ANSWERING NON-MONOTONIC QUERIES
IN RELATIONAL DATA EXCHANGE [*]

ANDRÉ HERNICH

Institut für Informatik, Humboldt-Universtität, Berlin, Germany
e-mail address: hernich@informatik.hu-berlin.de

ABSTRACT. Relational data exchange is the problem of translating relational data from a source schema into a target schema, according to a specification of the relationship between the source data and the target data. One of the basic issues is how to answer queries that are posed against target data. While consensus has been reached on the definitive semantics for *monotonic queries*, this issue turned out to be considerably more difficult for *non-monotonic queries*. Several semantics for non-monotonic queries have been proposed in the past few years.

This article proposes a new semantics for non-monotonic queries, called the *GCWA*-semantics*. It is inspired by semantics from the area of deductive databases. We show that the GCWA*-semantics coincides with the standard open world semantics on monotonic queries, and we further explore the (data) complexity of evaluating non-monotonic queries under the GCWA*-semantics. In particular, we introduce a class of schema mappings for which universal queries can be evaluated under the GCWA*-semantics in polynomial time (data complexity) on the core of the universal solutions.

1. INTRODUCTION

Data exchange is the problem of translating databases from a source schema into a target schema, whereby providing access to the source database through a materialized database over the target schema. It is a special case of data integration [28] and arises in tasks like data restructuring, updating data warehouses using ETL processes, or in exchanging data between different, possibly independently created, applications (see, e.g., [18, 10]). Tools for dealing with data exchange are available for quite a while [37, 18, 34]. Fundamental concepts and algorithmic issues in data exchange have been studied recently by Fagin, Kolaitis, Miller, and Popa in their seminal paper [10]. For a comprehensive overview on data exchange, the reader is referred to [10] or any of the surveys [26, 6, 24, 4].

This article deals with *relational* data exchange, which received a lot of attention in the data exchange community (see, e.g., the survey articles cited above). In this setting, the mapping from source data to target data is described by a *schema mapping* $M = (\sigma, \tau, \Sigma)$ which consists of relational database schemas σ and τ (finite sets of relation names with

associated arities), called *source schema* and *target schema*, respectively, and a finite set Σ of constraints (typically, sentences in some fragment of first-order logic) which can refer to the relation names in σ and τ. Typical constraints are tuple generating dependencies (tgds), which come in two flavors – st-tgds and t-tgds –, and equality generating dependencies (egds). For example, st-tgds are first-order sentences of the form $\forall \bar{x}, \bar{y} \left(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z} \, \psi(\bar{x}, \bar{z}) \right)$, where φ is a conjunction of relation atoms over σ, and ψ is a conjunction of relation atoms over τ. Their precise definitions are deferred to Section 2.3. Given a relational database instance S over σ (called *source instance* for M), a *solution* for S under M is a relational database instance T over τ such that the instance $S \cup T$ over $\sigma \cup \tau$ that consists of the relations in S and T satisfies all the constraints in Σ.

An important task in relational data exchange is to answer queries that are posed against the target schema of a schema mapping. The answer to a query should be semantically consistent with the source data and the schema mapping, that is, it should reflect the information in the source instance and the schema mapping as good as possible. Since a source instance usually has more than one solution, a fundamental question is: What is the semantics of a query, that is, which tuples constitute the set of answers to a query over the target schema of a schema mapping and a given source instance? Furthermore, in data exchange the goal is to answer queries using a materialized solution, without access to the source instance.[1] This brings us to a second fundamental question: Given a source instance, which solution should we compute in order to be able to answer queries?

Concerning the first question, the *certain answers semantics*, introduced in [10], has proved to be adequate for answering a wide range of queries such as unions of conjunctive queries (a.k.a. existential positive first-order queries). Under the certain answers semantics, a query q is answered by the set of all tuples that are answers to q no matter which solution q is evaluated on. More precisely, the certain answers consist of all those tuples $\bar{a}$ such that $q(\bar{a})$ is true in all solutions. Concerning the second question, the *universal solutions* proposed in [10] have proved to be very useful. Universal solutions can be regarded as most general solutions in the sense that they contain sound and complete information. In a number of settings, they can be computed efficiently [10, 12, 16, 23, 8, 33, 15]. It was shown that the certain answers to unions of conjunctive queries can be computed by evaluating such a query on an arbitrary universal solution, followed by a simple post-processing step [10]. Similar results hold for other monotonic queries, like unions of conjunctive queries with inequalities [10, 8, 5].

