
Logic and Data Exchange: Which Solutions are “Good” Solutions?

André Hernich
Nicole Schweikardt

Institut für Informatik
Goethe-Universität Frankfurt am Main
Postfach 11 19 32
D-60054 Frankfurt am Main, Germany
`{hernich,schweika}@informatik.uni-frankfurt.de`

Abstract

This paper gives an introduction into the area of data exchange, with a special emphasis on the question of which solutions can be considered “good” solutions. We will concentrate on notions of “good” solutions for query answering, in particular, *universal solutions*, the *core* of the universal solutions, and *CWA-solutions*.

1 Introduction

Data exchange deals with the problem of *translating* data that is structured in an “old” format into data structured in a “new” format. The “old” and the “new” format are called *source schema* and *target schema*, respectively. The task in data exchange can be described as follows: given (i) a database instance over the source schema and (ii) a specification of the relationship between the source and the target, construct a *solution*, i.e., a database instance over the target schema that satisfies the given relationship. Preferably, in case that solutions exist at all, one would like to find particular solutions that reflect the given source data as accurately as possible.

Such data exchange problems occur in many real-world applications of database systems; and systems for solving this task have been developed and implemented in the last few years [31, 22]. From a logician’s point of view, the setting in data exchange can be stated as follows: The source schema and the target schema are finite relational vocabularies, the database instances over the source schema and the target schema are finite structures over these vocabularies, and the specification of the relationship between the source and the target is given by a set of formulas of a suitable logic.

The present paper’s goal is to give an introduction into the area of data exchange, addressed to readers who have a background in logic, but who

are not necessarily experts in database theory. A special emphasis is put on the question of which solutions can be considered “good” solutions.

The paper is structured as follows: In Section 2 we give an introduction to the basic problem of data exchange, including the notions of *schema mappings* and *solutions*. In Section 3 we concentrate on the first two notions of “good” solutions: the *universal solutions* and the *core*. Section 4 gives a summary on how to compute solutions. Section 5 deals with the question of how to answer queries that are formulated with respect to the target schema, and points out certain deficiencies of some notions of semantics of queries. Section 6 presents the *CWA-solutions* as a notion of “good” solutions for which these deficiencies do not occur. Finally, Section 7 concludes the paper by pointing out further topics of research in the area of data exchange.

We are aware that data exchange does not belong to the core topics of the LOFT conference. However, we believe that data exchange still is of reasonable interest for the LOFT community: it deals with fundamental questions in mathematical logic, it is concerned with deciding on the “right” semantics of certain logics, and in connection with *peer data exchange* [9, 19, 20] it deals with the interaction of multiple peers. Furthermore, game theoretic concepts help to gain a better understanding of a particular notion of “good” solutions for data exchange: in Section 6 we present a new, game theoretic characterization of the CWA-solutions.

Of course, the present paper can only give a very brief introduction to some basic concepts and results in data exchange. As a starting point for gaining an in-depth knowledge of the area of data exchange we refer the interested reader to Kolaitis’ excellent survey [26], to Barceló’s very recent survey [7], and to the articles on data exchange published in the proceedings of the *ACM Symposium on Principles of Database Systems (PODS)* and the *International Conference on Database Theory (ICDT)*.

2 Basic notions

A *schema* is a finite relational vocabulary, i.e., a finite set of relation symbols where each relation symbol is associated with an arity. An *instance* I over a schema σ assigns to each relation symbol $R \in \sigma$ a *finite* relation R^I of the same arity as R . Thus, an instance over σ is a finite σ -structure. Given two schemas σ and τ (called *source schema* and *target schema*, respectively), and a specification of the relationship between the source and the target, the goal in data exchange is to transform an instance S (the *source instance*) over σ into an instance T (a *target instance*) over τ that satisfies the given specification. Let us illustrate this with an example.

Example 2.1. Consider the scenario of the fusion of two airlines, say *KLA* and *Air Flight (AF)*, for short, into a single airline. Suppose we have a source instance S over the source schema $\sigma = \{KLA, AF\}$, where KLA^S and

AF^S , respectively, contain the flights of *KLA* and *Air Flight*, represented by triples $(departure_city, arrival_city, flight_number)$. We want to create a target instance T over the target schema $\tau = \{New\}$ such that New^T contains the flights of the new airline, represented by triples $(departure_city, arrival_city, aircraft_type)$. Moreover, we require that the new airline offers (at least) the same direct connections as *KLA*, and that for each route of *Air Flight*, the cities remain reachable with at most one change of planes. \square

Throughout this article we will assume that σ and τ are disjoint. We write $S \cup T$ to denote the instance of schema $\sigma \cup \tau$ where each $R \in \sigma$ is assigned the relation R^S , and each $R \in \tau$ is assigned the relation R^T .

Furthermore, we assume that all elements that occur in a tuple that belongs to a relation R^I of some (source or target) instance I belong to a fixed infinite set Dom of potential data values.

2.1 Schema mappings

Requirements such as those described in Example 2.1 are formalized by *schema mappings*:

Definition 2.2. Let σ and τ be two disjoint schemas.

(a) A *schema mapping* M from σ to τ is a non-empty subset of

$$\{(S, T) : S \text{ is an instance over } \sigma, \text{ and } T \text{ is an instance over } \tau\}.$$

We call σ the *source schema* of M , and an instance over σ is called *source instance* for M . Analogously, we call τ the *target schema* of M , and an instance over τ is called *target instance* for M .

(b) Given a source instance S for a schema mapping M , a *solution* for S under M is a target instance T for M such that $(S, T) \in M$. \square

Note that schema mappings are not required to be mappings (i.e., functions) in the mathematical sense. According to Definition 2.2, a schema mapping can be an arbitrary relation between the class of instances over the source schema and the class of instances over the target schema. Thus, for a given source instance S there might exist either no solution, or exactly one solution, or more than one solution.

Example 2.3. The requirements from Example 2.1 can be formalized by a schema mapping M from the source schema $\{KLA, AF\}$ to the target schema $\{New\}$ that consists of precisely all tuples (S, T) , where S is a source instance and T is a target instance for M such that

(1) for every tuple $(x_1, x_2, y) \in KLA^S$, there is an aircraft type z such that $(x_1, x_2, z) \in New^T$, and

- (2) for every tuple $(x_1, x_2, y) \in AF^S$, there either is an aircraft type z such that (x_1, x_2, z) belongs to New^T , or there is an intermediate stop z_1 and aircraft types z' and z'' such that (x_1, z_1, z') and (z_1, x_2, z'') belong to New^T . \square

One of the goals in data exchange is: Given a schema mapping M and a source instance S , decide whether a solution for S under M exists — and if so, compute such a solution T . In particular, this leads to the following decision problem:

EXISTENCE-OF-SOLUTIONS(M)
Input: A source instance S for M .
Question: Is there a solution for S under M ?

Preferably, in case that solutions exist at all, one would like to find a solution that reflects the given data as accurately as possible.

2.2 How to specify schema mappings

For *specifying* schema mappings in a high-level, declarative way, it seems natural to consider formulas of a certain logic, e.g., first-order logic. Several formalisms for schema specification are conceivable, among them, for example:

- (1) *Logical interpretations*, e.g., first-order interpretations: Here, a tuple $f := (\varphi_R)_{R \in \tau}$ of formulas of vocabulary σ associates with every source instance S the target instance $f(S)$ where, for each relation symbol $R \in \tau$, $R^{f(S)}$ consists of all tuples \bar{a} with $S \models \varphi_R(\bar{a})$. The interpretation f thus specifies the schema mapping

$$M_f := \{ (S, f(S)) : S \text{ is a source instance} \}.$$

Note that such logical interpretations f allow for specifying only schema mappings where each source instance S has exactly one solution. For real-world applications, this is far too restrictive. Usually, one does not intend to fully specify the target, but only to describe certain properties that the target should have as, e.g., in the data exchange scenario described in Example 2.1.

One therefore usually considers the following, more general way of specifying schema mappings.

- (2) A set Σ of formulas of a certain logic, e.g., first-order logic over vocabulary $\sigma \cup \tau$, specifies the schema mapping

$$M_\Sigma = \left\{ (S, T) : \begin{array}{l} S \text{ is a source instance, } T \text{ a target instance,} \\ \text{and } S \cup T \models \Sigma \end{array} \right\}.$$

Thus, the formulas in Σ can be viewed as a formal specification of the relationship between the source and the target.

However, if the class of formulas allowed for choosing Σ is too powerful, the problem $\text{EXISTENCE-OF-SOLUTIONS}(M_\Sigma)$ may be undecidable. The following example exposes a particular first-order sentence φ such that $\text{EXISTENCE-OF-SOLUTIONS}(M_{\{\varphi\}})$ is undecidable.¹ Note that the difficulty in proving the undecidability is that the schema mapping is assumed to be fixed. If the schema mapping is regarded as part of the input, undecidability is a simple corollary to Trakhtenbrot’s theorem.

