Semantics for Non-Monotone Queries in Data Exchange and Data Integration

André Hernich

Humboldt University Berlin
Germany
hernich@informatik.hu-berlin.de

Abstract

A fundamental question in data exchange and data integration is how to answer queries that are posed against the target schema, or the global schema, respectively. While the certain answers semantics has proved to be adequate for answering *monotone queries*, the question concerning an appropriate semantics for *non-monotone queries* turned out to be more difficult. This article surveys approaches and semantics for answering non-monotone queries in data exchange and data integration.

1998 ACM Subject Classification H.2.5 Heterogeneous Databases, H.2.4 Systems, H.2.8 Database Applications

Keywords and phrases certain answers, closed world assumption, deductive databases, universal solutions, core

Digital Object Identifier 10.4230/DFU.Vol5.10452.161

1 Introduction

Query answering is a fundamental task both in data exchange and data integration. Indeed, the goal of *data integration* is to combine different sources of data and to provide a unified view through which these sources can be queried [29]. The data often resides at the sources while the view is *virtual* (i.e., not materialized). Hence, if a user queries the view, the query has to be answered using the source data, for example, by evaluating suitable queries on the sources and combining their results, or by materializing the relevant part of the view that is needed to answer the query. *Data exchange* is similar to data integration insofar as its goal is to translate databases over a source schema into databases over a target schema [11, 27, 6, 4], whereby providing a view (over the target schema) on the source database. However, unlike in data integration, the view is *materialized* and queries have to be answered directly on that view. In fact, in data exchange it is generally assumed that the source database is not available at the time the target database is queried [11].

In both areas, the basic approach for modeling the relationship between source and target is based on *schema mappings* [29, 27]. Schema mappings describe target databases (over a *target schema*) in terms of source databases (over a *source schema*) by means of high-level declarative assertions (typically expressed in a suitable fragment of first-order logic or even second-order logic). A *solution* for a source database is a target database that makes all assertions hold true. Usually schema mappings are *underspecified*, which means that source databases may have more than one solution. Hence, it is not obvious at all how to answer queries that are posed against the target schema of a schema mapping.

The following approach for answering queries is common in such a setting [29, 27, 6, 4]. Instead of answering a query Q on a single solution, the set of all tuples that are answers to Q on *all* solutions is returned. This set is called the set of the *certain answers* to Q on the given

Data Exchange, Integration, and Streams. *Dagstuhl Follow-Ups*, Volume 5, ISBN 978-3-939897-61-3.
Editors: Phokion G. Kolaitis, Maurizio Lenzerini, and Nicole Schweikardt; pp. 161–184

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

source instance and schema mapping. Informally, it contains all tuples that are *certain* to be answers to Q no matter on which solution Q is evaluated. The certain answers semantics has turned out to be adequate for answering *monotone queries* like unions of conjunctive queries with inequalities in the sense that it yields the best result obtainable from the information in the source database and the schema mapping. Although the definition of the certain answers involves a potentially infinite set of solutions, in many practical settings it is possible to compute them in polynomial time (in data complexity, i.e., for fixed schema mappings and queries) from a single solution called *universal solution* [11, 5], or by evaluating a suitably rewritten query over the source schema (cf., [29] and Chapter 5 of this book).

Researchers soon realized [11, 3, 31] that for *non-monotone* queries, the certain answers semantics may yield results that intuitively seem to be not accurate. The following example illustrates the basic problem.

▶ **Example 1.** Suppose we just want to *copy* source databases to target databases. For instance, assume that source databases contain a single binary relation E and the target database is going to be a database containing a single binary relation E'. Then the schema mapping M describing the translation from source to target could be specified by the *tuple-generating dependency*

$$\theta \;:=\; \forall x, y \big(E(x, y) \rightarrow E'(x, y)\big).$$

Informally, θ says that all tuples in E have to be in E'. Hence, the set of solutions for a source database S consists of all target databases whose relation E' contains all tuples in the relation E of S.

On the other hand, since schema mappings describe translations from source to target, it seems to be natural to expect that the result of translating a source database S according to M is the copy S' of S over the new schema $\{E'\}$. In particular, it seems to be natural to expect that a query posed against the target schema yields the same result as the same query evaluated on S'. However, this is not the case if we answer queries by the set of the certain answers to the query: If the source database S is such that E contains only the tuple (c, d), where c, d are distinct constants, then the expected set of answers to

$$Q(x, y) \;:=\; \forall z \big(E'(x, z) \rightarrow z = y\big)$$

would be $\{(c, d)\}$, yet the set of the certain answers to Q on S and M is empty (since the instance whose relation E' consists of the tuples (c, d) and (c, e) is a solution for S). ◀

As indicated by the example, the problem is really a matter of the semantics of schema mappings. Which target databases should constitute the set of solutions for a given source database? For the schema mapping M in the example, we argued that only the copy of a source database S should be a solution for S under M. To enforce this, we could have used a constraint like $\forall x, y \big(E(x, y) \leftrightarrow E'(x, y)\big)$ stating that a tuple is in E' precisely if it belongs to E. Then the set of the certain answers would be as desired. However, this approach – of using constraints that are not expressible by standard constraints like *tuple-generating dependencies* or *equality-generating dependencies* considered in the literature – seems to have received almost no attention.[1]

The approach pursued in the literature is to use custom-made semantics of query answering [13, 22, 32, 2, 21]. Under each of these semantics, a query Q is answered by the set of the

[1] An exception is [32], where an extension of tgds, *annotated tgds*, is considered whose semantics cannot be captured by any set of tgds and egds.

certain answers to Q with respect to a suitably restricted set $\mathcal{S}$ of solutions (i.e., by the set of all tuples that are answers to Q on all solutions from $\mathcal{S}$). Except for the semantics in [13], different forms of *non-monotonic reasoning*, specifically, variants of the *closed world assumption* [36], are implemented to arrive at the corresponding set of solutions. Here, the basic idea is to consider target databases as solutions only if they can be *derived* in a certain way from the source database and the schema mapping. Apart from trying to remedy the shortcomings of the certain answers semantics when it comes to answering non-monotone queries, I think that non-monotonic reasoning in data exchange and data integration is appealing in its own right. In principle, it allows for more compact specifications of schema mappings, since only the data that is actually moved from the source to the target has to be specified, without saying what should not be in the target database.

This chapter is intended to give an overview on the different semantics for answering non-monotone queries, and the complexity of query answering under those semantics.

The remaining part of this chapter is organized as follows. In Section 2 we introduce basic notions and notation used throughout this chapter, and in Section 3 we recall the certain answers semantics and its behavior on non-monotone queries. Section 4 surveys the different semantics that have been proposed in the literature for answering non-monotone queries. Finally, Section 5 compiles what is known about the complexity of answering non-monotone queries under those semantics.

2 Basics

Below, we recall standard notions from database theory and data exchange used in the rest of this chapter. For a detailed account of these, see, e.g., [1, 4].

We let $[n]$ be the set of all integers m with $1 \leq m \leq n$. Mappings $f\colon A \to B$ are extended to tuples $\bar{a} = (a_1, \ldots, a_k)$ over A via $f(\bar{a}) := (f(a_1), \ldots, f(a_k))$, and to relations $R \subseteq A^k$ via $f(R) := \{f(\bar{a}) \mid \bar{a} \in R\}$.

2.1 Databases

A *schema* is a finite set σ of relation symbols, where each $R \in \sigma$ has a fixed arity $\mathrm{ar}(R) \geq 1$. A *σ-instance* I assigns to each $R \in \sigma$ a finite relation R^I of arity $\mathrm{ar}(R)$. The *active domain of* I, that is, the set of all values that occur in I, is denoted by $\mathrm{dom}(I)$. As usual in data exchange, we assume that $\mathrm{dom}(I) \subseteq \mathit{Dom}$, where Dom is the union of two fixed disjoint infinite sets – the set Const of all *constants*, and the set Null of all *(labeled) nulls*. Constants are denoted by letters c, d, e and variants like c', c_1; different letters denote mutually distinct constants. Nulls serve as placeholders, or variables, for unknown constants; we will denote them by $\bot$ and variants like $\bot', \bot_1$. Instances without nulls are called *ground*. Let $\mathrm{const}(I) := \mathrm{dom}(I) \cap \mathit{Const}$ and $\mathrm{nulls}(I) := \mathrm{dom}(I) \cap \mathit{Null}$.

It will often be convenient to view instances as sets of atoms, where an *atom* is an expression of the form $R(\bar{a})$ with R a relation symbol and $\bar{a} \in \mathit{Dom}^{\mathrm{ar}(R)}$. Thus, we identify σ-instances I with the set $\{R(\bar{a}) \mid R \in \sigma, \bar{a} \in R^I\}$. This enables us to apply set theoretic notation to instances. For example, we may write $R(\bar{a}) \in I$ instead of "$R \in \sigma$ and $\bar{a} \in R^I$." Furthermore, we may write $I \subseteq J$ to indicate that all atoms of I are contained in J, or $I \cup J$ for the instance consisting of all the atoms of I and all the atoms of J.

Let I and J be instances. A *homomorphism* from I to J is a mapping $h\colon \mathrm{dom}(I) \to \mathrm{dom}(J)$ such that for all constants $c \in \mathrm{dom}(I)$ we have $h(c) = c$, and for all $R(a_1, \ldots, a_k) \in I$ we have $R(h(a_1), \ldots, h(a_k)) \in J$. We call J a *homomorphic image* of I if there is a

homomorphism h from I to J such that $J = h(I)$, where we define

$$h(I) := \{R(h(a_1), \ldots, h(a_k)) \mid R(a_1, \ldots, a_k) \in I\}.$$

Furthermore, we call I and J *homomorphically equivalent* if there is a homomorphism from I to J and a homomorphism from J to I. An *isomorphism* from I to J is a bijective homomorphism h from I to J such that h^{-1} is a homomorphism from J to I. If there is an isomorphism from I to J, we call I and J *isomorphic*. We say that J is a *core* of I if $J \subseteq I$ and there is a homomorphism from I to J, but no homomorphism from I to a proper subinstance of J. As shown in [19] (see also [13]), every instance has a core, and cores of homomorphically equivalent instances are isomorphic.