For many non-monotonic queries, the certain answers semantics yields results that intuitively do not seem to be accurate [10, 3, 29].[2] The following example illustrates the basic problem:

Example 1.1. Consider a schema mapping $M = (\{R\}, \{R'\}, \Sigma)$, where R, R' are binary relation symbols and Σ contains the single st-tgd

$$\theta := \forall x, y \left(R(x, y) \to R'(x, y) \right).$$

Let S be a source instance for M where R is interpreted by $R^S := \{(a, b)\}$. Since schema mappings describe translations from source to target, it seems natural to assume that M and S together give a complete description of the solutions for S under M, namely that

---

[1] A common assumption is that the source instance is not available after the data exchange has been performed [10].

[2] It was also pointed out in [3, 29] that similar problems arise for the universal solution-based semantics from [12].

such a solution contains the tuple (a, b) in R' (as implied by θ and the tuple (a, b) in R^S), but no other tuple (since this is not implied by M and S). In particular, it seems natural to assume that the instance T which interprets R' by the relation $\{(a, b)\}$ is the only solution for S under M, and that the answer to the query

$$q(x, y) := R'(x, y) \wedge \forall z \left(R'(x, z) \rightarrow z = y \right)$$

with respect to M and S is $\{(a, b)\}$. However, the certain answers to q with respect to M and S are empty.

The assumption that a schema mapping M and a source instance S for M give a complete description of the solutions for S under M corresponds to the *closed world assumption (CWA)* [36], as opposed to the *open world assumption (OWA)* underlying the certain answers semantics. To remedy the problems mentioned above, Libkin [29] proposed semantics based on the CWA, which were later extended to a more general setting [23] (a combined version of [29] and [23] appeared in [22]). While the CWA-semantics work well in a number of situations (e.g., if a unique inclusion-minimal solution exists), they still lead to counter-intuitive answers in certain other situations [30, 2]. To this end, Libkin and Sirangelo [30] proposed a combination of the CWA and the OWA, whereas Afrati and Kolaitis [2] studied a restricted version of the CWA-semantics, and showed it to be useful for answering aggregate queries. Henceforth, we use the term *non-monotonic semantics* to refer to the semantics from [22, 30, 2]. In contrast, we call the certain answers semantics *OWA-semantics*.

A drawback of the non-monotonic semantics is that most of them are not invariant under logically equivalent schema mappings. That is, they do not necessarily lead to the same answers with respect to schema mappings specified by logically equivalent sets of constraints (see Section 3). Since logically equivalent schema mappings intuitively specify the same translation of source data to the target schema, it seems natural, though, that the answer to a query is the same on logically equivalent schema mappings. Furthermore, it can be observed that the non-monotonic semantics do not necessarily reflect the standard semantics of first-order quantifiers (see Section 3). For example, consider a schema mapping $M = (\{P\}, \{Q\}, \{\theta\})$ with $\theta = \forall x \, (P(x) \rightarrow \exists y \, Q(x, y))$, and let S be a source instance for M with a single element a in P. Under almost all of the non-monotonic semantics, the answer to the query $q = $ "Is there exactly one y with $Q(a, y)$?" is *true*. However, existential quantification $\exists y \, Q(x, y)$ is typically interpreted as: there is one y with $Q(x, y)$, or there are two y with $Q(x, y)$, or there are three y with $Q(x, y)$, and so on. To be consistent with this interpretation, the answer to q should be *false*, as otherwise the possibility of having two or more y with $Q(x, y)$ is excluded. Another reason for why it is natural to answer q by *false* is that θ can be expressed equivalently as $\theta' = \forall x (P(x) \rightarrow \bigvee_c Q(x, c))$, where c ranges over all possible values. Since M and $M' = (\{P\}, \{Q\}, \{\theta'\})$ are logically equivalent, the answer to q should either be *true* or *false* with respect to both M and M'. Letting the answer be *true* would not reflect the intended meaning of the disjunction in θ', unless we wish to interpret disjunctions exclusively.

This article introduces a new semantics for answering non-monotonic queries, called *GCWA*-semantics*, that is invariant under logically equivalent schema mappings, and intuitively reflects the standard semantics of first-order quantifiers. The starting point for the development of the GCWA*-semantics is the observation that query answering with respect to schema mappings is very similar to query answering on *deductive databases* [13] (see Section 4), and that non-monotonic query answering on deductive databases is a well-studied topic (see, e.g., [36, 35, 38, 7, 13, 9]). Many of the query answering semantics proposed in this

area can be applied with minor modifications to answer queries in relational data exchange. Therefore, it seems obvious to study these semantics in the context of data exchange. This is done in Section 4. More precisely, we consider the semantics based on Reiter's *CWA* [36], the *generalized CWA (GCWA)* [35], the *extended GCWA (EGCWA)* [38], and the *possible worlds semantics (PWS)* [7]. It turns out that the semantics based on Reiter's CWA and the EGCWA are too strong, the GCWA-based semantics is too weak, and the PWS is not invariant under logically equivalent schema mappings. On the other hand, the GCWA-based semantics seems to be a good starting point for developing the GCWA*-semantics.