Example 2.4. Consider *Post’s correspondence problem*, which is well-known to be undecidable. The problem is defined as follows:

Post’s correspondence problem
Input: A number $k \geq 1$ and a list $L = (u_1, v_1), \dots, (u_k, v_k)$ of k pairs of non-empty words u_i and v_i over an alphabet $\{a_1, \dots, a_\ell\}$.
Question: Are there a number $m \geq 1$ and indices $i_1, \dots, i_m \in \{1, \dots, k\}$ such that $u_{i_1}u_{i_2} \cdots u_{i_m} = v_{i_1}v_{i_2} \cdots v_{i_m}$?

The input to Post’s correspondence problem can be represented by an instance S over the source schema $\sigma = \{Ord, U, V\}$, where Ord is a binary relation symbol, U and V are ternary relation symbols,

- Ord^S consists of all pairs (p, q) of integers such that $1 \leq p \leq q \leq \max\{k, \ell, |u_1|, \dots, |u_k|, |v_1|, \dots, |v_k|\}$, and
- U^S (resp. V^S) consists of all triples (i, p, λ) of integers where $1 \leq i \leq k$, $1 \leq p \leq |u_i|$ (resp., $|v_i|$), and $1 \leq \lambda \leq \ell$, such that the p -th letter in u_i (resp., in v_i) is a_λ .

We represent a solution

$$m \geq 1, \quad i_1, \dots, i_m \in \{1, \dots, k\}, \quad w = u_{i_1}u_{i_2} \cdots u_{i_m} = v_{i_1}v_{i_2} \cdots v_{i_m}$$

to Post’s correspondence problem by an instance T over the target schema $\tau = \{Ord', P, WU, WV\}$, where Ord' and P are binary relation symbols, and WU and WV are ternary relation symbols, as follows: For $n := |w|$,

- Ord'^T consists of all pairs (p, q) of integers such that $1 \leq p \leq q \leq \max\{n, m, k, \ell, |u_1|, \dots, |u_k|, |v_1|, \dots, |v_k|\}$,
- P^T consists of all pairs (j, i_j) with $1 \leq j \leq m$,

¹ A different example (based on the *halting problem for Turing machines* rather than *Post’s correspondence problem*) is sketched in [26].

- WU^T consists of all triples (r, j, p) of integers such that $1 \leq r \leq n$, $1 \leq j \leq m$, $1 \leq p \leq |u_{i_j}|$, and $r = |u_{i_1}| + \dots + |u_{i_{j-1}}| + p$ (i.e., triple (r, j, p) corresponds to the information that the r -th position in w is the p -th position in the j -th segment of the decomposition of w into $u_{i_1} \dots u_{i_m}$)
- WV^T consists of all triples (r, j, p) of integers such that $1 \leq r \leq n$, $1 \leq j \leq m$, $1 \leq p \leq |v_{i_j}|$, and $r = |v_{i_1}| + \dots + |v_{i_{j-1}}| + p$ (i.e., triple (r, j, p) corresponds to the information that the r -th position in w is the p -th position in the j -th segment of the decomposition of w into $v_{i_1} \dots v_{i_m}$).

Now it is an easy (but somewhat tedious) exercise to construct a first-order sentence φ of vocabulary $\sigma \cup \tau$, such that for any source instance S which represents an input to Post's correspondence problem, and for any target instance T , the following is true:² $S \cup T \models \varphi$ if, and only if, T is isomorphic to an instance which represents a solution to Post's correspondence problem.

The undecidability of Post's correspondence problem thus implies that the problem $\text{EXISTENCE-OF-SOLUTIONS}(M_{\{\varphi\}})$ is undecidable. \square

As a language for specifying schema mappings one therefore usually does not allow full first-order logic, but restricts attention to certain fragments of first-order logic. Ideally, such a fragment should be *expressive enough* to specify schema mappings used in practical applications, and *simple enough* to allow for efficient algorithms for the $\text{EXISTENCE-OF-SOLUTIONS}$ problem.

A systematic study of suitable schema specification languages has started just recently (cf., [10]). Up to date, most publications in the area of data exchange deal with schema mappings M_Σ that are specified by a set Σ that consists of so-called *dependencies* of the following kind.

Definition 2.5 (s-t-gds, t-gds, egds).

- (a) (i) A *source-to-target tuple generating dependency* (s-t-tgd, for short) is a formula δ of the form

$$\forall \bar{x} \forall \bar{y} \left(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}) \right),$$

where

- $\bar{x}, \bar{y}, \bar{z}$ are finite sequences of pairwise distinct first-order variables,
- $\varphi(\bar{x}, \bar{y})$ is a conjunction of relational atoms of the form $R(u_1, \dots, u_r)$ where $R \in \sigma$, r is the arity of R , and each u_i is either a variable in \bar{x}, \bar{y} or an element in Dom , and

² Recall from the beginning of Section 2 that, by definition, all instances are finite.

- $\psi(\bar{x}, \bar{z})$ is a conjunction of relational atoms of the form $R(u_1, \dots, u_r)$ where $R \in \tau$, r is the arity of R , and each u_i is either a variable in \bar{x}, \bar{z} or an element in Dom .
- (ii) A *target tuple generating dependency* (*t-tgd*, for short) is a formula δ of the same form as in (i), where both, $\varphi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ are conjunctions of atoms over τ .
- (iii) An *equality generating dependency* (*egd*, for short) is a formula of the form

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

where $\varphi(\bar{x})$ is a conjunction of atoms over τ , $\bar{x} = x_1, \dots, x_k$ is a finite sequence of first-order variables, and $i, j \in \{1, \dots, k\}$.

- (b) If δ is a s-t-tgd or a t-tgd, S is a source instance, and T a target instance, we say that $S \cup T$ *satisfies* δ (in symbols: $S \cup T \models \delta$), if for all interpretations of \bar{x}, \bar{y} with elements \bar{a}, \bar{b} in Dom for which each of the conjuncts in $\varphi(\bar{a}, \bar{b})$ is true in $S \cup T$, there exists an interpretation of \bar{z} with elements \bar{c} in Dom such that each of the conjuncts in $\psi(\bar{a}, \bar{c})$ is true in $S \cup T$.

Similarly, if δ is an egd, we say that $S \cup T$ satisfies δ , if for all interpretations of $\bar{x} = x_1, \dots, x_k$ with elements a_1, \dots, a_k in Dom for which each of the conjuncts in $\varphi(\bar{a})$ is true in $S \cup T$, it holds that $a_i = a_j$. \square

Definition 2.6. We write $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ to denote that M is a schema mapping from σ to τ defined via $M := M_{\Sigma}$, where $\Sigma = \Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ and Σ_{st} (the so-called set of *source-to-target dependencies*) is a finite set of s-t-tgds, and Σ_{t} (the so-called set of *target dependencies*) is a finite set of t-tgds and egds. \square

Example 2.7. Let us consider the schema mapping M described in Example 2.3. Condition (1) can be formalized by the s-t-tgd

$$\delta_1 := \forall x_1 \forall x_2 \forall y (KLA(x_1, x_2, y) \rightarrow \exists z \text{New}(x_1, x_2, z)).$$

Condition (2) can be formalized by the formula

$$\forall x_1 \forall x_2 \forall y \left(AF(x_1, x_2, y) \rightarrow \left(\exists z \text{New}(x_1, x_2, z) \vee \exists z \exists z' \exists z'' (\text{New}(x_1, z, z') \wedge \text{New}(z, x_2, z'')) \right) \right).$$

Note, however, that this is *not* an s-t-tgd since, according to Definition 2.5, disjunctions are not allowed in s-t-tgds. In order to formalize (a variant³

³ It should be mentioned that we use a little “hack” here: δ_3 makes sure that a “self-loop” is added to every destination airport.

of) condition (2) we therefore use the s-t-tgd $\delta_2 :=$

$$\forall x_1 \forall x_2 \forall y (AF(x_1, x_2, y) \rightarrow \exists z \exists z' \exists z'' (New(x_1, z, z') \wedge New(z, x_2, z'')))$$

and the t-tgd

$$\delta_3 := \forall x_1 \forall x_2 \forall y (New(x_1, x_2, y) \rightarrow \exists z New(x_2, x_2, z)).$$

With $M_{airline}$ we will henceforth denote the schema mapping $(\sigma, \tau, \Sigma_{st}, \Sigma_t)$ with $\sigma = \{KLA, AF\}$, $\tau = \{New\}$, $\Sigma_{st} = \{\delta_1, \delta_2\}$, and $\Sigma_t = \{\delta_3\}$. \square

2.3 Solutions

Recall that for a schema mapping $M = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ and a source instance S , a *solution for S under M* is a target instance T such that $S \cup T \models \Sigma_{st} \cup \Sigma_t$.