2.2 Queries and Constraints

The reader is assumed to be familiar with first-order logic (FO). Atomic FO-formulas over a schema σ are formulas of the form $R(u_1, \ldots, u_{\mathrm{ar}(R)})$ or $u_1 = u_2$, where $R \in \sigma$ and each u_i is a variable or an element of *Const*. Formulas of the first form are called *relation atoms* (over σ). FO-formulas over σ are built from atomic FO-formulas over σ in the usual way using negation, conjunction, disjunction, implication, existential quantification, and universal quantification. We write $\varphi(x_1, \ldots, x_k)$ to indicate that φ is a formula whose free variables are precisely $x_1, \ldots, x_k$; if φ is a sentence, we omit the parentheses.

Let $\mathrm{dom}(\varphi)$ be the set of all constants in φ. An assignment for φ in an instance I is a mapping α from the free variables of φ to $\mathrm{dom}(I) \cup \mathrm{dom}(\varphi)$, which we extend to *Const* via $\alpha(c) := c$ for all $c \in$ *Const*. We write $(I, \alpha) \models \varphi$ to indicate that φ is satisfied in I under α. The relation $\models$ is defined as usual, the only difference being that constants in φ are interpreted by themselves, and quantifiers range over $\mathrm{dom}(I) \cup \mathrm{dom}(\varphi)$. That is, we apply the *active domain semantics*. For example, we have $(I, \alpha) \models R(u_1, \ldots, u_{\mathrm{ar}(R)})$ precisely if $(\alpha(u_1), \ldots, \alpha(u_{\mathrm{ar}(R)})) \in R^I$; $(I, \alpha) \models u_1 = u_2$ precisely if $\alpha(u_1) = \alpha(u_2)$; and $(I, \alpha) \models \exists x\, \varphi$ precisely if there is an $a \in \mathrm{dom}(I) \cup \mathrm{dom}(\varphi)$ with $(I, \alpha[a/x]) \models \varphi$, where $\alpha[a/x]$ is the assignment defined like α, except that x is mapped to a. For an FO-formula $\varphi(x_1, \ldots, x_k)$ and a tuple $\bar{a} = (a_1, \ldots, a_k) \in (\mathrm{dom}(I) \cup \mathrm{dom}(\varphi))^k$, we write $I \models \varphi(\bar{a})$ instead of $(I, \alpha) \models \varphi$, where $\alpha(x_i) = a_i$ for each $i \in [k]$.

An FO-*query* over σ is an FO-formula φ over σ together with a tuple $\bar{x} = (x_1, \ldots, x_k)$ containing all the free variables in φ; we denote such queries by $\varphi(\bar{x})$. The *result* of $\varphi(\bar{x})$ on I is the set $\varphi(I) := \{\bar{a} \in (\mathrm{dom}(I) \cup \mathrm{dom}(\varphi))^k \mid I \models \varphi(\bar{a})\}$. We will often tacitly use the fact that for every FO-query Q over σ, there is a polynomial time algorithm that takes a σ-instance I as input and outputs $Q(I)$.

A *conjunctive query (CQ)* is an FO-query of the form $\varphi(\bar{x}) = \exists \bar{y}\, \psi$, where ψ is a conjunction of relation atoms. If ψ is a conjunction of relation atoms and inequalities $\neg u = v$, we call φ a *CQ with inequalities*, and if ψ is a conjunction of relation atoms and negated relation atoms, we call φ a *CQ with negation*. A *union of conjunctive queries (UCQ)* is a disjunction of CQs. A *UCQ with inequalities* is a disjunction of CQs with inequalities.

When we refer to the *atoms* of $\varphi(\bar{a})$ for some FO-formula $\varphi(\bar{x}) = R_1(\bar{y}_1) \wedge \cdots \wedge R_k(\bar{y}_k)$ and an assignment $\bar{a}$ for $\bar{x}$, we mean the atoms $R_i(\bar{b}_i)$, where $\bar{b}_i$ is obtained from $\bar{y}_i$ by replacing each variable in $\bar{y}_i$ with the corresponding value assigned to that variable by $\bar{a}$.

2.3 Schema Mappings

The following definitions are standard in data exchange (cf., e.g., [11, 13, 27, 6, 4]). A *schema mapping* $M = (\sigma, \tau, \Sigma)$ consists of disjoint schemas σ and τ, called *source schema*

and *target schema*, and a finite set Σ of assertions, where we distinguish between *source-to-target tuple-generating dependencies (st-tgds)*, *target tuple-generating dependencies (t-tgds)*, and *equality-generating dependencies (egds)*. *St-tgds* are FO-sentences of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, where φ is a conjunction of relation atoms over σ, ψ is a conjunction of relation atoms over τ, and φ and ψ contain no constants. *T-tgds* are defined similarly; they differ from st-tgds only in that φ, like ψ, is a conjunction of relation atoms over τ. *Egds* are FO-formulas of the form $\forall\bar{x}(\varphi(\bar{x}) \rightarrow y = z)$, where φ is a conjunction of relation atoms over τ, y and z occur in $\bar{x}$, and φ contains no constants.

A *source instance* S for M is a ground σ-instance. A *solution for S under M* is a τ-instance T such that $S \cup T$ satisfies all the tgds and egds in Σ.[2] Note that, unlike solutions, source instances are not allowed to contain nulls, and that a source instance may have no solution or more than one solution.

Concerning the question as to which solution should be materialized for data exchange, [11] proposes *universal solutions*, and makes a good case for materializing such solutions. A *universal solution* for S under M is a solution T for S under M such that for every solution T' for S under M there is a homomorphism from T to T'. A source instance for M might not have a universal solution, even if it has solutions. However, if M is specified by st-tgds, then every source instance S has a universal solution under M. For example, such a universal solution can be constructed from an initially empty instance over M's target schema by adding, for each st-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ in M and each pair $\bar{a},\bar{a}'$ of tuples with $S \models \varphi(\bar{a},\bar{a}')$ the atoms of $\psi(\bar{a},\bar{b})$, where $\bar{b}$ is a tuple of pairwise distinct fresh nulls. The resulting universal solution is unique up to isomorphism. We call it the *canonical solution* for S under M, and denote it by $\mathrm{CanSol}(M,S)$.

Particular important universal solutions are *core solutions*, which can be thought of as smallest universal solutions. If M is specified by st-tgds, then a *core solution* for S under M is defined as a core of $\mathrm{CanSol}(M,S)$. Since every two cores of $\mathrm{CanSol}(M,S)$ are isomorphic, there is a unique core solution for S under M up to isomorphism. Hence, we may speak of *the* core solution, denoted by $\mathrm{Core}(M,S)$. It is easy to verify that $\mathrm{Core}(M,S)$ is a solution for S under M. We will not need core solutions for more general schema mappings; see [13] for their definition and properties.

▶ **Example 2.** Consider the schema mapping $M = (\sigma,\tau,\Sigma)$ with σ consisting of a binary relation symbol *Book*, τ consisting of binary relation symbols *Author* and *BookInfo*, and Σ containing the st-tgd

$$\forall x\forall y\big(\mathit{Book}(x,y) \rightarrow \exists z(\mathit{Author}(y,z) \wedge \mathit{BookInfo}(z,x))\big).$$

Furthermore, consider the source instance

$$S := \{\mathit{Book}(\text{Comput. Compl.}, \text{S. Arora}), \mathit{Book}(\text{Comput. Compl.}, \text{B. Barak}),\\ \mathit{Book}(\text{Model Theory}, \text{W. Hodges})\},$$

which stores tuples of the form (book title, author) in Book^S. Then,

$$T := \{\mathit{Author}(\text{S. Arora}, \bot_1), \mathit{BookInfo}(\bot_1, \text{Comput. Compl.}),\\ \mathit{Author}(\text{B. Barak}, \bot_2), \mathit{BookInfo}(\bot_2, \text{Comput. Compl.}),\\ \mathit{Author}(\text{W. Hodges}, \bot_3), \mathit{BookInfo}(\bot_3, \text{Model Theory})\}$$

[2] A word of caution: In data exchange, solutions are usually finite, as introduced here, whereas in data integration, solutions may also be infinite. For simplicity, we consider only finite solutions.

is a universal solution for S under M. It is both the canonical solution and the core solution for S under M. Other universal solutions can be obtained from T by adding arbitrary tuples with nulls to T. Note that the instance T' obtained from T by identifying $\perp_1$ and $\perp_2$ is not a universal solution, since there is no homomorphism from T' to T. On the other hand, if we would add the egd $\forall x_1 \forall x_2 \forall y \big(\mathit{BookInfo}(x_1, y) \wedge \mathit{BookInfo}(x_2, y) \rightarrow x_1 = x_2 \big)$ to Σ, then T' would be the core solution for S under M. ◀

For later reference, we state:

▶ **Theorem 3** ([11, 13]). *Let M be a schema mapping defined by st-tgds. Then there are polynomial-time algorithms that take a source instance S for M as input and compute* $\mathrm{CanSol}(M, S)$ *and* $\mathrm{Core}(M, S)$*, respectively.*

3 The Certain Answers Semantics and Non-Monotone Queries

The basic approach for answering a query Q over the target schema of a schema mapping $M = (\sigma, \tau, \Sigma)$ is to return its *certain answers* [29, 11, 27, 6, 4]. Given any source instance S for M, the *certain answers* to Q on M and S are defined as

$$\mathit{cert}(Q, M, S) \;:=\; \{\bar{a} \mid \bar{a} \in Q(T) \text{ for all solutions } T \text{ for } M \text{ under } S\}.$$

So, informally, a tuple $\bar{a}$ belongs to $\mathit{cert}(Q, M, S)$ whenever it is an answer to Q no matter on which of S's solutions Q is evaluated. Note that if Q is a UCQ (or any other domain independent query), this means that $Q(\bar{a})$ logically follows from S, viewed as a set of atomic formulas, and Σ.[3]

▶ **Example 4.** Let M and S be as in Example 2, and consider the UCQ

$$Q(x) \;:=\; \exists y \, \big(\mathit{BookInfo}(y, \text{Comput. Compl.}) \wedge \mathit{Author}(x, y) \big),$$

which asks for all authors of "Comput. Compl." It is intuitively clear that the result of evaluating Q with respect to M and S should be "S. Arora" and "B. Barak." And indeed, $\mathit{cert}(Q, M, S) = \{\text{S. Arora}, \text{B. Barak}\}$. ◀

The *certain answers* have several good properties for query answering in data exchange and data integration. The most apparent one is their simple and natural definition. Another one is that in many practical settings it is possible to compute them in polynomial time (for fixed schema mappings and queries) from a single solution, namely a universal solution [11, 5], or by evaluating a suitably rewritten query over the source schema (cf., [29] and Chapter 5 of this book). For example, to compute $\mathit{cert}(Q, M, S)$ for a UCQ Q, we only need to evaluate Q on an arbitrary universal solution for S, and remove all tuples with nulls from its result:

▶ **Theorem 5** ([11]). *Let M be a schema mapping, let S be a source instance for M, and let Q be a UCQ over M's target schema. For any universal solution T for S under M,*

$$\mathit{cert}(Q, M, S) \;=\; \{\bar{c} \in Q(T) \mid \bar{c} \textit{ contains no nulls}\}.$$

[3] Here we use the standard first-order semantics. That is, $Q(\bar{a})$ logically follows from S and Σ if for all instances I with $S \subseteq I$ and $I \models \Sigma$ we have $I \models Q(\bar{a})$.