In contrast to the other non-monotonic semantics, the GCWA*-semantics is defined with respect to all possible schema mappings. It is based on the new concept of *GCWA*-solutions*, in the sense that, under the GCWA*-semantics, the set of answers to a query $q(\bar{x})$ with respect to a schema mapping M and a source instance S consists of all tuples $\bar{a}$ such that $q(\bar{a})$ holds in all GCWA*-solutions for S under M. GCWA*-solutions have a very simple definition in many of the settings considered in the data exchange literature (e.g., with respect to schema mappings specified by st-tgds and egds): in these settings they are basically unions of inclusion-minimal solutions.

The major part of this article deals with the *data complexity* of evaluating queries under the GCWA*-semantics. Data complexity here means that the schema mapping and the query are fixed (i.e., they are not part of the input). We show that the GCWA*-semantics and the OWA-semantics coincide for monotonic queries (Proposition 6.1), so that all results on evaluating monotonic queries under the OWA-semantics carry over to the GCWA*-semantics. On the other hand, there are simple schema mappings (e.g., schema mappings specified by LAV tgds), and simple non-monotonic Boolean first-order queries for which query evaluation under the GCWA*-semantics is co-NP-hard or even undecidable (Propositions 6.2 and 6.3).

The main result (Theorem 6.6) shows that *universal queries* (first-order queries of the form $\forall \bar{x}\, \varphi$ with φ quantifier-free) can be evaluated in polynomial time under the GCWA*-semantics, provided the schema mapping is specified by *packed* st-tgds, which we introduce in this article. Packed st-tgds are st-tgds of the form $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, where every two distinct atomic formulas in ψ share a variable from $\bar{z}$. This is a rather strong restriction, but still allows for non-trivial use of existential quantifiers in st-tgds. Surprisingly, the undecidability result mentioned above involves a schema mapping defined by packed st-tgds, and a first-order query starting with a block of existential quantifiers and containing just one universal quantifier. The main result does not only state that universal queries can be evaluated in polynomial time under the GCWA*-semantics and schema mappings defined by packed st-tgds, but it also shows that the answers can be computed from the *core of the universal solutions* (*core solution*, for short), without access to the source instance. The core solution is the smallest universal solution and has been extensively studied in the literature (see, e.g., [12, 16, 22, 8, 33, 15]). Furthermore, since the core solution can be used to evaluate unions of conjunctive queries under the OWA-semantics, we need only one solution, the core solution, to answer both types of queries, unions of conjunctive queries and universal queries.

The article is organized as follows. In Section 2, we fix basic definitions and mention basic results that are used throughout this article. Section 3 shows that the previously proposed non-monotonic semantics are not necessarily invariant under logically equivalent schema mappings, and that they do not necessarily reflect the standard semantics of first-order quantifiers. In Section 4, we then study several of the query answering semantics for deductive databases in the context of data exchange. The new GCWA*-semantics is

introduced and illustrated in Section 5, and the data complexity of answering queries under the GCWA*-semantics is explored in Section 6.

2. Preliminaries

We use standard terminology from database theory, but slightly different notation. See, e.g., [1] for a comprehensive introduction to database theory.

2.1. **Databases.** A *schema* is a finite set σ of relation symbols, where each $R \in \sigma$ has a fixed arity $\mathrm{ar}(R) \geq 1$. An *instance* I over σ assigns to each $R \in \sigma$ a finite relation R^I of arity $\mathrm{ar}(R)$. The *active domain of I* (the set of all values that occur in I) is denoted by $\mathrm{dom}(I)$. As usual in data exchange, we assume that $\mathrm{dom}(I) \subseteq Dom$, 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 $a, b, c, \ldots$ and variants like a', a_1. Nulls serve as placeholders, or variables, for unknown constants; we will denote them by $\bot$ and variants like $\bot', \bot_1$. Let $\mathrm{const}(I) := \mathrm{dom}(I) \cap Const$ and $\mathrm{nulls}(I) := \mathrm{dom}(I) \cap Null$. An instance is called *ground* if it contains no nulls.

An *atom* is an expression of the form $R(\bar{t})$, where R is a relation symbol, and $\bar{t} \in Dom^{\mathrm{ar}(R)}$. We often identify an instance I with the set of all atoms $R(\bar{t})$ with $\bar{t} \in R^I$, that is, we often view I as the set $\{R(\bar{t}) \mid R \in \sigma, \bar{t} \in R^I\}$. An atom $R(\bar{t})$ is called *ground* if $\bar{t}$ contains no nulls.