Example 2.8. Let us consider the schema mapping $M_{airline}$ from Example 2.7 and the source instance $S_{airline}$ with

$KLA^{S_{airline}} :=$ $\{ (AMS, TXL, KL1812),$ $(FRA, AMS, KL1772),$ $(FRA, AMS, KL1431) \}$	and	$AF^{S_{airline}} :=$ $\{ (CDG, TXL, AF123),$ $(CDG, AMS, AF837) \}$
---	-----	--

Then, each of the following target instances T_1 and T_2 is a solution for $S_{airline}$ under $M_{airline}$:

$New^{T_1} :=$ $\{ (AMS, TXL, F100),$ $(FRA, AMS, A310),$ $(FRA, AMS, F100),$ $(CDG, TXL, F100),$ $(TXL, AMS, B707),$ $(TXL, TXL, stay-here),$ $(AMS, AMS, stay-here) \}$	$New^{T_2} :=$ $\{ (AMS, TXL, F100),$ $(FRA, AMS, A310),$ $(CDG, AMS, B707),$ $(TXL, TXL, stay-here),$ $(AMS, AMS, stay-here) \}$
---	---

\square

In light of the fact that, for a given schema mapping M and a source instance S , either no solution or more than just one solution may exist, the question “Which solutions are particularly “good” solutions?” occurs in a natural way.

One kind of particularly good solutions are the *universal solutions* introduced in the next section.

3 Universal solutions and the core

Universal solutions were introduced by Fagin, Kolaitis, Miller, and Popa in [14] as a formalization of an intuitive notion of “most general solutions”.

Before giving the formal definition of universal solutions, let us first note that target instances may contain two different kinds of values: first of all, there may be values that are present already in the given source instance. E.g., in Example 2.8, the values AMS, TXL, FRA, CDG are present in S_{airline} and in every solution T for S_{airline} under M_{airline} . Such values are called *constants*. But the solutions T_1 and T_2 in Example 2.8 also contain values that are not present in S_{airline} , e.g., the values for *aircraft_type* (in T_1 and T_2 , these are the values F100, A310, B707, and stay-here). The dependencies in Σ_{st} and Σ_{t} enforce that such values are present in every solution, but they do not specify the precise choice of these values. E.g., when replacing in T_1 the tuple (AMS, TXL, F100) with the tuple (AMS, TXL, sunshine), one obtains another solution for S_{airline} under M_{airline} . It would therefore be nice to have “placeholders” at hand, i.e., particular values which tell us that we know that (a) some value has to be inserted here, and that (b) no constraint on the precise choice of this value is given. Such placeholders are called *nulls*.

From now on, we therefore assume that the set Dom of potential values is the union of two infinite, disjoint sets Const (the set of *constants*) and Null (the set of *nulls*). We will often use the symbols $\perp, \perp', \perp_1, \perp_2, \dots$ to denote nulls.

Source instances will usually contain only constants, whereas target instances may contain both, constants and nulls. The dependencies fixed in Definition 2.5 are allowed to contain constants, but no nulls (i.e., in Definition 2.5 (a), Dom must be replaced by Const). Two target instances T and T' are called *isomorphic* if they are identical up to renaming of nulls.

Let us now come back to the question “Which solutions are particularly “good” solutions?”. Intuitively, we are particularly interested in those solutions that carry *no more and no less* information than is required by S and M . In [14], the *universal solutions* were introduced as a kind of “most general solutions” in this sense. The following notion of *homomorphism* is crucial for the formal definition of universal solutions.

Definition 3.1 (homomorphism). Let T and T' be two instances over τ . A *homomorphism* from T to T' is a mapping $h : \text{Dom} \rightarrow \text{Dom}$ which has the following two properties:

- (1) h is the identity on Const , i.e., $h(c) = c$ for all $c \in \text{Const}$, and
- (2) for all $R \in \tau$ and all tuples $(a_1, \dots, a_r) \in R^T$, we have $(h(a_1), \dots, h(a_r)) \in R^{T'}$. □

Note that, by definition, a homomorphism has to be the identity on Const , but it may map a null onto either a null or a constant.

Definition 3.2 (universal solution [14]). Let M be a schema mapping and let S be a source instance for M . A solution T for S under M is a *universal solution* if for every solution T' for S under M there exists a homomorphism h from T to T' . \square

Thus, a universal solution is a “most general solution” in the sense that it can be homomorphically mapped into every other solution.

Example 3.3. Consider the schema mapping M_{airline} from Example 2.7 and the source instance S_{airline} from Example 2.8. Then, none of the solutions T_1 and T_2 from Example 2.8 is a universal solution. However, the solutions T_3 and T_4 given in Figure 1 are universal solutions. \square

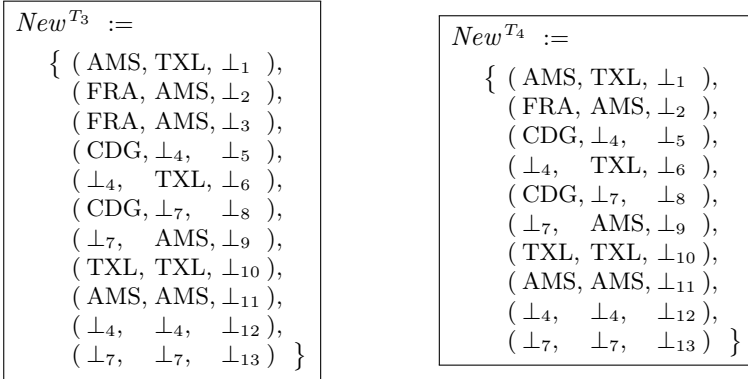


FIGURE 1. Two universal solutions T_3 and T_4 for S_{airline} under M_{airline} .

Note that there is no guarantee that universal solutions exist. There are examples of schema mappings and source instances for which solutions do exist, but universal solutions do not:

Example 3.4. Let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be the schema mapping where $\sigma = \{F\}$, $\tau = \{E\}$, $\Sigma_{\text{st}} = \{\forall x_1 \forall x_2 (F(x_1, x_2) \rightarrow E(x_1, x_2))\}$ and $\Sigma_{\text{t}} = \{\forall x \forall y (E(y, x) \rightarrow \exists z E(x, z))\}$. Let S be the source instance with $F^S = \{(a, b)\}$. Then, the target instance T with $E^T = \{(a, b), (b, b)\}$ is a solution for S under M .

However, it is not difficult to see that there is no *universal* solution for S under M : Assume, for contradiction, that T' is a universal solution for S under M . Viewing $E^{T'}$ as the set of edges of a directed graph, let n be

the size of the smallest cycle in this graph (a cycle must exist since T' is finite and M enforces that every node has out-degree at least one). Now let T'' be a solution for S under M that consists of a cycle on $n+1$ nodes. Since T' is a universal solution, there must be a homomorphism h from T' to T'' . Letting v_1, \dots, v_n be the nodes on the cycle of length n in T' , the homomorphic images $h(v_1), \dots, h(v_n)$ of these nodes must form a cycle in T'' . However, by definition, the shortest cycle in T'' has length $n+1$. \square

It therefore makes sense to consider, apart from the problem EXISTENCE-OF-SOLUTIONS(M), also the following problem:

EXISTENCE-OF-UNIVERSAL-SOLUTIONS(M)

Input: A source instance S for M .

Question: Is there a universal solution for S under M ?

In [16], Fagin, Kolaitis, and Popa introduced the notion of a unique “minimal” universal solution, the so-called *core of the universal solutions*. To give the formal definition of the core of the universal solutions, we need to fix the notion of a *sub-instance*. An instance C over τ is a sub-instance of an instance T (in symbols: $C \subseteq T$) if $R^C \subseteq R^T$, for every $R \in \tau$.

Definition 3.5 ([23]). Let T be an instance over τ . An instance $C \subseteq T$ is a *core* of T , if there is a homomorphism h from T to C , and there is no homomorphism from T to any C' with $C' \subseteq C$ and $C' \neq C$. \square

It can be easily seen (cf., [23]) that every instance over τ has a core, and that the core is unique up to isomorphism. Furthermore, it has been shown in [16] that all universal solutions for S under M have the same core (up to isomorphism), and this core is a universal solution itself.

Definition 3.6. If universal solutions for S under M exist, then $\text{CORE}_M(S)$ denotes the *core* of the universal solutions for S under M . \square

Note that universal solutions for S under M exist if, and only if, $\text{CORE}_M(S)$ exists.