In particular, if M and Q are fixed, and M is such that for any source instance S for M, S has a universal solution if it has a solution, and a universal solution for S can be computed in polynomial time, then $\mathit{cert}(Q, M, S)$ can be computed in polynomial time. By Theorem 3, schema mappings defined by st-tgds have this property. Much broader classes of schema mappings with this property are known, see, e.g., [11, 13, 16, 9, 28, 34, 15, 18] and Chapter 1 of this book.

▶ Remark. Extensions of universal solutions and Theorem 5 that are suitable for answering general monotone queries like UCQs with inequalities appeared in [9]. However, computing the certain answers to such queries (for fixed schema mappings and queries) is in co-NP, and co-NP-complete in general [11, 33]. Certain fragments of UCQs with inequalities were shown to be tractable, though [11, 5].

Despite their good properties, it has been realized that for *non-monotone* queries the certain answers may yield counter-intuitive results. We have illustrated the basic problem in Example 1. Other problems have been pointed out in [11, 3, 31], for example:

- A *copying schema mapping* is a schema mapping $M = (\sigma, \tau, \Sigma)$, where τ consists of copies R' for each $R \in \sigma$, and Σ consists of st-tgds $\forall \bar{x}\big(R(\bar{x}) \to R'(\bar{x})\big)$ for each $R \in \sigma$. For example, the schema mapping from Example 1 is a copying schema mapping. Although copying schema mappings intuitively say nothing else than to copy each relation R to the relation R', [3] showed that there is a copying schema mapping $M = (\sigma, \tau, \Sigma)$ and a simple FO-query Q (actually, a union of a CQ and a CQ with negation) that is not rewritable to an FO-query Q' over σ such that for every source instance S for M, $Q'(S) = \mathit{cert}(Q, M, S)$. They also proved that it is not rewritable to an FO-query Q' over τ such that for every source instance S for M, $Q'(T) = \mathit{cert}(Q, M, S)$ with $T \in \{\mathrm{Core}(M, S), \mathrm{CanSol}(M, S)\}$.
- As shown in [3], if M is a schema mapping defined by st-tgds, then for every Boolean FO-query Q over M's target schema, either $\mathit{cert}(Q, M, S) = \emptyset$ for all source instances S for M, or $\mathit{cert}(\neg Q, M, S) = \emptyset$ for all source instances S for M. To see this, suppose that $\mathit{cert}(Q, M, S) \neq \emptyset$ for some source instance S for M. Then for all solutions T for S under M we have $T \models Q$. Now, if S' is an arbitrary source instance for M, it is not hard to see that there is a solution T' for S' under M that is also a solution for S under M. Then, $T' \not\models \neg Q$, and therefore $\mathit{cert}(\neg Q, S', M) = \emptyset$. It follows that either Q or $\neg Q$ has a trivial answer, namely $\emptyset$, that does not depend on the source instance.

As already pointed out in the introduction, the problem is really a matter of the semantics of schema mappings, that is, a matter of which target instances of a schema mapping M are considered as solutions for a source instance under M. One way to enforce a suitable set of solutions would be to use a more expressive constraint language for specifying schema mappings. This approach is certainly worth pursuing, but to the best of my knowledge it seems to have received almost no attention in the literature to date. The approaches proposed in the literature are based on the certain answers to queries with respect to a restricted set of solutions.

4 Semantics for Non-Monotone Queries

A variety of semantics for answering non-monotone queries over the target schema of a schema mapping have been proposed in the literature [13, 22, 32, 2, 21]. These semantics are based on the following basic idea: for each schema mapping M and each source instance S, define an appropriate set $[\![M, S]\!]$ of solutions, and answer queries Q by the certain answers to Q on $[\![M, S]\!]$, that is, $\{\bar{a} \mid \bar{a} \in Q(T) \text{ for all } T \in [\![M, S]\!]\}$.

In [13], it is proposed to let $[\![M, S]\!]$ be the set of all universal solutions for S under M. However, the resulting semantics has similar problems as the certain answers semantics [3].[4] Therefore, we will not consider that semantics here.

The semantics proposed in [22, 32, 2, 21] are based on *non-monotonic reasoning*, specifically variants of the *Closed World Assumption (CWA)* [36], to arrive at the sets $[\![M, S]\!]$. This means that a solution will be in $[\![M, S]\!]$ if it can be derived in a certain way from M and S. The goal is always to define a set of solutions that intuitively captures "precisely the positive information in M and S, and nothing more." This is quite natural, since – as Libkin argued in [31] – in data exchange (but the same applies to data integration), data is moved from source to target according to the tgds and egds of a schema mapping. Therefore, answers to queries should only depend on that data, and not on data that could later be added to the target database. For instance, this seems to be natural in Example 1 as we have argued there. In the following, we review these semantics in more detail.

4.1 Libkin's CWA-Semantics

The CWA-semantics [31, 23] (see also [22]) was the first semantics explicitly designed for answering non-monotone queries in data exchange. It was introduced by Libkin [31] for schema mappings defined by st-tgds, and extended by Schweikardt and myself [23] to schema mappings as considered in this chapter. As the name suggests, it[5] is based on the CWA. As mentioned above, this means that for answering a query Q on M and S we take into account only those solutions for S under M which can be derived in a certain way from M and S. We call such solutions *CWA-solutions*.

4.1.1 CWA-Solutions

Informally, CWA-solutions for a source instance S under a schema mapping M are all those solutions T for S under M that satisfy the following properties:

1. All atoms in T are justified in a certain sense by M and S.
2. Each justification for atoms is used at most once.
3. Each "positive statement" (Boolean conjunctive query) that is true in T logically follows from S and the set of tgds and egds in M. That is, T should not "invent" new facts compared to what can be inferred from S using M.

Below, we give an idea of how to formalize these informal requirements.

In the context of schema mappings defined by st-tgds, the requirements can be formalized as follows. Assume that M is defined by st-tgds. Regarding the first requirement, an atom is *justified* if it can be obtained from S by means of "applying" an st-tgd in M to S, where st-tgds are considered as rules for deriving new atoms, similar to Datalog rules. A *justification for atoms* consists of an st-tgd θ in M, say $\theta = \forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, and assignments $\bar{a}, \bar{a}'$ to $\bar{x}, \bar{y}$ such that $S \models \varphi(\bar{a}, \bar{a}')$. We denote it by $(\theta, \bar{a}, \bar{a}')$. An atom in T is justified if there is a justification $(\theta, \bar{a}, \bar{a}')$ with $\theta = \forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, and an assignment $\bar{b}$ for $\bar{z}$ such that $T \models \psi(\bar{a}, \bar{b})$, and the atom is one of the relation atoms in $\psi(\bar{a}, \bar{b})$. The second requirement insists that each justification j is "used" with a unique assignment $\bar{b}_j$ for the existentially quantified variables of the st-tgd in j. An atom in T is then justified if there is a justification j such that the atom is justified by j as above, except that the assignment $\bar{b}$

[4] Example 1 is true for the universal solution-based semantics, too.

[5] As we shall see below, the CWA-semantics form a family of semantics. But for the moment, let us refer to this family as the CWA-semantics.

must be the assignment $\bar{b}_j$. It was shown in [31, 22] that a solution T for S under M satisfies the two requirements precisely if T is a homomorphic image of $\mathrm{CanSol}(M, S)$. Furthermore, the third requirement, once properly formalized, turns out to be equivalent to the property of being a *universal solution*. Hence:

▶ **Theorem 6** ([31, 22]). *Let M be a schema mapping defined by st-tgds, and let S be a source instance for M. A solution T for S under M is a CWA-solution for S under M iff*

1. *T is a homomorphic image of $\mathrm{CanSol}(M, S)$, and*
2. *T is a universal solution for S under M*

(or, equivalently, there is a homomorphism from T to $\mathrm{CanSol}(M, S)$).

This characterization immediately implies that $\mathrm{CanSol}(M, S)$ is the "maximal" CWA-solution for S under M up to isomorphism in the sense that $\mathrm{CanSol}(M, S)$ is a CWA-solution for S, and that every CWA-solution for S is a homomorphic image of $\mathrm{CanSol}(M, S)$. Furthermore, it was shown in [31, 22] that $\mathrm{Core}(M, S)$ is the unique "smallest" CWA-solution for S under M up to isomorphism.