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

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 $h \in legal(I)$ and $h(I) \subseteq J$. If $h(I) = J$, then J is called a *homomorphic image* of I. I and J are *homomorphically equivalent* if there is a homomorphism from I to J, and a homomorphism from J to I. An *isomorphism* from I to J is a homomorphism h from I to J such that h is bijective, and h^{-1} is a homomorphism from J to I. If there is an isomorphism from I to J, we call I and J *isomorphic*, and denote this by $I \cong J$. A *core* is an instance K such that there is no homomorphism from K to a proper subinstance of K. A *core* of I is a core $K \subseteq I$ such that there is a homomorphism from I to K. It is known [19] that if I and J are homomorphically equivalent, K is a core of I, and K' is a core of J, then $K \cong K'$. In particular, any two cores of I are isomorphic, so that we can speak of *the* core of I.

Given some property P of instances, a *minimal* instance with property P is an instance I with property P such that there is no instance $J \subsetneq I$ with property P.

2.2. **Queries.** As usual [1], a k-ary query over a schema σ is a mapping from instances over σ to Dom^k that is C-generic for some finite set $C \subseteq Const$ (i.e., the query is invariant under renamings of values in $Dom \setminus C$). In the context of queries defined by logical formulas, we will often use the words *formula* and *query* as synonyms. Whenever we speak of a first-order formula (FO-formula) over a schema σ, we mean a FO formula over the vocabulary that consists of all relation symbols in σ, and all constants in *Const*.

Let φ be a FO-formula over σ, and let $\mathrm{dom}(\varphi)$ be the set of all constants in φ. An assignment for φ in an instance I is a mapping 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 $I \models \varphi(\alpha)$ to indicate that φ is satisfied in I under α in the naive sense.[3] 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* [1]. For example, we have $I \models R(u_1, \ldots, u_{\mathrm{ar}(R)})(\alpha)$ precisely if $(\alpha(u_1), \ldots, \alpha(u_{\mathrm{ar}(R)})) \in R^I$; $I \models (u_1 = u_2)(\alpha)$ precisely if $\alpha(u_1) = \alpha(u_2)$; and $I \models (\exists x\, \varphi)(\alpha)$ precisely if there is an $v \in \mathrm{dom}(I) \cup \mathrm{dom}(\varphi)$ with $I \models \varphi(\alpha[v/x])$, where $\alpha[v/x]$ is the assignment defined like α, except that x is mapped to v. For an FO-formula $\varphi(x_1, \ldots, x_k)$ and a tuple $\bar{u} = (u_1, \ldots, u_k) \in (\mathrm{dom}(I) \cup \mathrm{dom}(\varphi))^k$, we often write $I \models \varphi(\bar{u})$ instead of $I \models \varphi(\alpha)$, where $\alpha(x_i) = u_i$ for each $i \in \{1, \ldots, k\}$.

A query $q(\bar{x})$ over σ is *monotonic* if $q(I) \subseteq q(J)$ for all instances I, J over σ with $I \subseteq J$. It is easy to see that all queries preserved under homomorphisms are monotonic. Here, a query $q(\bar{x})$ over σ is *preserved under homomorphisms* if and only if for all instances I, J over σ, all homomorphisms h from I to J, and all tuples $\bar{t} \in q(I)$, we have $h(\bar{t}) \in q(J)$. For example, conjunctive queries, unions of conjunctive queries, Datalog queries, and the Datalog$^{\mathbf{C}(\neq)}$ queries of [5] are preserved under homomorphisms. *Unions of conjunctive queries with inequalities* (see, e.g., [10] for a definition) are an example of monotonic queries that are not preserved under homomorphisms.

At various places in sections 3–5, we will need formulas of the infinitary logic $L_{\infty\omega}$. A $L_{\infty\omega}$ formula over a schema σ is built from atomic FO formulas over σ using negation, existential quantification, universal quantification, *infinitary disjunctions* $\bigvee \Phi$, where Φ is an arbitrary set of $L_{\infty\omega}$ formulas over σ, and *infinitary conjunctions* $\bigwedge \Phi$, where Φ is an arbitrary set of $L_{\infty\omega}$ formulas over σ. The semantics of infinitary disjunctions and infinitary conjunctions is the obvious one: for an assignment α of the variables that occur in the formulas in Φ we have $I \models \bigvee \Phi(\alpha)$ if and only if there is some $\varphi \in \Phi$ with $I \models \varphi(\alpha)$, and $I \models \bigwedge \Phi(\alpha)$ if and only if for all $\varphi \in \Phi$, $I \models \varphi(\alpha)$.