4 How to compute solutions

An important question in the area of data exchange is: Given a schema mapping M and a source instance S for M , how can we compute a solution T for S under M , how can we compute a universal solution, and how can we compute $\text{CORE}_M(S)$?

Let us first note that the class of schema mappings fixed in Definition 2.6 still contains schema mappings for which the problems EXISTENCE-OF-SOLUTIONS, respectively, EXISTENCE-OF-UNIVERSAL-SOLUTIONS are undecidable:

Theorem 4.1 (Kolaitis, Panttaja, Tan [27]). There is a schema mapping $M' = (\sigma', \tau', \Sigma'_{st}, \Sigma'_t)$ (consisting of one s-t-tgd, two t-tgds, and one egd) such that $\text{EXISTENCE-OF-SOLUTIONS}(M')$ is undecidable. \square

Theorem 4.2 (Hernich, Schweikardt [24]). There is a schema mapping $M'' = (\sigma'', \tau'', \Sigma''_{st}, \Sigma''_t)$ (containing s-t-tgds and t-tgds, but no egds) such that $\text{EXISTENCE-OF-UNIVERSAL-SOLUTIONS}(M'')$ is undecidable. \square

The proof of Theorem 4.1 is by a reduction from the *embedding problem for finite semigroups*, which is known to be undecidable. The proof of Theorem 4.2 is by a reduction from the halting problem for Turing machines. Interestingly, M' and M'' can be chosen in such a way that the problems $\text{EXISTENCE-OF-SOLUTIONS}(M'')$ and $\text{EXISTENCE-OF-UNIVERSAL-SOLUTIONS}(M')$ are trivial.

It is not difficult to see that, if M has only source-to-target dependencies (i.e., the set Σ_t of target dependencies is empty), then *every* source instance has at least one universal solution, i.e., the problems $\text{EXISTENCE-OF-SOLUTIONS}(M)$ and $\text{EXISTENCE-OF-UNIVERSAL-SOLUTIONS}(M)$ are trivial. Furthermore, a subclass of the schema mappings fixed in Definition 2.6 has been identified, for which the $\text{EXISTENCE-OF-SOLUTIONS}$ problem is decidable, and solutions can be computed efficiently: the so-called *weakly acyclic* schema mappings.

Definition 4.3 (weakly acyclic schema mappings [14, 12]).

Let $M = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be a schema mapping.

- A *position* over τ is a pair (R, i) such that $R \in \tau$ and $i \in \{1, \dots, r\}$, where r is the arity of R . Given a conjunction φ of relational atomic formulas, and a variable x , we say that x *appears at position* (R, i) in φ if φ contains a conjunct $R(u_1, \dots, u_r)$ with $u_i = x$.
- The *dependency graph* of M is a directed graph, whose vertices are the positions over τ , and whose edges are given as follows: For every t-tgd $\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$ in Σ_t , every variable x in \bar{x} , and every position (R, i) at which x appears in φ , there is
 - a *copying edge* from (R, i) to every position at which x appears in ψ , and
 - an *existential edge* from (R, i) to every position at which some variable from \bar{z} appears in ψ .
- M is called *weakly acyclic* if no cycle in the dependency graph of M contains an existential edge. \square

Example 4.4. Consider a schema mapping $M_1 = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_t)$, where Σ_t contains exactly one t-tgd δ_1 of the form $\forall x \forall y (E(x, y) \rightarrow \exists z E(x, z))$. Then, M_1 is weakly acyclic (cf., the dependency graph of M_1 , illustrated in Figure 2).

On the other hand, if $M_2 = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_t)$ is such that Σ_t contains exactly one t-tgd δ_2 of the form $\forall x \forall y (E(y, x) \rightarrow \exists z E(x, z))$, then M_2 is not weakly acyclic (see Figure 2). \square



FIGURE 2. The dependency graphs of M_1 (left) and M_2 (right). Copying edges are grey, existential edges are black.

The following theorem shows that for weakly acyclic schema mappings, the problems EXISTENCE-OF-SOLUTIONS(M) and EXISTENCE-OF-UNIVERSAL-SOLUTIONS(M) coincide, and solutions can be computed in polynomial time.

Theorem 4.5 (Fagin, Kolaitis, Miller, Popa [14]).

Let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_t)$ be a weakly acyclic schema mapping.

- (a) For any source instance S , the following is true: There is a solution for S under M if and only if there is a universal solution for S under M .
- (b) There is a polynomial-time⁴ algorithm which, given a source instance S , tests whether a solution for S under M exists and, if so, produces a universal solution T for S . \square

The algorithm from Theorem 4.5(b) is based on the *chase* procedure.

For describing the chase procedure, it is convenient to identify an instance I with the set $atoms(I)$ of all atoms in I , i.e., $atoms(I)$ consists of all statements $R(\bar{u})$ where R is a relation symbol in the underlying schema and \bar{u} is a tuple of elements in Dom such that $\bar{u} \in R^I$. Similarly, if A is a finite set of atoms, we write $inst(A)$ to denote the instance I with $atoms(I) = A$.

Now let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_t)$ be a weakly acyclic schema mapping. For a given source instance S , the chase procedure starts with the set $A_1 := atoms(S)$ and proceeds with steps $i = 1, 2, 3, \dots$ as follows: If $inst(A_i)$ satisfies all dependencies in $\Sigma_{\text{st}} \cup \Sigma_t$, then stop and output the target instance

⁴ Note that here the schema mapping is assumed to be fixed and the complexity of the algorithm is measured only in terms of the size of the source instance S . The complexity of the variant of the problem where the schema mapping M is part of the input has been studied in [27].

T with $\text{atoms}(T) = A_i \setminus \text{atoms}(S)$ (obviously, T then is a solution for S under M). Otherwise, there must exist a s-t-tgd in Σ_{st} or a t-tgd in Σ_{t} or an egd in Σ_{t} that is not satisfied by $\text{inst}(A_i)$. We distinguish between two cases.

Case 1: Σ_{t} contains an egd of the form $\forall \bar{x}(\varphi(\bar{x}) \rightarrow x_i = x_j)$ for which there is a tuple \bar{a} of elements in Dom such that each of the conjuncts in $\varphi(\bar{a})$ is satisfied, but $a_i \neq a_j$. If a_i and a_j are both constants, then the chase procedure stops and outputs a failure notice. Otherwise, if one of a_i is a null, say \perp , and the other one is either a constant or a null, say c , then A_{i+1} is obtained from A_i by replacing every occurrence of \perp by c . This finishes step i , and the algorithm proceeds with step $i + 1$.

Case 2: $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ contains an s-t-tgd or a t-tgd of the form

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

for which there are tuples \bar{a} and \bar{b} of elements in Dom such that each of the conjuncts in $\varphi(\bar{a}, \bar{b})$ belongs to A_i , but there is no tuple \bar{c} over Dom such that each of the conjuncts in $\psi(\bar{a}, \bar{c})$ belongs to A_i . Then let \bar{c} be a tuple of pairwise distinct nulls that do not occur in A_i , and let A_{i+1} be the union of A_i with all conjuncts in $\psi(\bar{a}, \bar{c})$. This finishes step i , and the algorithm proceeds with step $i + 1$.

It is shown in [14] that weak acyclicity of M enforces that the chase procedure stops after a number of steps that is polynomial in the size of the source instance S . If the output of the procedure is a failure notice, then there exists no solution for S under M ; otherwise the output of the procedure is a *universal* solution for S under M .

Solutions produced by the chase procedure are sometimes referred to as *canonical* universal solutions. For example, the universal solution T_4 from Example 3.3 is a canonical universal solution.

It is known that for weakly acyclic mappings also the *core* (i.e., the “smallest” universal solution) can be computed in polynomial time — the algorithm, however, is much more involved than the chase procedure.

Theorem 4.6 (Gottlob, Nash [21]). For every weakly acyclic schema mapping $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$, there is a polynomial-time algorithm which, given a source instance S , tests whether a solution for S under M exists and, if so, it produces $\text{CORE}_M(S)$. \square

Recently, Deutsch, Nash, and Rimmel [11] introduced a generalization of the notion of weakly acyclic schema mappings, the so-called *stratified* schema mappings, and showed that the chase procedure can be used to generalize Theorem 4.5 to the class of all stratified schema mappings.

Note, however, that there also exist schema mappings which are not stratified, but for which solutions still can be computed efficiently:

Example 4.7. Let $M = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ be the schema mapping from Example 3.4. From Example 4.4 we know that M is not weakly acyclic. In fact, M is not stratified in the sense of [11]. However, it is not difficult to see that (a) solutions always exist (and can be computed easily), and (b) universal solutions exist for a source instance S if, and only if, the target instance T obtained from S by renaming F into E (i.e., $E^T := F^S$) satisfies the t-tgd in Σ_t . \square

It remains an important future task to identify further classes of schema mappings for which solutions can be computed efficiently.