▶ **Example 7.** Let M be the schema mapping defined by the st-tgd θ from Example 1, and let S be the source instance exhibited in the same example. Then there is a unique justification consisting of θ, and assigning x, y the values c, d. Since θ has no existentially quantified variables, the only atom that can be justified using this justification is $E'(c, d)$. Hence, $T := \{E'(c, d)\}$ is the unique CWA-solution for S under M. Indeed, we have $T = \mathrm{Core}(M, S) = \mathrm{CanSol}(M, S)$. ◀

▶ **Example 8.** Let M, S and T be as in Example 2. In the same example, we mentioned that $T = \mathrm{CanSol}(M, S) = \mathrm{Core}(M, S)$. Hence, T is the unique CWA-solution for S under M up to isomorphism. ◀

To lift Libkin's CWA-semantics to schema mappings defined by st-tgds, t-tgds and egds, it is necessary to formalize the first two requirements above for such schema mappings. In [23, 22], this is done using a derivation-based approach using a suitably controlled version of the *chase procedure*. In addition, [24] (see also [20]) introduces a *2-player game* and characterizes the requirements using this game. It is shown that CWA-solutions can still be characterized as particular universal solutions, and that the core solution, if it exists, is the "smallest" CWA-solution up to isomorphism. On the other hand, in general there is no "maximal" CWA-solution, that is, a CWA-solution with the same properties as the canonical solution in the context of schema mappings defined by st-tgds.

Concerning the question whether a given source instance has a CWA-solution, it is easy to see that a source instance has a CWA-solution whenever it has a universal solution. If M is a schema mapping defined by st-tgds, by Theorem 3, we can even compute CWA-solutions in polynomial time. For most of the classes of schema mappings mentioned in Section 3, for which universal solutions can be computed in polynomial time (data complexity), it is possible to compute CWA-solutions in polynomial time (data complexity). However:

▶ **Theorem 9** ([23, 22]). *There is a schema mapping $M = (\sigma, \tau, \Sigma)$ with Σ consisting only of st-tgds and t-tgds such that the following problem is undecidable: Given a source instance S for M, is there a CWA-solution for S under M?*

4.1.2 Query Answering under the CWA

Given a schema mapping M and a source instance S for M, it seems now perfectly reasonable to answer queries Q over M's target schema by the certain answers to Q on the CWA-solutions

for S under M, that is, by the set of all tuples that are answers to Q on all CWA-solution for S under M. However, we should be careful about what constitutes the result of a query on an individual CWA-solution. To explain why, we need a little background on incomplete instances.

An *incomplete σ-instance* is a set $\mathcal{I}$ of ground σ-instances [1, 37]. The idea is that $\mathcal{I}$ represents an unknown instance I, and the instances in $\mathcal{I}$ are the possibilities for I. Every σ-instance I represents an incomplete σ-instance. This is because nulls, which may occur in I, are place-holders for unknown constants, and therefore, any instance obtained from I by substituting constants for the nulls in I is a ground instance that could possibly be represented by I. Consequently, I represents the incomplete σ-instance

$$\mathit{rep}(I) := \{h(I) \mid h\colon \mathrm{dom}(I) \to \mathit{Const},\ h \text{ is the identity on } \mathrm{const}(I)\}\,.$$

Here we are more interested in incomplete instances represented by CWA-solutions. Technically, CWA-solutions are instances I together with a set Σ of integrity constraints (the set of t-tgds and egds of the schema mapping). Several ways of associating an incomplete instance with such an instance have been proposed (see, e.g., [1, 37]). We choose the one proposed in [26, 37], which is

$$\mathit{rep}_\Sigma(I) := \{J \mid J \in \mathit{rep}(I),\ J \models \Sigma\}\,.$$

Now, if I is an instance with nulls, and Q is a non-monotone query, returning $Q(I)$ as the answer to Q on I may lead to counter-intuitive results [26], mainly due to the fact that distinct nulls may represent the same constant. To circumvent this, one typically uses semantics designed for answering queries on incomplete instances. There are several such semantics, but the most common one is the *certain answers semantics* [1, 37]. The *certain answers* to a query Q on an incomplete instance $\mathcal{I}$ are defined by:

$$\mathit{cert}(Q,\mathcal{I}) := \{\bar{a} \mid \bar{a} \in Q(I) \text{ for all } I \in \mathcal{I}\}\,.$$

For an instance I and a set Σ of constraints, we let

$$\mathit{cert}(Q,I) := \mathit{cert}(Q,\mathit{rep}(I)) \quad \text{and} \quad \mathit{cert}_\Sigma(Q,I) := \mathit{cert}(Q,\mathit{rep}_\Sigma(I)).$$

▶ Remark. It is no coincidence – and will do no harm – that we use the same name both for the certain answers with respect to schema mappings and source instances, and for the certain answers with respect to incomplete instances. Indeed, the set of solutions for a source instance S under a schema mapping M is almost an incomplete instance $\mathcal{T}$, except that it may contain non-ground instances (think of each solution in $\mathcal{T}$ as a possible outcome of translating S to the target).

For answering queries over target schemas of schema mappings, [22] propose the following variant of the certain answers on CWA-solutions:[6]

▶ **Definition 10.** Let $M = (\sigma, \tau, \Sigma_{\mathrm{st}} \cup \Sigma_{\mathrm{t}})$ be a schema mapping, where Σ_{st} is a set of st-tgds and Σ_{t} is a set of t-tgds and egds, let S be a source instance for M, and let Q be a query over τ. Then the set of the *CWA-answers to Q on M and S* is defined as

$$\mathit{cert}_{\mathrm{CWA}}(Q,M,S) := \{\bar{a} \mid \bar{a} \in \mathit{cert}_{\Sigma_{\mathrm{t}}}(Q,T) \text{ for all CWA-solutions } T \text{ for } S \text{ under } M\}.$$

[6] Three more semantics have been proposed in [22]. For brevity, we consider only the most basic one.

The following example shows that for the schema mapping, source instance and query in Example 1, the CWA-answers leads to the desired result.

▶ **Example 11.** Let M and S be as in Example 7. As shown in Example 7, $T := \{E'(c, d)\}$ is the unique CWA-solution for S under M. Note that $rep(T) = \{T\}$, since T contains no nulls. In particular, for the query $Q(x, y)$ from Example 1, we have $cert(Q, T) = \{(c, d)\}$, and therefore $cert_{\mathrm{CWA}}(Q, M, S) = \{(c, d)\}$, as desired. ◀

More generally, if M is a copying schema mapping, then every source instance S for M has a unique CWA-solution S', namely its copy, and the CWA-answers to a query Q on M and S are precisely $Q(S')$, as desired. This implies that Q can be trivially rewritten into a query Q' over M's source schema such that for all source instances S we have $Q'(S) = cert_{\mathrm{CWA}}(Q, M, S)$. Hence, the CWA-semantics remedies the problems of the certain answers semantics on non-monotone queries described at the end of Section 3.

Let us finally consider an example that involves st-tgds with existential quantifiers:

▶ **Example 12.** Let M, S and T be as in Example 2. As shown in Example 8, T is the unique CWA-solution up to isomorphism. Consider the query

$$Q(t) \;:=\; \exists^{=1} a \, \exists x \, \big(\mathit{BookInfo}(x, t) \wedge \mathit{Author}(a, x)\big),$$

which, intuitively, asks for all single-authored books. However, $cert_{\mathrm{CWA}}(Q, M, S) = \emptyset$, since "Comput. Compl." cannot be in $cert_{\mathrm{CWA}}(Q, M, S)$, and $rep(T)$ contains an instance obtained from T by replacing $\bot_1, \bot_2, \bot_3$ with the same constant. On the other hand, this is not really surprising, since M does not tell us that $\bot_1, \bot_2, \bot_3$ could not represent the same value.

Now let M' be the extension of M by the egd

$$\eta \;:=\; \forall x \forall y_1 \forall y_2 \big(\mathit{BookInfo}(x, y_1) \wedge \mathit{BookInfo}(x, y_2) \to y_1 = y_2\big).$$

Then, T would still be the unique CWA-solution for S under M', but $cert(Q, M', S) = \{\text{Model Theory}\}$, since $rep_{\{\eta\}}(T)$ does not contain any instance that arises from T by mapping $\bot_3$ to the same constant as $\bot_1$ or $\bot_2$. ◀

For a number of schema mappings, including schema mappings defined by st-tgds, the task of computing the CWA-answers to a query can be reduced – as in the case of the certain answers semantics, explained in Section 3 – to the task of evaluating the query on a single incomplete instance, which is a well-studied topic, see, e.g., [1, 37]:

▶ **Theorem 13** ([22]). *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping where Σ is a set of st-tgds, let S be a source instance for M, and let Q be a query over τ. Then:*

$$cert_{\mathrm{CWA}}(Q, M, S) \;=\; cert(Q, \mathrm{CanSol}(M, S)).$$

The result also holds for schema mappings defined by st-tgds and egds with $\mathrm{CanSol}(M, S)$ extended appropriately.

4.2 A Relaxation of the CWA-Semantics

The CWA interprets existential quantifiers in tgds in a very restrictive way. For instance, in Example 8, each entry in Book^S introduces precisely one null which corresponds to a new value assigned to the variable z in the unique st-tgd in M. In some cases, like Example 2, this might be desirable. But in other cases, this might be too restrictive.

▶ **Example 14** ([22]). Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, with σ containing a unary relation symbol *Person*, τ containing a binary relation symbol *Child*, and Σ containing a single st-tgd

$$\theta := \forall x\big(\mathit{Person}(x) \to \exists y\, \mathit{Child}(x, y)\big).$$

Intuitively, θ states that for each person x there is a child y. Certainly, there could be more than one child. However, under the CWA, the effect of θ would be that each person x has *exactly* one child y. Indeed, if S is a source instance for M, then there is a unique CWA-solution T for S under M which assigns to each person $p \in \mathit{Person}^S$ a unique null $\bot_p$ such that $(p, \bot_p) \in \mathit{Child}^T$. Hence, the CWA-answers to the Boolean query $\forall x \exists^{=1} y\, \mathit{Child}(x, y)$ on M and S would yield true (i.e., a non-empty result), even though this was not intended. ◀

Libkin and Sirangelo [32] propose a relaxation of the CWA, which admits finer control over the degree of "closedness." The basic idea is to control for each position in a solution whether it should be *open* for adding new values at this position, or *closed*. In the following, we illustrate the basic idea with an example.