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

---

[3]Nulls that may occur in I are treated as if they were constants. In general, this may lead to counter-intuitive semantics [31, 25], since distinct nulls may represent the same constant.

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

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

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

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

A *universal solution* for S under M is a solution T for S under M such that for all solutions T' for S under M there is a homomorphism from T to T'. Note that all universal solutions for S under M are homomorphically equivalent, which implies that their cores are isomorphic. Hence, up to isomorphism there is a unique target instance, denoted by $\text{Core}(M, S)$, that is isomorphic to the cores of all universal solutions for S under M. For many schema mappings M, $\text{Core}(M, S)$ is a solution for S under M. For example, if Σ contains only st-tgds, then $\text{Core}(M, S)$ is a solution for S under M [12], which can be computed in polynomial time:

Theorem 2.1 ([12]). *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of st-tgds. Then there is a polynomial time algorithm that, given a source instance S for M, outputs $\text{Core}(M, S)$.*

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

2.4. **The Certain Answers.** Given a query $q(\bar{x})$ over τ, and a set $\mathcal{T}$ of instances over τ, we define the *certain answers to $q(\bar{x})$ on $\mathcal{T}$* by

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

The set of the certain answers to $q(\bar{x})$ on M and S under the *OWA-semantics*, as defined in [10], is then defined as

$$cert_{\text{OWA}}(q, M, S) := cert(q, \{ T \mid T \text{ is a solution for } S \text{ under } M \}).$$

If q is preserved under homomorphisms, $cert_{\text{OWA}}(q, M, S)$ can be computed from a single universal solution:

Proposition 2.2 ([10]). *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, let T be a universal solution for S under M, and let $q(\bar{x})$ be a query that is preserved under homomorphisms. Then*

$$cert_{OWA}(q, M, S) = \{\bar{a} \in q(T) \mid \bar{a} \text{ contains only constants}\}.$$

3. Review of Non-Monotonic Semantics in Relational Data Exchange

As mentioned in the introduction, for many non-monotonic queries, the OWA-semantics is counter-intuitive. In [12], Fagin, Kolaitis, and Popa propose an alternative semantics, where the set of answers to a query $q(\bar{x})$ on a schema mapping M and a source instance S is defined by $cert(q, \{T \mid T \text{ is a universal solution for } S \text{ under } M\})$. However, their semantics has similar problems as the OWA-semantics [3, 22]. For example, note that Example 1.1 goes through unchanged if the universal solution-semantics is used instead of the OWA-semantics.

Libkin [29] realized that the counter-intuitive behavior of the OWA-semantics and the universal solution-semantics can be remedied by adopting the CWA. He then introduced semantics based on the CWA. These semantics were designed for schema mappings defined by st-tgds,[4] but were later extended to schema mappings defined by st-tgds, t-tgds (see, e.g., [10] for a definition of t-tgds), and egds.

To define the CWA-semantics, we need to introduce *CWA-solutions*. For simplicity, we give their definition only for schema mappings defined by st-tgds. Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping where Σ consists of st-tgds, and let S be a source instance for M. Libkin identified the following requirements that should be satisfied by any CWA-solution T for S under M:

(1) Every atom in T is justified by M and S.
(2) Justifications are applied at most once.
(3) Every Boolean conjunctive query true in T is a logical consequence of S and Σ.

Here, a justification for an atom consists of an st-tgd $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ in Σ and assignments $\bar{a}, \bar{b}$ for $\bar{x}, \bar{y}$ such that $S \models \varphi(\bar{a}, \bar{b})$. Such a justification can be applied with an assignment $\bar{u}$ for $\bar{z}$, and would then justify the atoms in $\psi(\bar{a}, \bar{u})$. We require that all atoms that are justified belong to T. Once all requirements are properly formalized, it turns out that a *CWA-solution* for S under M is a universal solution for S under M that is a homomorphic image of $\mathrm{CanSol}(M, S)$ [22].