5 Query answering in data exchange

Another important question in the area of data exchange is: What are the “right” semantics for answering queries that are formulated with respect to the target schema, if only information about the source instance and the schema mapping is available? More precisely: Given a schema mapping M , a source instance S for M , and a query q that is formulated with respect to the target schema, what does “answering q with respect to S and M ” mean? Let us illustrate this with an example.

Example 5.1. Let $M_{airline}$ and $S_{airline}$ be the schema mapping and the source instance from Example 2.7 and 2.8, respectively. Let q be the query that asks for all tuples (c_1, c_2) of cities that are connected by a direct flight of the new airline. This query can be formalized in first-order logic via

$$q(x_1, x_2) := \exists z \text{ New}(x_1, x_2, z).$$

For a target instance T we write $q(T)$ to denote the set of tuples (c_1, c_2) of cities for which the formula $q(x_1, x_2)$ is satisfied in T when interpreting the variables x_1, x_2 with the constants c_1, c_2 . We have already seen that several solutions for $S_{airline}$ under $M_{airline}$ exist — and evaluating the query q on different solutions may lead to different query answers. Considering, e.g., the solutions T_1 and T_2 from Example 2.8, the tuple (CDG, TXL) belongs to $q(T_1)$, but not to $q(T_2)$. \square

In general, if M , S , and q are given, and q is evaluated on two different solutions T_1 and T_2 for S under M , it is possible that two different sets of answers, $q(T_1)$ and $q(T_2)$, are produced. But what should then be the precise semantics of query answering in data exchange? Fagin, Kolaitis, Miller, and Popa [14] proposed to adopt the concept of *certain answers* that had already been successfully used in the area of *incomplete databases* (see, e.g., [1, 25]). Precisely, the set of *certain answers of q on S with respect to M* is the set⁵

$$\text{certain}^M(q, S) := \bigcap \{ q(T) : T \text{ is a solution for } S \text{ under } M \}.$$

⁵ If M is a set whose elements are sets, we write $\bigcap M$ to denote the set $\bigcap_{N \in M} N$.

I.e., the set of certain answers contains only those tuples that belong to $q(T)$, for every solution T for S under M .

Example 5.2. It is not difficult to see that for the query q from Example 5.1, the set $\text{certain}^{M_{\text{airline}}}(q, S_{\text{airline}})$ consists of the following tuples: (AMS, TXL), (FRA, AMS), (TXL, TXL), (AMS, AMS). \square

In [14], the authors identified a large class of queries, for which the set $\text{certain}^M(q, S)$ is determined by an arbitrary universal solution: the *unions of conjunctive queries*. These queries are disjunctions of formulas of the form $\exists \bar{y} \varphi(\bar{x}, \bar{y})$, where \bar{x} and \bar{y} are (possibly empty) finite tuples of first-order variables, and $\varphi(\bar{x}, \bar{y})$ is a conjunction of atoms of the form $R(u_1, \dots, u_r)$, where $R \in \tau$, r is the arity of R , and each u_i is either one of the variables in \bar{x}, \bar{y} or an element in Const .

Theorem 5.3 (Fagin, Kolaitis, Miller, Popa [14]). If $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ is a schema mapping, S is a source instance for M , q is a union of conjunctive queries with respect to the target schema τ , and T is an arbitrary universal solution for M under S , then

$$\text{certain}^M(q, S) = q(T)_{\downarrow},$$

where $q(T)_{\downarrow}$ is the set of all “null-free” tuples in $q(T)$ (i.e., tuples \bar{c} , where each component in \bar{c} is an element in Const). \square

Together with Theorem 4.5 (b) this, in particular, tells us that for every *weakly acyclic* schema mapping M and every query q that is a union of conjunctive queries, there is a polynomial-time algorithm which, given a source instance S , computes the certain answers $\text{certain}^M(q, S)$.

In [16], the authors introduced a variant of the certain answers semantics, the *certain universal answers*, which are defined as follows:

$$\text{certain}_{\text{univ}}^M(q, S) := \bigcap \{ q(T) : T \text{ is a universal solution for } S \text{ under } M \}.$$

It is obvious that $\text{certain}^M(q, S)$ is a subset of $\text{certain}_{\text{univ}}^M(q, S)$. From Theorem 5.3 it immediately follows that $\text{certain}^M(q, S) = \text{certain}_{\text{univ}}^M(q, S)$ if q is a union of conjunctive queries. Furthermore, it has been shown in [16] that $\text{CORE}_M(S)$ can be used to compute $\text{certain}_{\text{univ}}^M(q, S)$ if q is an existential query (that may include negation).

However, it has been noted (cf., e.g., [3, 28]) that the certain answers semantics and the certain universal answers semantics give rise to some anomalies where the semantics behave in a counterintuitive way, i.e., queries produce results that intuitively do not seem to be accurate. Let us illustrate this with the following example.

Example 5.4. Consider a variant of the airline setting $M_{airline}$ from Example 2.7, namely the setting $M'_{airline} = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ where σ and τ are chosen in the same way as in $M_{airline}$, $\Sigma_{st} = \{\delta'_1, \delta'_2\}$, and $\Sigma_t = \emptyset$, where

$$\begin{aligned}\delta'_1 &:= \forall x_1 \forall x_2 \forall y (KLA(x_1, x_2, y) \rightarrow \exists z New(x_1, x_2, z)), \\ \delta'_2 &:= \forall x_1 \forall x_2 \forall y (AF(x_1, x_2, y) \rightarrow \exists z New(x_1, x_2, z)).\end{aligned}$$

Intuitively, the schema mapping $M'_{airline}$ tells us that the new airline simply takes the union of all the connections of the two airlines *KLA* and *Air Flight* and replaces the information on flight numbers with information on aircraft types. Now consider the query

$$q(x) := \exists y_1 \exists z_1 (New(x, y_1, z_1) \wedge \forall y_2 \forall z_2 (New(x, y_2, z_2) \rightarrow y_2 = y_1)),$$

which asks for all cities x from which exactly one city can be reached by a direct flight of the new airline. Intuitively, one would expect that for the source instance $S_{airline}$ from Example 2.8, the query $q(x)$ should yield the results AMS and FRA.

The certain answers $\text{certain}^{M'_{airline}}(q, S_{airline})$ and the certain universal answers $\text{certain}_{\text{univ}}^{M'_{airline}}(q, S_{airline})$, however, yield as result the empty set. To see why, note that the two target instances T'_1 and T'_2 with

$$\boxed{\begin{array}{l} New^{T'_1} := \\ \{ (AMS, TXL, \perp_1), \\ (FRA, AMS, \perp_2), \\ (CDG, TXL, \perp_3), \\ (CDG, AMS, \perp_4) \} \end{array}} \quad \text{and} \quad \boxed{\begin{array}{l} New^{T'_2} := New^{T'_1} \cup \\ \{ (AMS, \perp_5, \perp_6), \\ (FRA, \perp_7, \perp_8) \} \end{array}}$$

(and $\perp_i \neq \perp_j$ whenever $i \neq j$) are universal solutions for $S_{airline}$ under $M'_{airline}$, and that $q(T'_2) = \emptyset$. \square

The same counterintuitive behaviour of the certain (universal) answers semantics occurs in so-called *copying scenarios*, i.e., schema mappings $M = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ where $\sigma = \{R_1, \dots, R_n\}$, $\tau = \{R'_1, \dots, R'_n\}$, $\Sigma_t = \emptyset$, and Σ_{st} contains, for each relation symbol R_i in σ , a s-t-tgd of the form $\forall \bar{x} (R_i(\bar{x}) \rightarrow R'_i(\bar{x}))$. Intuitively, such a schema mapping just changes the names of the relation symbols. One would therefore expect that a query q that is formulated with respect to the target schema, can be rewritten into an equivalent query \tilde{q} over the source schema by just replacing each occurrence of a relation symbol R'_i by its old name R_i . Thus, one would expect that evaluating q on “good” solutions for S under M yields the same result as evaluating \tilde{q} on S . However, for queries similar to query q from Example 5.4, one obtains that $\text{certain}_{\text{univ}}^M(q, S) = \emptyset$ although $\tilde{q}(S)$ is non-empty.

In [28], Libkin observed that the reason for this counterintuitive behaviour is the fact that the presence of “incomplete” information (i.e., nulls) in target instances is ignored, and the certain answers semantics and the certain universal answers semantics are defined with respect to sets of solutions treating nulls in the same way as constants.

6 CWA-solutions

At the end of Section 5 we pointed out that the certain answers semantics and the certain universal answers semantics give rise to some anomalies where the semantics behave in a counterintuitive way, i.e., queries produce results that intuitively do not seem to be accurate. To resolve these anomalies, let us again consider the question “*Which solutions are particularly “good” solutions?*”.