▶ **Example 15** (Example 14, continued). Let us annotate each occurrence of a variable in the head[7] of θ as *closed* (cl) or *open* (op) as follows:

$$\forall x\big(\mathit{Person}(x) \to \exists y\, \mathit{Child}(x^{\mathrm{cl}}, y^{\mathrm{op}})\big).$$

Then the annotated version of θ induces the following annotated version of $\mathrm{CanSol}(M, S)$ for the source instance S with $\mathit{Person}^S = \{p_1, p_2\}$:

$$T = \{\mathit{Child}(p_1^{\mathrm{cl}}, \bot_1^{\mathrm{op}}), \mathit{Child}(p_2^{\mathrm{cl}}, \bot_2^{\mathrm{op}})\}.$$

The basic idea is that at a position annotated with *op* we may "insert" arbitrary many values, while at a position annotated with *cl*, the value is fixed. That is, the atom $\mathit{Child}(p_i^{\mathrm{cl}}, \bot_i^{\mathrm{op}})$ corresponds to "there exist one or more c with $\mathit{Child}(p_i, c)$," as desired. ◀

More generally, let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds. The starting point is always an annotation α of the positions in the heads of the st-tgds in M, which must be provided by the user. To be precise, a *position* in the head of an st-tgd

$$\theta := \forall \bar{x} \forall \bar{y}\, \big(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z} \bigwedge_{i=1}^{k} R_i(\bar{u}_i)\big)$$

is represented by a pair (i, j), where $i \in [k]$ and $j \in [\mathrm{ar}(R_i)]$. Such a pair corresponds to the variable at position j in $\bar{u}_i$. Then for each st-tgd $\theta \in \Sigma$ and each position (i, j) in θ's head, we have an annotation $\alpha(\theta, i, j) \in \{\mathrm{cl}, \mathrm{op}\}$. For instance, the annotation of the st-tgd in Example 15 corresponds to $\alpha(\theta, 1, 1) = \mathrm{cl}$ and $\alpha(\theta, 1, 2) = \mathrm{op}$.

Given a source instance S, we now define the annotated canonical solution $\mathrm{CanSol}_\alpha(M, S)$ for S under M and α, which is a set of pairs $(R(u_1, \ldots, u_k), \alpha')$ consisting of an atom $R(u_1, \ldots, u_k)$ from $\mathrm{CanSol}(M, S)$, and an annotation $\alpha' \colon [k] \to \{\mathrm{cl}, \mathrm{op}\}$. The construction is as indicated in Example 15: starting from an empty set, for each st-tgd $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ and each assignment $\bar{a}, \bar{a}'$ for $\bar{x}, \bar{y}$ with $S \models \varphi(\bar{a}, \bar{a}')$, we pick a tuple $\bar{b}$ of pairwise distinct fresh nulls and add all pairs (A, α') to the set with A an atom of $\psi(\bar{a}, \bar{b})$, and α' the annotation induced by α on A.

[7] Given a tgd θ of the form $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, we refer to the formula $\psi(\bar{x}, \bar{z})$ as the head of θ.

As in the case of $\mathrm{CanSol}(M,S)$, the annotated canonical solution $\mathrm{CanSol}_\alpha(M,S)$ represents an incomplete instance, denoted by $\mathit{rep}(\mathrm{CanSol}_\alpha(M,S))$, which is then used for query answering. Before we give its definition, let us continue with our example.

▶ **Example 16** (Example 14, continued). Let α be the annotation of θ's head as shown in Example 15. Recall $T = \mathrm{CanSol}_\alpha(M,S)$ from the same example, and that the meaning of each atom $\mathit{Child}(p_i^{\mathrm{cl}}, \bot_i^{\mathrm{op}})$ in T is "there exist one or more c with $\mathit{Child}(p_i,c)$." Hence, T represents the incomplete instance $\mathit{rep}(T)$ which consists of all τ-instances T' such that $\mathit{Child}^{T'}$ contains tuples (p_1,a) and (p_2,b) with possibly identical constants a,b, and all tuples in $\mathit{Child}^{T'}$ have the form (p_i,c) with $i \in [2]$ and $c \in \mathit{Const}$. In particular, each $T' \in \mathit{rep}(T)$ contains tuples (p_1,a) and (p_2,b), which represents the information that for each $i \in [2]$ there is at least one c with (p_i,c). Furthermore, for any such tuple we can add a new tuple by replacing the values at the positions annotated as "open" with new values. The resulting instance would be in $\mathit{rep}(T)$, too. ◀

In general, $\mathit{rep}(\mathrm{CanSol}_\alpha(M,S))$ consists of all ground instances T such that there is a homomorphism h from $\mathrm{CanSol}(M,S)$ to T with $h(\mathrm{CanSol}(M,S)) \subseteq T$, and for each atom $R(a_1,\dots,a_k) \in T \setminus h(\mathrm{CanSol}(M,S))$ there is a pair $(R(b_1,\dots,b_k),\alpha') \in \mathrm{CanSol}_\alpha(M,S)$ such that $a_i = h(b_i)$ for all positions $i \in [k]$ with $\alpha'(i) = \mathrm{cl}$. That is, each atom A in $T \setminus h(\mathrm{CanSol}(M,S))$ coincides with an atom from $h(\mathrm{CanSol}(M,S))$ on all positions that are annotated as "closed"; A may have arbitrary values at positions annotated as "open".

Analogous to Theorem 13, which characterizes the certain answers to Q on M and S by $\mathit{cert}(Q, \mathit{rep}(\mathrm{CanSol}(M,S)))$, Libkin and Sirangelo propose to answer Q on (M,α) and S by[8]

$$\mathit{cert}(Q,M,\alpha,S) := \mathit{cert}\big(Q, \mathit{rep}(\mathrm{CanSol}_\alpha(M,S))\big).$$

Note that under this semantics, the answers to a query depend on the annotation α. Note also that for the annotation that assigns to each position in the head of a st-tgd the label *cl*, we obtain the CWA-semantics. The other extreme is to assign to each position the label *op*. In this case, we arrive at the certain answers semantics as introduced in Section 3. For details, the reader is referred to [32].

▶ **Example 17** (Example 14, continued). Let α be the annotation of θ's head as shown in Example 15. For the Boolean query

$$Q := \forall x \exists^{=1} y\, \mathit{Child}(x,y)$$

from Example 14 we have $\mathit{cert}(Q,M,\alpha,S) = \emptyset$, as desired. ◀

Libkin and Sirangelo introduced their "mixed world" semantics for schema mappings defined by st-tgds only, and left open the task of extending it to more general schema mappings.

▶ Remark. Afrati and Kolaitis [2] proposed a much stricter version of the CWA, and argued that it leads to an interesting semantics for *aggregate queries* under schema mappings defined by st-tgds. Recall that for such schema mappings M, if S is a source instance and Q is a query over M's target schema, the CWA-answers to Q on M and S can be characterized as $\mathit{cert}(Q, \mathrm{CanSol}(M,S))$. That is, a tuple belongs to the set of CWA-answers to Q on M and S iff it belongs to $Q(T)$ for all $T \in \mathit{rep}(\mathrm{CanSol}(M,S))$. Afrati and Kolaitis argue that

[8] In fact, this is a characterization of their semantics, in the same way as Theorem 13 is a characterization of the CWA-semantics.

the CWA-semantics is too weak in the context of aggregate queries, since $\mathit{rep}(\mathrm{CanSol}(M,S))$ may contain instances with values that do not occur in S, and propose a new semantics based on the set of all *endomorphic images* of $\mathrm{CanSol}(M,S)$. Here, an instance I is an *endomorphic image* of an instance J if there is a homomorphism h from J to J such that $h(J) = I$. Given a query Q over M's target schema, we could now answer Q on M and S under this "endomorphic images semantics" by

$$\mathit{cert}_{\mathrm{endo}}(Q,M,S) \;:=\; \{\bar{a} \mid \bar{a} \in Q(T) \text{ for all endomorphic images } T \text{ of } \mathrm{CanSol}(M,S)\}.$$

4.3 The GCWA*-Semantics

Consider two schema mappings $M_1 = (\sigma,\tau,\Sigma_1)$ and $M_2 = (\sigma,\tau,\Sigma_2)$ over the same source schema σ and the same target schema τ. We say that M_1 and M_2 are *logically equivalent* if Σ_1 and Σ_2 are logically equivalent under the standard FO-semantics[9] (i.e., for every $\sigma\cup\tau$-instance I we have $I \models \Sigma_1$ if and only if $I \models \Sigma_2$). If M_1 and M_2 are logically equivalent, it seems desirable that for each source instance S for M_1 (resp., M_2) and each query Q over τ, the answer to Q on M_1 and S is the same as the answer to Q on M_2 and S, since intuitively M_1 and M_2 specify the same translation of source data to the target. Yet, for the semantics introduced above, this is not necessarily true:

▶ **Example 18.** Let $M_1 = (\sigma,\tau,\Sigma_1)$ and $M_2 = (\sigma,\tau,\Sigma_2)$ with $\sigma = \{P\}$, $\tau = \{E\}$, and

$$\begin{aligned}\Sigma_1 \;&=\; \big\{\forall x\big(P(x) \to E(x,x)\big)\big\}\,,\\ \Sigma_2 \;&=\; \Sigma_1 \cup \big\{\forall x\big(P(x) \to \exists y\, E(x,y)\big)\big\}\,.\end{aligned}$$

Clearly, M_1 and M_2 are logically equivalent. Now, for $S := \{P(c)\}$ we have

$$\begin{aligned}T_1 \;&:=\; \mathrm{CanSol}(M_1,S) = \{E(c,c)\},\\ T_2 \;&:=\; \mathrm{CanSol}(M_2,S) = \{E(c,c), E(c,\bot)\}.\end{aligned}$$

Hence, if

$$Q(x) \;:=\; \exists^{=1} y\, E(x,y),$$

then $\mathit{cert}_{\mathrm{CWA}}(Q,M_1,S) = \{c\}$, since T_1 is the unique CWA-solution for S under M_1, while $\mathit{cert}_{\mathrm{CWA}}(Q,M_2,S) = \emptyset$, since T_2 is a CWA-solution for S under M_2. Analogously, we have $\mathit{cert}_{\mathrm{endo}}(Q,M_1,S) = \{c\}$ and $\mathit{cert}_{\mathrm{endo}}(Q,M_2,S) = \emptyset$.