We now recall the simplest of the four CWA-semantics introduced in [22], which we henceforth call *CWA-semantics*. Recall that solutions in data exchange may contain nulls. Nulls represent unknown constants, in particular, two distinct nulls may represent the same constant. Treating nulls as constants may thus lead to counter-intuitive answers [25]. A standard way to answer queries while taking into account the semantics of nulls is to return the certain answers. To this end, one views an instance T over τ as a set $poss(T)$ of ground instances obtained by substituting concrete constants for each null. Formally, we let

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

---

[4]Strictly speaking, Libkin considered sentences of the form $\forall \bar{x}\, (\varphi(\bar{x}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{y}))$, where φ is a first-order formula over the source schema, and ψ is a conjunction of relational atoms over the target schema.

where a *valuation* of T is a mapping $v \colon \operatorname{dom}(T) \to Const$ with $v \in legal(T)$. Then the certain answers to a query $q(\bar{x})$ on T are

$$cert(q, T) := cert(q, poss(T)).$$

Now, Libkin's idea was to define the *CWA-answers to a query $q(\bar{x})$ on M and S* by

$$cert_{\mathrm{CWA}}(q, M, S) := \bigcap \{cert(q, T) \mid T \text{ is a CWA-solution for } S \text{ under } M\}.$$

That is, $cert_{\mathrm{CWA}}(q, M, S)$ is the set of the certain answers to $q(\bar{x})$ on the CWA-solutions for S under M, but instead of answering $q(\bar{x})$ on an individual CWA-solution T by $q(T)$, the certain answers are used. As shown in [22], $cert_{\mathrm{CWA}}(q, M, S)$ can be computed from the canonical solution: $cert_{\mathrm{CWA}}(q, M, S) = cert(q, \mathrm{CanSol}(M, S))$.

While the CWA-semantics works well in a number of situations (e.g., for schema mappings defined by full st-tgds, full t-tgds and egds), and in particular resolves the problems observed in [3, 22] and Example 1.1, one of the drawbacks of the CWA-semantics is that it is not invariant under logically equivalent schema mappings. That is, there are schema mappings $M_1 = (\sigma, \tau, \Sigma_1)$ and $M_2 = (\sigma, \tau, \Sigma_2)$, where Σ_1 is logically equivalent to Σ_2, and a query $q(\bar{x})$ such that the answers to $q(\bar{x})$ with respect to M_1 differ from the answers to $q(\bar{x})$ with respect to M_2:

Example 3.1. Let $M_1 = (\sigma, \tau, \Sigma_1)$ and $M_2 = (\sigma, \tau, \Sigma_2)$ be schema mappings, where σ contains a unary relation symbol P, τ contains a binary relation symbol E, and

$$\Sigma_1 := \big\{ \forall x \, \big(P(x) \to E(x, x) \big) \big\},$$
$$\Sigma_2 := \Sigma_1 \cup \big\{ \forall x \, \big(P(x) \to \exists z \, E(x, z) \big) \big\}.$$

Then M_1 and M_2 are logically equivalent.

Let S be an instance over σ with $P^S = \{a\}$. Furthermore, let T_1 and T_2 be instances over τ with $E^{T_1} = \{(a, a)\}$ and $E^{T_2} = \{(a, a), (a, \bot)\}$. Note that $T_i = \mathrm{CanSol}(M_i, S)$ for each $i \in \{1, 2\}$. Even more, T_1 is the unique CWA-solution for S under M_1, whereas T_2 is a CWA-solution for S under M_2. Thus, for the query

$$q(x) := \exists z \, \big(E(x, z) \land \forall z'(E(x, z') \to z' = z) \big),$$

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

Remark 3.2. If we replace q by the query q' which asks for all x such that there are at least two z with $E(x, z)$, then q' is answered differently on M_1 and M_2 with respect to the maybe answers-semantics from [22]. The other two semantics in [22] are invariant under logically equivalent schema mappings, though.

Intuitively, logically equivalent schema mappings specify the same translation of source data to the target schema, so it seems natural that the answer to a query is the same on logically equivalent schema mappings. Furthermore, invariance under logically equivalent schema mappings seems to be a good way to achieve a "syntax-independent" semantics, that is, a semantics that does not depend on the concrete presentation of the sentences of the schema mapping like the CWA-semantics. Gottlob, Pichler, and Savenkov [17] suggest a different approach for achieving syntax-independence that is based on first normalizing a schema mapping, and then "applying" the semantics. Let me mention that weaker notions of equivalence between schema mappings have been considered in the literature [11]. Instead of requiring invariance under logical equivalence, one could use any of these weaker notions.