Intuitively, “good” solutions are those solutions that carry *no more and no less* information than required by S and M . — And for defining the semantics of queries, only such solutions should be taken into account. In [28], Libkin formulated the following three requirements that such solutions should meet, and he used the name *CWA-solutions* to address such solutions.⁶

- (1) The presence of each atom in a CWA-solution is justified by the source instance and the dependencies in the schema mapping.
- (2) Justifications should not be overused. I.e., each justification for introducing atoms does not generate more atoms than necessary.
- (3) Each fact that is true in a CWA-solution follows from the source instance and the dependencies in the schema mapping.⁷

The name “CWA-solution” is chosen due to the fact that CWA-solutions are based on the *closed world assumption* (cf., [32]) in the sense that all facts which are true in a CWA-solution must be justified by the source instance and the dependencies in the schema mapping.

For schema mappings $M = (\sigma, \tau, \Sigma_{st}, \Sigma_t)$ where $\Sigma_t = \emptyset$, the requirements (1)–(3) were formalized by Libkin in [28]. A formalization for the general case (where Σ_t is not necessarily empty) was given by Hernich and Schweikardt in [24]. This formalization is explained in the next two subsections; for simplicity in presentation, however, we will restrict attention to the special case where Σ_t contains no egds (details on the general case where Σ_t may contain egds as well as t-tgds can be found in [24]).

⁶Similarly as in Section 4, we identify an instance I with the set $atoms(I)$ of atoms in I .

⁷ I.e., CWA-solutions are not allowed to contain “invented” facts that cannot be inferred by the schema mapping and the source instance.

6.1 Formalization of requirements (1) and (2)

The first two requirements can be formalized in terms of a game. Let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_t)$ be a schema mapping, let S be a source instance for M , and let T be a target instance for M . For the description of the game we also need the set

$$\mathcal{J}_M := \left\{ (\delta, \bar{a}, \bar{b}) : \delta \text{ is a s-t-tgd in } \Sigma_{\text{st}}, \text{ or a t-tgd in } \Sigma_t, \text{ of the form} \right. \\ \left. \forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})), \text{ and } \bar{a} \text{ and } \bar{b} \text{ are} \right. \\ \left. \text{interpretations of } \bar{x} \text{ and } \bar{y} \text{ with elements in } \text{Dom} \right\}.$$

An element $(\delta, \bar{a}, \bar{b}) \in \mathcal{J}_M$, where $\delta = \forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$, can be used to *justify* the presence (in T) of each of the conjuncts in $\psi(\bar{a}, \bar{c})$ (for some interpretation \bar{c} for the variables in \bar{z}) with (i) the presence (in $S \cup T$) of each of the conjuncts in $\varphi(\bar{a}, \bar{b})$, and (ii) the fact that the presence of the conjuncts in $\psi(\bar{a}, \bar{c})$ is a possible consequence of the presence of the conjuncts in $\varphi(\bar{a}, \bar{b})$ with respect to δ .

The game, denoted by $\mathcal{G}(M, S, T)$, is played by two players, the *verifier* and the *falsifier*. The verifier’s goal is to show that T satisfies the requirements (1) and (2) with respect to S and M , whereas the falsifier’s goal is to show the converse. The game has at most $|\text{atoms}(T)| + 1$ rounds.

In round 0, the verifier fixes the two sets $A_1 := \text{atoms}(T)$ and $\mathcal{J}_1 := \mathcal{J}_M$ and picks a linear order \prec on A_1 . Intuitively, A_1 is the set of atoms that need to be justified, and \mathcal{J}_1 is the set of potential justifications that may be used for this purpose. The linear order \prec determines that for justifying an atom $R(\bar{u})$, only atoms that either belong to S or are smaller than $R(\bar{u})$ (w.r.t. \prec) can be used.

For each $i \geq 1$, round i then proceeds as follows: First, the falsifier picks an atom $R(\bar{u})$ in A_i (or loses if A_i is empty). Then, the verifier has to “justify” $R(\bar{u})$ by picking a tuple $(\delta, \bar{a}, \bar{b}) \in \mathcal{J}_i$, where δ has the form $\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$, and an interpretation \bar{c} of \bar{z} with elements in Dom such that:

- $S \cup T \models \varphi(\bar{a}, \bar{b})$, $T \models \psi(\bar{a}, \bar{c})$, and $R(\bar{u})$ is a conjunct in $\psi(\bar{a}, \bar{c})$, and
- for each conjunct $R'(\bar{u}')$ in $\varphi(\bar{a}, \bar{b})$, we either have $R'(\bar{u}') \in \text{atoms}(S)$, or $R'(\bar{u}') \prec R(\bar{u})$.

If this is not possible, the verifier loses. Otherwise, the game proceeds with round $i + 1$, where A_{i+1} consists of all atoms in A_i that do not occur as a conjunct in $\psi(\bar{a}, \bar{c})$, and $\mathcal{J}_{i+1} := \mathcal{J}_i \setminus \{(\delta, \bar{a}, \bar{b})\}$.

Note that by removing all conjuncts in $\psi(\bar{a}, \bar{c})$ from A_i , we ensure that these conjuncts do not need to be justified again (they are already justified by $(\delta, \bar{a}, \bar{b})$). By removing $(\delta, \bar{a}, \bar{b})$ from the set \mathcal{J}_i , we ensure that every justification is used at most once (this corresponds to requirement (2)).

A target instance T for M is called *CWA-presolution* for S under M if T is a solution for S under M and the verifier has a winning strategy in the game $\mathcal{G}(M, S, T)$. Intuitively, the CWA-presolutions are precisely the solutions that meet the requirements (1) and (2).

Example 6.1. Consider the schema mapping M_{airline} from Example 2.7 and the source instance S_{airline} from Example 2.8. Then, the universal solution T_4 from Example 3.3 is a CWA-presolution for S_{airline} under M_{airline} . Indeed, it is not difficult to see that the verifier can win the game $\mathcal{G}(M_{\text{airline}}, S_{\text{airline}}, T_4)$ by picking, in round 0 a linear order \prec on $\text{atoms}(T_4)$ such that the atoms $\text{New}(\perp_4, \perp_4, \perp_{12})$ and $\text{New}(\perp_7, \perp_7, \perp_{13})$ are the two largest atoms with respect to \prec .

On the other hand, the solutions T_5 and T_6 with

$$\begin{aligned} \text{New}^{T_5} &:= \text{New}^{T_4} \cup \{(\text{LHR}, \text{GVA}, \perp_{14}), (\text{GVA}, \text{GVA}, \perp_{15})\}, \\ \text{New}^{T_6} &:= \text{New}^{T_4} \cup \{(\text{AMS}, \text{TXL}, \perp'_1)\} \end{aligned}$$

are no CWA-presolutions for S_{airline} under M_{airline} . For example, to win the game $\mathcal{G}(M_{\text{airline}}, S_{\text{airline}}, T_5)$, the falsifier simply picks, in round 1, the atom $\text{New}(\text{LHR}, \text{GVA}, \perp_{14})$, which the verifier cannot justify. To win the game $\mathcal{G}(M_{\text{airline}}, S_{\text{airline}}, T_6)$, the falsifier can choose $\text{New}(\text{AMS}, \text{TXL}, \perp_1)$ in round 1, and $\text{New}(\text{AMS}, \text{TXL}, \perp'_1)$ in round 2. \square

A different, but equivalent, definition of CWA-presolutions relies on the α -chase procedure [24]. For simplicity, we here only describe the special case of schema mappings without egds (see [24] for details on how to handle egds). Let α be a mapping that maps every tuple $(\delta, \bar{a}, \bar{b}) \in \mathcal{J}_M$, where δ has the form $\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$, to an interpretation \bar{c} of \bar{z} with elements in Dom . The α -chase procedure is defined in the same way as the chase procedure described in Section 4, except for the following: Suppose that in step i , there is a s-t-tgd or a t-tgd δ in $\Sigma_{\text{st}} \cup \Sigma_{\text{t}}$ of the form $\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$ for which there are tuples \bar{a} and \bar{b} of elements in Dom such that each of the conjuncts in $\varphi(\bar{a}, \bar{b})$ belongs to A_i , and at least one of the conjuncts in $\psi(\bar{a}, \bar{c})$, where $\bar{c} := \alpha(\delta, \bar{a}, \bar{b})$, does not belong to A_i . Then, A_{i+1} is the union of A_i with all conjuncts in $\psi(\bar{a}, \bar{c})$, and the procedure continues with step $i + 1$. The procedure stops only if in some step there is no such s-t-tgd or t-tgd.

It is not difficult to verify that for every solution T for S under M , the following is true: There is a mapping α such that the α -chase procedure outputs T if, and only if, the verifier has a winning strategy in the game $\mathcal{G}(M, S, T)$.

