied query answering semantics for deductive databases (based on the CWA, the GCWA, the EGCWA and the PWS) in the context of relational data exchange. Furthermore, inspired by the GCWA, we developed the GCWA*-answers semantics, and argued that it is well-suited with respect to a number of schema mappings, including schema mappings defined by st-tgds and egds.

We also made first steps towards understanding the complexity of computing GCWA*-answers. Here, we focused on data complexity and schema mappings specified by st-tgds. The main result is the identification of universal queries as a class of queries for which the GCWA*-answers can be computed in polynomial time (data complexity), provided the schema mapping is defined by packed st-tgds. I believe that the techniques used for proving this result can be extended to prove the analogous result for the more general case of schema mappings defined by st-tgds.

Acknowledgements. I am grateful to André Böhm, Nicole Schweikardt and the anonymous referees for invaluable comments on earlier versions of this paper.

8. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] F. N. Afrati and P. G. Kolaitis. Answering aggregate queries in data exchange. In *PODS*, pages 129–138, 2008.

[3] M. Arenas, P. Barceló, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *PODS*, pages 229–240, 2004.

[4] M. Arenas, P. Barceló, and J. Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. In *ICDT*, 2009.

[5] M. Arenas and L. Libkin. XML data exchange: Consistency and query answering. *JACM*, 55(2):Article 7, 2008.

[6] P. Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.

[7] E. P. F. Chan. A possible world semantics for disjunctive databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(2):282–292, 1993.

[8] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *PODS*, pages 149–158, 2008.

[9] R. Fagin. Inverting schema mappings. In *PODS*, pages 50–59, 2006.

[10] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

[11] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *ACM TODS*, 30(1):174–210, 2005.

[12] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM TODS*, 30(4):994–1055, 2005.

[13] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan. Peer data exchange. *ACM TODS*, 31(4):1454–1498, 2006.

[14] H. Gallaire, J. Minker, and J.-M. Nicholas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):153–185, June 1984.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[16] G. Gottlob and A. Nash. Efficient core computation in data exchange. *JACM*, 55(2):Article 9, 2008.

[17] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109(1–3):117–126, 1992.

[18] A. Hernich and N. Schweikardt. Logic and data exchange: Which solutions are "good" solutions? In *Logic and the Foundations of Game and Decision Theory (LOFT 8)*. To appear.

[19] A. Hernich and N. Schweikardt. CWA-solutions for data exchange settings with target dependencies. In *PODS*, pages 113–122, 2007.

[20] T. Imielinski and W. Lipski. Incomplete information in relational databases. *JACM*, 31(4):761–791, 1984.

[21] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

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

[23] L. Libkin. Data exchange and incomplete information. In *PODS*, pages 60–69, 2006.

[24] L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *PODS*, pages 139–148, 2008.

[25] A. Mądry. Data exchange: On the complexity of answering queries with inequalities. *Information Processing Letters*, 94(6):253–257, 2005.

[26] J. Minker. On indefinite databases and the closed world assumption. In *CADE*, volume 138, pages 292–308, June 1982.

[27] R. Reiter. On closed world data bases. In *Logic and Data Bases*, pages 55–76. Plenum Press, 1978.

[28] R. van der Meyden. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, pages 307–356. Kluwer, 1998.

[29] A. Yahya and L. J. Henschen. Deduction in non-horn databases. *Journal of Automated Reasoning*, 1(2):141–160, 1985.