Next we turn to the "mixed world" semantics. Fix an annotation α for the st-tgds in Σ_2. In particular, α yields an annotation for the st-tgd in Σ_1. As long as the second position in the head of the st-tgd in Σ_1 is annotated as *closed* by α, we have $\mathit{cert}(Q,M_1,\alpha,S) = \{c\}$ and $\mathit{cert}(Q,M_2,\alpha,S) = \emptyset$. Note that if the second position is annotated as *open*, the resulting semantics is very close to the certain answers semantics. ◀

There is a second issue related to the interpretation of tgds under the CWA-semantics, which is best illustrated with an example:

[9] Different notions of equivalence between schema mappings have been considered in [12]. Logical equivalence is the strongest such notion. Instead of invariance under logical equivalence one could also require invariance under any of the other notions of schema mapping equivalence.

▶ **Example 19.** Let $M = (\sigma, \tau, \Sigma)$ be defined by $\sigma = \{R\}$, $\tau = \{E, F\}$, and $\Sigma = \{\theta\}$, where

$$\theta \;=\; \forall x, y\Big(R(x,y) \to \exists z\,\big(E(x,z) \wedge F(z,y)\big)\Big).$$

Intuitively, θ states that "if $R(x,y)$, then there is at least one z such that $E(x,z)$ and $F(z,y)$ hold." There could be exactly one such z, but there could also be more than one such z. In particular, the possibility that there are precisely two such z, or precisely three such z etc. is perfectly consistent with θ, and should not be denied when answering queries. Hence, given a source instance S for M, we should expect that the answer to

$$Q(x,y) \;:=\; \exists^{=1} z\,\big(E(x,z) \wedge F(z,y)\big)$$

on M and S is empty.

However, if we consider the source instance $S = \{R(c,d)\}$, we have $\mathrm{CanSol}(M,S) = \{E(c,\bot), F(\bot,d)\}$, so that $\mathit{cert}_{\mathrm{CWA}}(Q,M,S) = \mathit{cert}_{\mathrm{endo}}(Q,M,S) = \{(c,d)\}$. Hence, both the CWA-semantics and the endomorphic images semantics exclude the possibility that there is more than one z satisfying $E(x,z)$ and $F(z,y)$, although θ explicitly states that it is possible that more than one such z exists.

Note that the existential quantifier in θ can be expressed via an infinite disjunction over all possible choices of values for z (recall that nulls are just place-holders for unknown constants, so we do not have to consider nulls here):

$$\theta' \;:=\; \forall x, y\Big(R(x,y) \to \bigvee_{c \in \mathit{Const}} \big(E(x,c) \wedge F(c,y)\big)\Big).$$

Thus, we argued above that we need a semantics that interprets this disjunction *inclusively* rather than exclusively. ◀

Under the "mixed world" semantics, the query Q in Example 19 is answered as expected as long as the occurrences of z in θ are annotated as *open*. On the other hand, if the occurrences of z in θ are annotated as *open*, then, in a way, the "mixed world" semantics is "too open" in that it allows atoms to appear in solutions that intuitively cannot be justified by the source instance and the st-tgds.

▶ **Example 20** (Example 19, continued)**.** In light of the rewriting θ' of θ in Example 19 the only reasonable solutions for $S = \{R(c,d)\}$ under M seem to be those solutions T for which there is a finite set $X \subseteq \mathit{Dom}$ such that $T = \{E(c,x) \mid x \in X\} \cup \{F(x,d) \mid x \in X\}$. For example,

$$T^* \;:=\; \{E(c,e), E(c,e'), F(e,d)\}$$

should not be a valid solution, since the occurrence of $E(c,e')$ in T can intuitively not be explained in terms of S and θ (for this, $F(e',d)$ should be in T). Therefore, we should expect the answer to

$$Q'(x) \;:=\; \forall z\big(E(x,z) \to \exists y\, F(z,y)\big)$$

on M and S to be $\{c\}$. But let α be an annotation for θ that annotates the second position of $E(x,z)$ in θ as *open*. Then, $\mathit{cert}(Q', M, \alpha, S) = \emptyset$ since $T^* \in \mathit{rep}(\mathrm{CanSol}_\alpha(M,S))$. Intuitively, the "mixed world" semantics is "too open" in that it allows $E(c,e')$ to occur in T^* without enforcing that the corresponding atom $F(e',d)$ is present in T^*. ◀

Motivated by the above examples, [21] proposes a new semantics, called *GCWA*-semantics*. This semantics is invariant under logically equivalent schema mappings, and interprets existential quantifiers in a natural way. A detailed discussion of the latter property can be found in [21]. The starting point for the development of the GCWA*-semantics was the observation that query answering with respect to schema mappings is very similar to query answering on *deductive databases* [14], and that non-monotone query answering on deductive databases is a well-studied topic in this area (see, e.g., [14, 36, 35, 38, 25, 8]). Therefore, it seemed obvious to use these semantics in the context of data exchange. For data exchange, the semantics based on Reiter's formalization of the CWA [36], and variants of the CWA like Minker's *generalized CWA (GCWA)* [35] seemed to be particularly interesting. It turns out, though, that these semantics are too strong, too weak, or do not have the desired properties. Nevertheless, their analysis provided a good starting point for developing the GCWA*-semantics. For details, see [21].

For schema mappings defined by st-tgds and egds, the GCWA*-semantics has a very simple definition in terms of *minimal solutions*. Here, a solution T for S under M is *minimal* if there is no solution T' for S under M with $T' \subsetneq T$.

▶ **Definition 21.** Let M be a schema mapping defined by st-tgds and egds, and let S be a source instance for M.

1. A *GCWA*-solution for S under M* is a ground solution that is the union of minimal solutions for S under M.
2. Given a query Q over M's target schema, the set of all *GCWA*-answers to Q on M and S* is defined by

$$cert_{\text{GCWA}^*}(Q, M, S) \;:=\; \{\bar{a} \mid \bar{a} \in Q(T) \text{ for all GCWA}^*\text{-solutions for } S \text{ under } M\}\,.$$

▶ Remark. For schema mappings whose specification additionally contains t-tgds, an extended definition of GCWA*-solutions is necessary. See [21] for details.

It should be clear that the GCWA*-semantics is invariant under logical equivalent schema mappings. Therefore, the problem described at the beginning of this section does not appear for the GCWA*-semantics.

▶ **Example 22.** Recall Example 19. The minimal solutions for the source instance $S = \{R(c, d)\}$ under M are all solutions for S under M of the form $T_a := \{E(c, a), F(a, d)\}$ for some $a \in Dom$. Now, the GCWA*-solutions for S under M are precisely those instances T for which there is a finite set $C \subseteq Const$ with $T = \bigcup_{a \in C} T_a$. Intuitively, this reflects "precisely the positive information in M and S, and nothing more." It is easy to see that for the query Q from Example 19 and the query Q' from Example 20 we have $cert_{\text{GCWA}^*}(Q, M, S) = \emptyset$ and $cert_{\text{GCWA}^*}(Q', M, S) = \{c\}$, as desired.

▶ Remark. Gottlob et al. [17] propose a different approach of enforcing unique answers on logically equivalent schema mappings under the CWA-semantics (and its relatives). The idea is to first *normalize* the schema mapping as described in their paper, and then answer queries under the desired semantics.

5 The Complexity of Answering Non-Monotone Queries

This section's goal is to compile what is known about the complexity of answering queries under the different semantics introduced in Section 4, which we henceforth call *non-monotone semantics*. To the best of our knowledge, only the *data complexity* of this problem (i.e., its

complexity as a function of the size of the source instance only) has been considered in the literature. For each $s \in \{\text{CWA}, \text{GCWA}^*\}$, each schema mapping M for which $cert_s$ is defined, and each FO-query Q over M's target schema, we will therefore consider the complexity of

$\text{EVAL}_s(M, Q)$	
Input:	a source instance S for M, and a tuple $\bar{a}$ over $\text{dom}(S) \cup \text{dom}(Q)$
Question:	Is $\bar{a} \in cert_s(Q, M, S)$?

Concerning the "mixed world semantics" from Section 4.2, an important parameter is the maximum number of *open positions per atom* in the head of an st-tgd. For an annotation α for the st-tgds in M, let us denote this number by $\#_{\text{op}}(\alpha)$. Then, for each $k \geq 1$ we consider

$\text{EVAL}_k(M, Q)$	
Input:	a source instance S for M, an annotation α for the st-tgds in M such that $\#_{\text{op}}(\alpha) = k$, and a tuple $\bar{a}$ over $\text{dom}(S) \cup \text{dom}(Q)$
Question:	Is $\bar{a} \in cert(Q, M, \alpha, S)$?

Note that for $k = 0$, the problem would correspond to $\text{EVAL}_{\text{CWA}}(M, Q)$.

Let me point out that for monotone queries, most of the non-monotone semantics coincide with the certain answers semantics [22, 32, 2, 21]. The only exception is the CWA-semantics, but only in the context of schema mappings defined by st-tgds, t-tgds, and possibly also egds. This, however, seems to be only due to the choice we made for the incomplete instances represented by instances under integrity constraints. It might well be the case that this changes when we represent such instances in any of the other ways proposed in the literature. Anyway, the collapse to the certain answers semantics indicates once more that the certain answers semantics is well-suited for monotone queries. It also shows that all results concerning the certain answers semantics directly carry over to the non-monotone semantics. So, for example, we know from [33] that there are schema mappings M defined by st-tgds and CQs Q with only two inequalities such that $\text{EVAL}_{\text{CWA}}(M, Q)$ is co-NP-hard. But we also know from [11] that the problem is in PTIME if Q is a UCQ with at most one inequality per disjunct. For the GCWA*-semantics, the latter is true even for the much broader class of *weakly acyclic* schema mappings considered in [11]. This does not hold for the CWA-semantics in general, since it does not coincide with the certain answers semantics on such schema mappings.