6.2 CWA-solutions

To formalize requirement (3) and thus complete the definition of CWA-solutions, we need the following notion: A *fact* (over a schema τ) is a

Boolean conjunctive query over τ , i.e., a first-order sentence of the form $\exists \bar{x} \varphi(\bar{x})$, where \bar{x} is a finite sequence of first-order variables and $\varphi(\bar{x})$ is a conjunction of atoms of the form $R(u_1, \dots, u_r)$ where $R \in \tau$, r is the arity of R , and each u_i is either one of the variables in \bar{x} or a constant (i.e., an element in Const). An example of a fact over the target schema of the schema mapping M_{airline} in Example 2.7 is $\exists x (\text{New}(\text{FRA}, \text{AMS}, x) \wedge \text{New}(\text{AMS}, \text{TXL}, x))$. It expresses that the same aircraft type can be used to fly from FRA to AMS and from AMS to TXL.

Definition 6.2 (CWA-solution [28, 24]). Let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, and let S be a source instance for M . A *CWA-solution* for S under M is a CWA-presolution T for M such that each fact that is true in T , is true in *every* solution for S under M . \square

Example 6.3. Consider, once again, the schema mapping M_{airline} from Example 2.7 and the source instance S_{airline} from Example 2.8. Then, both, T_3 and T_4 in Example 3.3 are CWA-solutions for S_{airline} under M_{airline} .

On the other hand, the solutions T_1 and T_2 for S_{airline} from Example 2.8, and the solutions T_5 and T_6 from Example 6.1 are no CWA-solutions, since they are not even CWA-presolutions for S_{airline} under M_{airline} .

Finally, let T_7 be the solution for S_{airline} that is obtained from T_4 by replacing the value \perp_2 with \perp_1 . Then T_7 is a CWA-presolution. However, it is no CWA-solution for S_{airline} under M_{airline} , since, for example, the fact $\exists x (\text{New}(\text{FRA}, \text{AMS}, x) \wedge \text{New}(\text{AMS}, \text{TXL}, x))$ is true in T_7 , but it is not true in T_4 . \square

The next theorem draws the connection between CWA-solutions and the universal solutions from Section 3.

Theorem 6.4 (Libkin [28], Hernich and Schweikardt [24]).

Let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, and let S be a source instance for M . Then the following is true:

(a) For every target instance T for M , the following are equivalent:

- T is a CWA-solution for S under M .
- T is a universal solution for S under M , and T is a CWA-presolution for S under M .

(b) If $\text{CORE}_M(S)$ exists, then $\text{CORE}_M(S)$ is a CWA-solution for S under M . \square

In particular, Theorem 6.4 implies that if $\text{CORE}_M(S)$ exists, then it is the unique “minimal” CWA-solution for S under M , in the sense that every CWA-solution T for S under M has a sub-instance that is isomorphic

to $\text{CORE}_M(S)$. If M contains no target dependencies, then there also is a unique CWA-solution T_{\max} , which is “maximal” in the sense that every CWA-solution T for S under M is a homomorphic image of T_{\max} [28]. However, [24] gives an example of a schema mapping M with $\Sigma_t \neq \emptyset$, where there is, for each $n \in \mathbb{N}$, a source instance S_n of size $O(n)$ for which (at least) 2^n “maximal” CWA-solutions T_1, T_2, \dots, T_{2^n} exist, each of which is (up to the renaming of nulls) no homomorphic image of another CWA-solution. See Figure 3 for an illustration of the space of CWA-solutions.

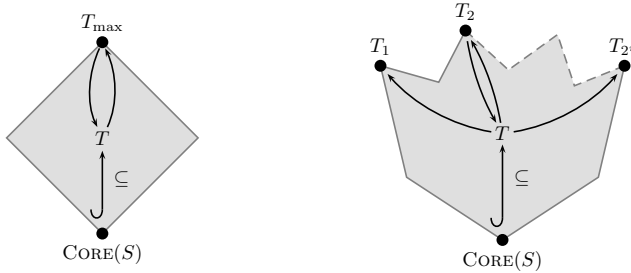


FIGURE 3. A representation of the space of CWA-solutions for the case that M has no target dependencies (left), and for the case that M has target dependencies (right). The arcs indicate homomorphisms.

6.3 How to compute CWA-solutions

Given a schema mapping M and a source instance S for M , how can we decide whether a CWA-solution for S under M exists — and if so, how can we compute a CWA-solution?

First of all, note that from Theorem 6.4 we know that for every schema mapping $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_t)$ and every source instance S for M , the following are equivalent: (i) there is a CWA-solution for S under M , (ii) there is a universal solution for S under M , (iii) $\text{CORE}_M(S)$ exists. Theorem 4.2 thus implies that there is a schema mapping $M'' = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_t)$ such that the problem

EXISTENCE-OF-CWA-SOLUTIONS(M'')

Input: A source instance S for M'' .

Question: Is there a CWA-solution for S under M'' ?

is undecidable.

On the other hand, if M is a weakly acyclic schema mapping, then, by Theorem 4.5, there is a polynomial-time algorithm which, given a source instance S for M , tests whether a universal solution for S under M exists,

and if so, computes a universal solution for S . Since a CWA-solution for S exists if and only if a universal solution for S exists, the same algorithm can be applied to test whether there is a CWA-solution for S . However, it is easy to construct weakly acyclic schema mappings for which the algorithm’s result is in general no CWA-solution. Still, if CWA-solutions for S under M exist, we can compute a CWA-solution for S under M , namely $\text{CORE}_M(S)$, using the polynomial-time algorithm from Theorem 4.6.

Similarly as with Section 4, it is an important future task to identify broader classes of schema mappings for which CWA-solutions can be computed efficiently.

6.4 Query answering using CWA-solutions

Let us now turn back to the task of query answering. In the following, we will introduce the CWA-solution-based query answering semantics that have been proposed by Libkin [28] in order to get rid of the anomalies pointed out at the end of Section 5.

Let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, let S be a source instance for M , and let q be a query over τ . To obtain the result of q under any of the semantics from [28], one takes the intersection or the union of the answers to q on CWA-solutions for S . To give the precise definition, we first explain how q is answered on a single CWA-solution for S .

Notice that a CWA-solution T for S under M may contain nulls, which are placeholders for existent, but unknown (constant) values (see Section 3). That is, T represents a number of solutions \hat{T} for S without nulls, and these solutions can be obtained from T by replacing every null with a constant in Dom . Let $\text{Rep}_M(T)$ denote the set of all solutions \hat{T} for S under M that are possibly represented by T , i.e., all solutions \hat{T} for S under M which can be obtained from T by replacing every null with an element in Const . The *certain answers* to q on T are then defined as follows (cf., e.g., [1, 25]):

$$\square_M q(T) := \bigcap \{q(\hat{T}) : \hat{T} \in \text{Rep}_M(T)\}.$$

This results in the *certain CWA answers semantics* and the *potential certain CWA answers semantics*, respectively, which are defined as

$$\begin{aligned} \text{certain}_{\square}^M(q, S) &:= \bigcap \{\square_M q(T) : T \text{ is a CWA-solution for } S \text{ under } M\}, \\ \text{certain}_{\diamond}^M(q, S) &:= \bigcup \{\square_M q(T) : T \text{ is a CWA-solution for } S \text{ under } M\}. \end{aligned}$$

I.e., the set of *certain CWA answers* $\text{certain}_{\square}^M(q, S)$ contains only those tuples that appear as certain answers for *all* CWA-solutions, whereas the set of *potential certain CWA answers* $\text{certain}_{\diamond}^M(q, S)$ contains those tuples that appear as certain answers for at least one CWA-solution.

Example 6.5. Recall the source instance S_{airline} from Example 2.8 and the schema mapping M'_{airline} and the solution T'_1 from Example 5.4. It is not difficult to see that T'_1 is a CWA-solution. In fact, there are only two different CWA-solutions, namely T'_1 and the solution T'_3 with $\text{New}^{T'_3} := \text{New}^{T'_1} \cup \{(\text{FRA}, \text{AMS}, \perp_5)\}$. Therefore, for the query q from Example 5.4, the certain CWA answers $\text{certain}_{\square}^{M'_{\text{airline}}}(q, S_{\text{airline}})$ and the potential certain CWA answers $\text{certain}_{\diamond}^{M'_{\text{airline}}}(q, S_{\text{airline}})$ yield the results AMS and FRA, as intuitively expected. \square

Furthermore, note that for any *copying scenario* M (cf., the end of Section 5), each source instance S has a unique CWA-solution, namely its “copy” T_S which is obtained from S by renaming each relation R_i into R'_i (i.e., $R'_i{}^{T_S} := R_i^S$). Therefore, for each query q that is formulated with respect to the target schema, one obtains that $\text{certain}_{\square}^M(q, S) = \text{certain}_{\diamond}^M(q, S) = \tilde{q}(S)$, as one intuitively expects (here, \tilde{q} is the query obtained from q by replacing each occurrence of R'_i with R_i).