There is another drawback of the CWA-semantics, though: it does not necessarily reflect the standard semantics of FO quantifiers. Typically, existential quantification $\exists x \, \varphi$ is interpreted as: there is one x satisfying φ, or there are two x satisfying φ, or there are three x satisfying φ, and so on. But this is not necessarily reflected by the CWA-semantics:

Example 3.3. Let $M = (\sigma, \tau, \Sigma)$ be the schema mapping, where σ contains a unary relation symbol P, τ contains a binary relation symbol E, and Σ consists of the st-tgd

$$\theta \; := \; \forall x \, \big(P(x) \to \exists z \, E(x, z) \big).$$

Let S be the source instance for M with $P^S = \{a\}$, and let $T := \mathrm{CanSol}(M, S)$. Then up to renaming of nulls, T is the unique CWA-solution for S under M, and $E^T = \{(a, \bot)\}$. Therefore, for the query

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

we have $cert_{\mathrm{CWA}}(q, M, S) = \{a\}$. In other words, $cert_{\mathrm{CWA}}(q, M, S)$ excludes the possibility that there is more than one value z with $E(a, z)$. However, this seems to be too strict since if we interpret $\exists z \, E(x, z)$ in the standard first-order way, S and θ tell us that there is one z satisfying $E(a, z)$, or there are two z satisfying $E(a, z)$, or there are three z satisfying $E(a, z)$, and so on. In particular, they explicitly state that it is possible that there is more than one z satisfying $E(a, z)$.

Remark 3.4. The example also shows that the potential certain answers semantics from [22] does not necessarily reflect the standard semantics of FO quantifiers. For the remaining two semantics in [22], we can replace q by the query q' asking for all x for which there are at least two z with $E(x, z)$. Then the set of answers to q' under those semantics will be empty. In other words, they tell us that it is not possible that there are more than two z that satisfy $E(x, z)$, which is not desired.

In [30], Libkin and Sirangelo proposed a generalization of the CWA-semantics based on a combination of the CWA and the OWA, using which we can resolve the problem described in Example 3.3. The idea is to annotate each position (occurrence of a variable) in the head of an st-tgd as *closed* or *open*, where open positions correspond to those positions where more than one value may be created. For example, recall the schema mapping M and the source instance S from Example 3.3, and let α be the following annotation of θ:

$$\theta_\alpha \; := \; \forall x \, \big(P(x) \to \exists z \, E(x^{closed}, z^{open}) \big).$$

Then the valid solutions for S under M and α are all solutions for S under M of the form $\{E(a, b) \mid b \in X\}$, where X is a finite set of constants. That is, the first position – which corresponds to the closed position in the head of θ_α – is restricted to the value a assigned to x when applying θ to S, while the second position – which corresponds to the open position in the head of θ_α – is unrestricted in the sense that an arbitrary finite number of values may be created at this position. Let us write $\mathrm{sol}_\alpha(M, S)$ for the set of all these solutions. The set of answers to a query $q(\bar{x})$ on M, α and S is then

$$cert_\alpha(q, M, S) \; := \; cert(q, \mathrm{sol}_\alpha(M, S)).$$

In particular, it is easy to see that for the query q in Example 3.3 we have $cert_\alpha(q, M, S) = \emptyset$, as desired. However, the "mixed world" semantics may still be counter-intuitive:

Example 3.5. Let $M = (\sigma, \tau, \Sigma)$ be defined by $\sigma = \{R\}$, $\tau = \{E, F\}$, and $\Sigma = \{\theta\}$, where

$$\theta := \forall x, y \left(R(x, y) \to \exists z \left(E(x, z) \wedge F(z, y) \right) \right). \tag{3.1}$$

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. The possibility that there are precisely two such z, or precisely three such z et cetera is perfectly consistent with θ, and should not be denied when answering queries. So, it seems that the only solutions for $S = \{R(a, b)\}$ under M should be those solutions T for which there is a finite set $X \subseteq Const$ such that $T = \{E(a, x) \mid x \in X\} \cup \{F(x, b) \mid x \in X\}$. In particular, we should expect that the answer to

$$q(x, y) := \exists^{=1} z \left(E(x, z) \wedge F(z, y) \right)$$

on M and S is empty, and that the answer to

$$q'(x) := \forall z \left(E(x, z) \to \exists y \, F(z, y) \right)$$

on M and S is $\{a\}$.

Now consider an annotation α for θ where the occurrence of z in $E(x, z)$ is *closed*. According to the definition in [30], for all valid solutions T for S under M and α, and all tuples $(c, d), (c', d') \in E^T$ we have $d = d'$, so that $cert_\alpha(q, M, S) = \{(a, b)\}$. Hence, $cert_\alpha(q, M, S)$ excludes 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. The same is true if the occurrence of z in $F(z, y)$ is *closed*.

Let us finally consider an annotation α' where both occurrences of z in the head of θ are *open*. According to the definition in [30], we have $cert_{\alpha'}(q', M, S) = \emptyset$, since

$$T^* := \{E(a, c), E(a, c'), F(c, b)\}$$

would be a valid solution for S under M and α' according to this definition. Intuitively, the "mixed world" semantics is "too open" in that it allows $E(a, c')$ to occur in T^* without enforcing that the corresponding atom $F(c', b)$ is present in T^*.