5.1 General FO-Queries

Not much is known about the complexity of answering *non-monotone* queries under the non-monotone semantics. In many cases, it is hard to evaluate such queries, which is not surprising given that the non-monotone semantics introduce implicit negation. Concerning the complexity of evaluating general FO-queries under the CWA-semantics and the mixed world semantics, we know:

▶ **Theorem 23** ([22, 32]). *If we restrict ourselves to schema mappings M defined by st-tgds, and* FO*-queries Q over M's target schema, then:*

1. $\text{EVAL}_{CWA}(M, Q) \in$ co-NP*, and there is a schema mapping M defined by st-tgds, and a* FO*-query Q such that $\text{EVAL}_{CWA}(M, Q)$ is* co-NP*-complete.*
2. $\text{EVAL}_1(M, Q) \in$ co-NEXPTIME*, and there is a schema mapping M defined by st-tgds, and a* FO*-query Q such that $\text{EVAL}_1(M, Q)$ is* co-NEXPTIME*-complete.*

3. *For all $k \geq 2$, there is a schema mapping M defined by st-tgds, and a* FO*-query Q such that* $\text{EVAL}_k(M, Q)$ *is undecidable.*
4. *If Q has the form $\forall\bar{x}\exists\bar{y}\,\varphi$, where φ contains no quantifiers, then* $\text{EVAL}_k(M, Q) \in$ co-NP *for all $k \geq 1$.*

The membership part of Theorem 23(2a) follows directly from Theorem 13, whereas for the hardness part we can use [33]. On the other hand, the membership part of Theorem 23(2b) requires more sophisticated techniques. It involves a games argument and is technically quite involved. Hardness is proved by a reduction from a NEXPTIME-complete version of the tiling problem to the complement of $\text{EVAL}_1(M, Q)$. Theorem 23(2c) is established by a reduction from the finite validity problem for first-order logic [30].

▶ Remark. As shown in [22], the upper bound in Theorem 23(2a) holds for more general schema mappings, called *richly acyclic*, which form a subclass of the weakly acyclic schema mappings considered in [11]. It does not hold for weakly acyclic schema mappings, as there are weakly acyclic schema mappings M and FO-queries Q over M's target schema such that $\text{EVAL}_{\text{CWA}}(M, Q)$ is undecidable [22].

Under the GCWA*-semantics, query answering seems to be harder:

▶ **Theorem 24** ([21])**.**
1. *There is a schema mapping M defined by st-tgds and a CQ Q with one negated atom such that* $\text{EVAL}_{GCWA^*}(M, Q)$ *is* co-NP*-hard.*
2. *There is a schema mapping M defined by st-tgds and a* FO*-query Q of the form $\exists\bar{x}\forall y\,\varphi$, where φ contains no quantifiers, such that* $\text{EVAL}_{GCWA^*}(M, Q)$ *is undecidable.*

Remember, though, that all of the above-mentioned results concern the data complexity of query evaluation. For instance, co-NP-hardness of $\text{EVAL}_{\text{CWA}}(M, Q)$ holds for *some* schema mappings M and FO-queries Q, but there are many schema mappings M and FO-queries Q for which $\text{EVAL}_{\text{CWA}}(M, Q)$ is easy. Think, for example, of schema mappings that contain only tgds without existentially quantified variables, and arbitrary FO-queries. For them, $\text{EVAL}_{\text{CWA}}(M, Q)$ is in PTIME. The same is of course true for EVAL_k, $k \geq 1$, and $\text{EVAL}_{\text{GCWA}^*}$. It would be interesting to identify more general classes of schema mappings and queries for which these problems are in PTIME.

5.2 Universal Queries

A very general and natural class of queries for which tractability of query answering under a non-monotone semantics – namely, the GCWA*-semantics – could be established is the class of *universal queries*. Universal queries are FO-queries of the form $\forall\bar{y}\,\varphi$, where φ contains no quantifiers. For schema mappings defined by st-tgds, it is not hard to show:

▶ **Theorem 25** ([21])**.** *For all schema mappings M defined by st-tgds, and for all universal queries Q over M's target schema we have* $\text{EVAL}_{GCWA^*}(M, Q) \in$ co-NP*.*

On the other hand, if we restrict consideration to schema mappings defined by *packed st-tgds*, then universal queries can be answered in *polynomial time*, as we shall see below.

▶ **Definition 26** (Packed st-tgd)**.** An 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 every two distinct relation atoms in ψ share a common variable from $\bar{z}$.

▶ Remark. The schema mapping that is constructed in [21] for proving Theorem 24(2b) is defined by packed st-tgds.

Although quite restrictive, packed st-tgds still allow for non-trivial use of existentially quantified variables in heads of st-tgds. Note that every st-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \to \exists\bar{z}\,\psi(\bar{x},\bar{z}))$, where ψ contains at most two relation atoms with variables from $\bar{z}$, is logically equivalent to a set of packed st-tgds of size at most the number of relation atoms in ψ. Hence, the class of schema mappings defined by packed st-tgds forms an interesting class of schema mappings. An example of an st-tgd that is not packed is $\forall x(P(x) \to \exists y\exists z\exists u(E(x,y)\wedge E(y,z)\wedge E(z,u)))$. The remaining part of this section is devoted to the following result:

▶ **Theorem 27** ([21]). *Let M be a schema mapping defined by packed st-tgds, and let Q be a universal query over M's target schema. Then there is a polynomial time algorithm that takes the core solution for some source instance S for M as input and outputs $cert_{\text{GCWA}^*}(Q,M,S)$.*

Combined with Theorem 3, this result leads to a polynomial time algorithm that takes a source instance S for M as input and outputs $cert_{\text{GCWA}^*}(Q,M,S)$. In particular, we have $\text{EVAL}_{\text{GCWA}^*}(M,Q) \in \text{PTIME}$. Also, recall from Section 3 that core solutions can be used to compute the certain answers to UCQs and other queries. As a consequence, one only needs to materialize the core solution in order to answer such queries and universal queries.

The proof of Theorem 27 is technically very involved. The core part consists of proving the following "decision variant" of Theorem 27:

▶ **Theorem 28** ([21]). *Let M be a schema mapping defined by packed st-tgds, and let Q be a universal query over M's target schema. Then there is a polynomial time algorithm that, given the core solution for some source instance S for M and a tuple $\bar{a}$ as input, decides whether $\bar{a} \in cert_{\text{GCWA}^*}(Q,M,S)$.*

The remainder of this section presents a sketch of the proof of Theorem 28.

5.2.1 GCWA*-Answers and Core Solutions

Let us first see how GCWA*-answers can be obtained from core solutions. Consider a schema mapping M, a source instance S for M, an FO-query Q over M's target schema, and a tuple $\bar{a}$ over $\text{dom}(S)\cup\text{dom}(Q)$. Using the core solution for S under M, how can we decide whether $\bar{a} \in cert_{\text{GCWA}^*}(Q,M,S)$?

Observe that $\bar{a} \notin cert_{\text{GCWA}^*}(Q,M,S)$ if and only if there is a GCWA*-solution T' for S under M such that $\bar{a} \in \neg Q(T')$. Furthermore, recall that GCWA*-solutions are ground solutions that are the union of minimal solutions for S under M. In the case of schema mappings defined by st-tgds, this is equivalent to being a union of ground minimal solutions for S under M. Now let T be the core solution for S under M, and recall from Section 4.1.2 that it represents an incomplete instance $rep(T)$. The following lemma implies that $\bar{a} \notin cert_{\text{GCWA}^*}(Q,M,S)$ if and only if there are $k \geq 1$ and minimal instances $T_1,\ldots,T_k \in rep(T)$ such that $\bar{a} \in \neg Q(T_1 \cup \cdots \cup T_k)$.[10]

▶ **Lemma 29** ([21]). *Let M be a schema mapping defined by st-tgds, let S be a source instance for M, and let T be the core solution for S under M. Then the set of all ground minimal solutions for S under M coincides with the set of all minimal instances in $rep(T)$.*

If Q is a universal query, then $\neg Q$ is equivalent to an *existential query*, i.e., a FO-query of the form $\exists\bar{x}\,\varphi$, where φ is quantifier-free. Thus we have reduced the initial problem of deciding $\bar{a} \in cert_{\text{GCWA}^*}(Q,M,S)$ to a *satisfiability problem for existential queries* over the

[10] As for solutions, an instance $T' \in rep(T)$ is *minimal* if there is no $T'' \in rep(T)$ with $T'' \subsetneq T'$.

set of all instances that are unions of minimal instances in $rep(T)$. Towards solving this satisfiability problem in its full generality, an important subproblem to be solved is the corresponding satisfiability problem for the case that Q has the form $\neg R(\bar{c})$ for some tuple $\bar{c}$ of constants. In this case, the problem simplifies to deciding whether there is a minimal instance in $rep(T)$ that contains $R(\bar{c})$. The next section shows how to find such an instance if one exists.

5.2.2 Finding Atoms in Minimal Possible Worlds

Let T be the core solution for S under M, and let $rep_{\min}(T)$ be the set of all minimal instances in $rep(T)$. Given an atom A, how can we find an instance $T' \in rep_{\min}(T)$ with $A \in T'$ if there is one? Note that in general there are infinitely many instances in $rep(T)$, since each null can be substituted by an arbitrary element of *Const*. However, as shown in [21], it suffices to restrict attention to finitely many representatives. Still, in the worst case there are exponentially many such representatives left, and it is not clear at all how to find a representative containing A.

A very nice structural property of core solutions under schema mappings defined by st-tgds comes to the rescue: that the number of nulls in the *atom blocks* of such core solutions does only depend on the schema mapping.

▶ **Definition 30** ([16])**.** The *Gaifman graph of the atoms* of T is the undirected graph which has the atoms of T as nodes, and an edge between two distinct atoms A and A' if there is a null that occurs both in A and A'. An *atom block* of T is a connected component in the Gaifman graph of the atoms of T.

It follows immediately from results in [13] that for every schema mapping M defined by st-tgds there is an integer s such that for every source instance S for M each atom block in the core solution for S under M contains at most s nulls. The obvious idea is now to try to look only at single atom blocks B of T, and search for an instance in $rep_{\min}(B)$ containing A. Unfortunately, this does not lead to a correct algorithm: If no instance in $rep_{\min}(B)$ contains A, then we can be sure that no instance in $rep_{\min}(T)$ contains A, but if there is an instance in $rep_{\min}(B)$ containing A, then this does not imply that there is also an instance in $rep_{\min}(T)$ that contains A. An example is given in [21].