The following theorem summarizes some of the basic properties of the new CWA-solution-based query semantics.

Theorem 6.6 (Libkin [28], Hernich and Schweikardt [24]).

Let $M = (\sigma, \tau, \Sigma_{\text{st}}, \Sigma_{\text{t}})$ be a schema mapping, and let S be a source instance for M such that S has at least one CWA-solution under M . Then the following is true for each query q over τ :

- (a) $\text{certain}_{\diamond}^M(q, S) = \square_M q(\text{CORE}_M(S))$.
- (b) If M has no target dependencies, then $\text{certain}_{\square}^M(q, S) = \square_M q(T_{\text{max}})$, where T_{max} is the unique “maximal” CWA-solution for S under M mentioned at the end of Section 6.2.
- (c) If q is preserved under homomorphisms (e.g., a union of conjunctive queries), then $\text{certain}_{\square}^M(q, S) = \text{certain}_{\diamond}^M(q, S) = \text{certain}^M(q, S)$. \square

In particular, for a weakly acyclic schema mapping M and a query q that is a union of conjunctive queries, Theorem 6.6 (c), Theorem 5.3, and Theorem 4.5 (b) tell us that there is a polynomial-time algorithm which, given a source instance S for M , computes $\text{certain}_{\square}^M(q, S)$ (resp., $\text{certain}_{\diamond}^M(q, S)$).

However, moving from unions of conjunctive queries to more expressive query languages, the data complexity⁸ quickly increases to co-NP-hard. Table 1 summarizes what is known about the data complexity (with respect to certain_{\square} and $\text{certain}_{\diamond}$) of first-order queries (FO), unions of conjunctive

⁸ Here, the term *data complexity* refers to the fact that M and q are assumed to be fixed, and the complexity is measured only in terms of the size of the source instance.

queries (UCQ), and unions of conjunctive queries with at most one inequality per disjunct. Here, *richly acyclic* schema mappings [24] are particular weakly acyclic mappings; *full tgds* are s-t-tgds or t-tgds in which no existential quantifier occurs, i.e., they are of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y}) \rightarrow \psi(\bar{x}))$.

		query language		
		UCQ	UCQ with at most one inequality per disjunct	FO
restriction of schema mapping	weakly acyclic			co-NP-hard
	richly acyclic		co-NP-complete	
	only s-t-tgds, egds	PTIME		
	only full tgds, egds			

TABLE 1. Data complexity of computing $\text{certain}_{\square}^M(q, S)$ and $\text{certain}_{\diamond}^M(q, S)$ for certain restrictions of M and q [24].

7 Further topics

In this paper we gave a brief introduction into the area of data exchange, with a special emphasis on the question “*Which solutions are “good” solutions?*”. Of course, there is a long list of topics that have not been discussed in this paper.

First of all, we should note that it is still not clear which solutions can be considered to be “the best” solutions. Even for the CWA-solution-based query semantics it is possible to construct examples where the semantics behave in a counterintuitive way. For this reason, Libkin and Sirangelo [29] recently proposed a notion of solutions that is a mixture between the solutions from Section 2 and the CWA-solutions from Section 6. Moreover, Afrati and Kolaitis [2] argued that, for answering so-called aggregate queries, one needs a rather strong restriction of CWA-solutions.

A lot of work in data exchange has been done concerning Bernstein’s *metadata management* framework [8], in particular, on the composition of schema mappings [17, 29] and the inverse of a schema mapping [13, 18, 6]. Moreover, [15] studied the problem of *schema mapping optimization*, which is closely related to metadata management.

The *complexity of query answering* for various query languages has been studied, e.g., in [14, 16, 30, 27, 4]. Closely related to the query answering problem is the *query rewriting* problem [3], dealing with the following goal: Given a query q , find a query q' , called *rewriting* of q , such that the result of q' over a materialized solution T is equivalent to the result of q

with respect to some particular query answering semantics, e.g., the certain answers semantics or the certain CWA answers semantics.

In the present paper, we restricted attention on the problem of exchanging *relational* data from a *single* source to a *single* target. Of course, several variations are conceivable. Arenas and Libkin [5], for example, considered the problem of *XML data exchange*. And the exchange of data with *multiple* sources and targets is the subject of *peer data exchange* [9, 19, 20].

Let us conclude by pointing out two directions for future research. Certainly, it remains an important future task to identify broader classes of schema mappings, for which “good” solutions can be computed efficiently. Furthermore, as described at the beginning of Section 7, the question “*Which solutions are “good” solutions?*” has still not been fully answered. In the end, it seems that, for different types of schema mappings and different types of queries that one wants to support, different notions of “good” solutions need to be developed.

Bibliography

- [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 *Proc. PODS’08*, pages 129–138, 2008.
- [3] M. Arenas, P. Barceló, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *Proc. PODS’04*, pages 229–240, 2004.
- [4] M. Arenas, P. Barceló, and J. Reutter. Query languages for data exchange: Beyond unions of conjunctive queries. In *Proc. ICDT’09*, pages 73–83, 2009.
- [5] M. Arenas and L. Libkin. XML data exchange: consistency and query answering. In *Proc. PODS’05*, pages 13–24, 2005.
- [6] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: Bringing exchanged data back. In *Proc. PODS’08*, pages 13–22, 2008.
- [7] P. Barceló. Logical foundations of relational data exchange. *SIGMOD Record*, 38(1):49–58, 2009.
- [8] P. A. Bernstein. Applying model management to classical meta-data problems. In *Proc. CIDR’03*, pages 209–220, 2003.

- [9] L. E. Bertossi and L. Bravo. Query answering in peer-to-peer data exchange systems. In *EDBT Workshops*, pages 476–485, 2004.
- [10] B. ten Cate and P. G. Kolaitis. Structural characterizations of schema-mapping languages. In *Proc. ICDT’09*, 2009.
- [11] A. Deutsch, A. Nash, and J. B. Remmel. The chase revisited. In *Proc. PODS’08*, pages 149–158, 2008.
- [12] A. Deutsch and V. Tannen. Reformulation of XML queries and constraints. In *Proc. ICDT’03*, pages 225–241, 2003.
- [13] R. Fagin. Inverting schema mappings. In *Proc. PODS’06*, pages 50–59, 2006.
- [14] 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. Full version of an ICDT’03 paper.
- [15] R. Fagin, P. G. Kolaitis, A. Nash, and L. Popa. Towards a theory of schema-mapping optimization. In *Proc. PODS’08*, pages 33–42, 2008.
- [16] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *ACM Transactions on Database Systems*, 30(1):174–210, 2005. Full version of a PODS’03 paper.
- [17] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. In *Proc. PODS’04*, pages 83–94, 2004.
- [18] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Quasi-inverses of schema mappings. In *Proc. PODS’07*, pages 123–132, 2007.
- [19] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan. Peer data exchange. *ACM Transactions on Database Systems*, 31(4):1454–1498, 2006. Full version of a PODS’05 paper.
- [20] G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. On reconciling data exchange, data integration, and peer data management. In *Proc. PODS’07*, pages 133–142, 2007.
- [21] G. Gottlob and A. Nash. Efficient core computation in data exchange. *Journal of the ACM*, 55(2):Article 9, May 2008. Full version of PODS’05 and PODS’06 papers.
- [22] L. M. Haas, M. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: from research prototype to industrial tool. In *Proc. SIGMOD’05*, pages 805–810, 2005.

- [23] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109(1-3):117–126, 1992.
- [24] A. Hernich and N. Schweikardt. CWA-solutions for data exchange settings with target dependencies. In *Proc. PODS'07*, pages 113–122, 2007.
- [25] T. Imielinski and W. Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.
- [26] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. PODS'05*, pages 61–75, 2005.
- [27] P. G. Kolaitis, J. Panttaja, and W. C. Tan. The complexity of data exchange. In *Proc. PODS'06*, pages 30–39, 2006.
- [28] L. Libkin. Data exchange and incomplete information. In *Proc. PODS'06*, pages 60–69, 2006.
- [29] L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *Proc. PODS'08*, pages 139–148, 2008.
- [30] A. Małdry. Data exchange: On the complexity of answering queries with inequalities. *Information Processing Letters*, 94:253–257, 2005.
- [31] R. J. Miller, L. M. Haas, and M. Hernández. Schema mapping as query discovery. In *Proc. VLDB'00*, pages 77–89, 2000.
- [32] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 119–140. Plenum Publ. Co., New York, 1978.