Instead, it can be shown that there is a subset $\mathcal{S}$ of the representatives of instances in $rep_{\min}(T)$ such that $\mathcal{S}$ has size polynomial in the size of T, and such that it suffices to consider only the instances in $\mathcal{S}$ in order to decide whether there is an instance in $rep_{\min}(T)$ containing A. Furthermore, it is possible to enumerate the instances in $\mathcal{S}$ in polynomial time. The set $\mathcal{S}$ is defined using the following homomorphisms:

▶ **Definition 31.** Let B be an atom block of T, let $\bar{B} := T \setminus B$, and let $C \subseteq Const$.

- Let $val_C(T, B)$ be the set of all mappings $h\colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ such that $h(c) = c$ for all $c \in \mathrm{const}(T)$, $h(\bot) = \bot$ for all $\bot \in \mathrm{nulls}(\bar{B})$, and for every atom $R(a_1, \ldots, a_k) \in B$ we have: if $R(h(a_1), \ldots, h(a_k)) \notin B$, then every null that occurs in $R(h(a_1), \ldots, h(a_k))$ also occurs in B.
- Let $minval_C(T, B)$ be the set of all mappings $h \in val_C(T, B)$ such that there is no $h' \in val_C(T, B)$ with $h'(T) \subsetneq h(T)$.

Let C be the set of all constants that occur in A. Then the set $\mathcal{S}$ is the set of all instances that are the core of $h(T)$ for some $h \in minval_C(T, B)$ and some atom block B of T. For enumerating $\mathcal{S}$, we simply enumerate all the atom blocks B of T and all $h \in minval_C(T, B)$, and compute the core of $h(T)$ using a slight modification of the *blocks algorithm* from [13].

Showing that (1) the instances in $\mathcal{S}$ are indeed representatives of instances in $rep_{\min}(T)$, and that (2) for every atom A contained in some instance in $rep_{\min}(T)$ there is an instance in $\mathcal{S}$ containing A is technically involved. For proving (2), the property that M is defined by packed st-tgds is very important. For details the reader should consult [21].

5.2.3 Solving the General Satisfiability Problem

Finally, let us see how we can put together the results so as to solve the satisfiability problem in its full generality. Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping defined by packed st-tgds, and let Q be an existential query over τ. Given the core solution T for a source instance S under M and a tuple $\bar{a}$ over $\mathrm{dom}(S) \cup \mathrm{dom}(Q)$, how can we decide whether there are $k \geq 1$ and $T_1, \ldots, T_k \in rep_{\min}(T)$ such that $\bar{a} \in Q(T_1 \cup \cdots \cup T_k)$?

We first observe that Q is logically equivalent to a query of the form

$$Q'(\bar{x}) := \bigvee_{i=1}^{m} Q_i(\bar{x}),$$

where each $Q_i(\bar{x})$ 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. It therefore remains to decide whether for some $i \in [m]$, there are $k \geq 1$ and $T_1, \ldots, T_k \in rep_{\min}(T)$ such that $\bar{a} \in Q_i(T_1 \cup \cdots \cup T_k)$.

We can do this as follows. Let $i \in [m]$. For simplicity, assume that the relation atoms in Q_i are $\varphi_{i,1}, \ldots, \varphi_{i,\ell}$. Then for all $j \in [\ell]$, we consider the set $\mathcal{T}_j$ of all pairs (T_j, α_j), where T_j is an instance in the set $\mathcal{S}$ mentioned in Section 5.2.2, and α_j is an assignment for $\varphi_{i,j}$ with $(T_j, \alpha_j) \models \varphi_{i,j}$. Combining tuples from $\mathcal{T}_1, \ldots, \mathcal{T}_\ell$ that are *compatible* in a certain sense,[11] we obtain a set $\mathcal{T}$ of pairs (T^*, α^*) such that T^* has the form $T_1 \cup \cdots \cup T_k$ with each T_i isomorphic to an instance in $\mathcal{S}$, and α^* is an assignment for $\bigwedge_{j=1}^{\ell} \varphi_{i,j}$ such that $(T^*, \alpha^*) \models \bigwedge_{j=1}^{\ell} \varphi_{i,j}$. Finally, we check whether there is a pair $(T^*, \alpha^*) \in \mathcal{T}$ such that T^* can be padded with a large enough number of disjoint copies of T so that the resulting instance T^{**} satisfies $\bar{a} \in Q_i(T^{**})$. For the proof of correctness, which is technically quite involved, I refer the reader to [21].

6 Conclusions

Query answering is a fundamental task in data exchange and data integration. The standard semantics for queries in these areas is the certain answers semantics. While this is adequate for monotone queries, it may lead to counter-intuitive answers for *non-monotone* queries. This chapter surveyed various semantics (the CWA-semantics, the "mixed world" semantics, the endomorphic images semantics, and the GCWA*-semantics) that were designed for answering non-monotone queries. Each of these semantics is based on a variant of the CWA to reduce the set of solutions, and to answer queries with respect to the reduced set of

[11] Informally, (T_1, α_1) and (T_2, α_2) are compatible if T_1 and T_2 can be "glued together" by identifying the values assigned to some variable that occurs both in the domain of α_1 and in the domain of α_2, while leaving the other values untouched.

solutions. Answering non-monotone queries under any of these semantics may be co-NP-hard, or co-NEXPTIME-hard (in the case of the "mixed world" semantics with at most one open position per atom in an st-tgd), or even undecidable. Note, however, that these results speak about the *data complexity* of the problem. In particular, single schema mappings M and queries Q were exhibited for which the problem is hard. For many schema mappings and queries, the problem is easy, though, and it would be interesting to identify more general classes of schema mappings and queries for which the problem is tractable. We presented one such example: the GCWA*-answers to universal queries can be computed in polynomial time under schema mappings defined by packed st-tgds.

Apart from identifying more general classes of schema mappings and queries for which query answering is tractable, there are many more problems that still need to be solved. To give three examples: Since there are several semantics for non-monotone queries, it would be nice to have *formal criteria* (e.g., in the style of [7], see also [10]) for comparing them and to understand their strengths and weaknesses relative to each other. Furthermore, it is worth studying the *combined complexity* of query answering under non-monotone semantics. There are a few results on the combined complexity of computing the certain answers [5], but for non-monotone semantics there are no such results. Finally, a technical question concerning the polynomial time algorithm for computing GCWA*-answers to universal queries (Theorem 27): Can it be *extended* to more general schema mappings? It seems possible to do this for schema mappings defined by st-tgds.

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 *Proceedings of the 27th ACM Symposium on Principles of Database Systems (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 *Proceedings of the 23th ACM Symposium on Principles of Database Systems (PODS)*, pages 229–240, 2004.

4 M. Arenas, P. Barceló, L. Libkin, and F. Murlak. *Relational and XML Data Exchange*. Morgan & Claypool, 2010.

5 M. Arenas, P. Barceló, and J. Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. *Theory of Computing Systems*, 49(2):489–564, 2011.

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

7 S. Brass and J. Dix. Characterizations of the disjunctive stable semantics by partial evaluation. *Journal of Logic Programming*, 32(3):207–228, 1997.

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

9 A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*, pages 149–158, 2008.

10 J. Dix, U. Furbach, and I. Niemelä. Nonmonotonic reasoning: Towards efficient calculi and implementations. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 19, pages 1241–1354. The MIT Press, 2001.

11 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.

12 R. Fagin, P. G. Kolaitis, A. Nash, and L. Popa. Towards a theory of schema-mapping optimization. In *Proceedings of the 27th Symposium on Principles of Database Systems (PODS)*, pages 33–42, 2008.

13 R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *ACM Transactions on Database Systems*, 30(1):174–210, 2005.

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

15 F. Geerts and B. Marnette. Static analysis of schema-mappings ensuring oblivious termination. In *Proceedings of the 13th International Conference on Database Theory (ICDT)*, pages 183–195, 2010.

16 G. Gottlob and A. Nash. Efficient core computation in data exchange. *Journal of the ACM*, 55(2):Article 9, 2008.

17 G. Gottlob, R. Pichler, and V. Savenkov. Normalization and optimization of schema mappings. *The VLDB Journal*, 20(2):277–302, 2011.

18 S. Greco and F. Spezzano. Chase termination: A constraints rewriting approach. *Proceedings of the VLDB Endowment (PVLDB)*, 3(1):93–104, 2010.

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

20 A. Hernich. *Foundations of Query Answering in Relational Data Exchange*. PhD thesis, Institut für Informatik, Goethe-Universität Frankfurt am Main, 2010. Published at Logos Verlag Berlin, ISBN 978-3-8325-2735-8, 2010.

21 A. Hernich. Answering non-monotonic queries in relational data exchange. *Logical Methods in Computer Science*, 7(3):Paper 9, 2011. Special Issue for the 13th International Conference on Database Theory, ICDT 2010.

22 A. Hernich, L. Libkin, and N. Schweikardt. Closed world data exchange. *ACM Transactions on Database Systems*, 36(2):Article 14, 2011.

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

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

25 T. Imielinski. Incomplete deductive databases. *Annals of Mathematics and Artificial Intelligence*, 3(2–4):259–294, 1991.

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

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

28 G. Lausen, M. Meier, and M. Schmidt. On chase termination beyond stratification. *Proceedings of the VLDB Endowment (PVLDB)*, 2(1):970–981, 2009.

29 M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the 21th ACM Symposium on Principles of Database Systems (PODS)*, pages 229–240, 2002.

30 L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

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

32 L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. *Journal of Computer and System Sciences*, 77(3):542–571, 2011.

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

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

35 J. Minker. On indefinite databases and the closed world assumption. In D. W. Loveland, editor, *Proceedings of the International Conference on Automated Deduction (CADE)*, volume 138 of *Lecture Notes in Computer Science*, pages 292–308. Springer-Verlag, 1982.

36 R. Reiter. On closed world data bases. In H. Galaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, 1978.

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

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