Remark 3.6. The existential quantifier in (3.1) 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), as in

$$\theta' := \forall x, y \left(R(x, y) \to \bigvee_{c \in Const} \left(E(x, c) \wedge F(c, y) \right) \right). \tag{3.2}$$

The interpretation of the existential quantifier discussed in Example 3.5 then corresponds to an inclusive interpretation of the disjunction in (3.2). Intuitively, the desired set of solutions for $S = \{R(a, b)\}$ under M – the set of solutions of the form $T = \{E(a, x) \mid x \in X\} \cup \{F(x, b) \mid x \in X\}$ for some finite set $X \subseteq Const$ – is the smallest set of solutions that reflects an inclusive interpretation of the disjunction in (3.2).

Remark 3.7. Afrati and Kolaitis [2] showed that a restriction of Libkin's CWA is useful for answering *aggregate queries*. Their semantics is defined with respect to schema mappings specified by st-tgds. In principle, we could use it to answer non-aggregate queries as follows. 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 $q(\bar{x})$ be a query over τ. Instead of answering $q(\bar{x})$ by the certain answers to $q(\bar{x})$ on $poss(\text{Core}(M, S))$, we answer $q(\bar{x})$ by the certain answers to $q(\bar{x})$

on the set of all *endomorphic images* of $\mathrm{CanSol}(M, S)$. Here, an endomorphic image of $\mathrm{CanSol}(M, S)$ is an instance T such that $T = h(\mathrm{CanSol}(M, S))$ for some homomorphism h from $\mathrm{CanSol}(M, S)$ to $\mathrm{CanSol}(M, S)$. However, the endomorphic images-semantics seems to be too strong – it is stronger than the CWA-semantics. In particular, Example 3.1 shows that the endomorphic images-semantics is not invariant under logically equivalent schema mappings, and Example 3.3 shows that it does not necessarily reflect the standard semantics of FO quantifiers.

Our goal is to develop a semantics for answering non-monotonic queries that is invariant under logically equivalent schema mappings, and is just "open enough" to interpret existential quantifiers (when viewed as an infinite disjunction, as suggested in Remark 3.6) inclusively. We start by studying semantics for answering non-monotonic queries on deductive databases.

Remark 3.8. Note the following side-effect of invariance under logical equivalent schema mappings. Consider the schema mappings M_1, M_2 and source instance S from Example 3.1. Then under a reasonable closed world semantics, $T = \{E(a, a)\}$ should be the only "valid solution" for S under M_1. Invariance under logical equivalence thus enforces that T is also the only "valid solution" for S under M_2. This seems to be counter-intuitive, but only as long as one considers the two st-tgds in M_2 isolated.

4. Deductive Databases and Relational Data Exchange

Query answering on deductive databases and query answering in relational data exchange are very similar problems. Both require answering a query on a ground database that is equipped with a set of constraints. On the other hand, answering non-monotonic queries on deductive databases is a well-studied topic. In this section, we translate some of the semantics that were proposed to answer non-monotonic queries on deductive databases into the context of relational data exchange.

A *deductive database* [13] over a schema σ is a set of FO sentences, called *clauses*, of the form

$$\forall \bar{x} \big(\neg R_1(\bar{y}_1) \vee \cdots \vee \neg R_m(\bar{y}_m) \vee R'_1(\bar{z}_1) \vee \cdots \vee R'_n(\bar{z}_n) \big), \tag{4.1}$$

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

Several semantics for answering non-monotonic queries on deductive databases were proposed (see, e.g., [36, 35, 38, 7], and the survey articles [13, 9]). Often, these semantics can be applied with some minor modifications to more general sets of logical sentences such as

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

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

In the following, we concentrate on Reiter's *CWA* [36], since this is the basic assumption underlying all previous approaches for non-monotonic query answering in relational data exchange. However, we also consider variants of Reiter's CWA such as semantics based on Minker's *generalized CWA (GCWA)* [35], Yahya's and Henschen's *extended GCWA (EGCWA)* [38] as well as Chan's *possible worlds semantics (PWS)* [7].

4.1. **Reiter's Closed World Assumption (RCWA).** Reiter's *closed world assumption (RCWA)*,[5] formalized by Reiter in [36], assumes that every ground atom that is not implied by a database is false. This is a common assumption for relational databases.

Reiter formalized the RCWA as follows. For a deductive database D and a formula φ, we write $D \models \varphi$ if and only if for all instances I with $I \models D$, we have $I \models \varphi$. Given a deductive database D over a schema σ, let

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

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

Translated into the relational data exchange framework, we obtain:

Definition 4.1 (RCWA-solution, RCWA-answers)**.** Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let $q(\bar{x})$ be a query over τ.
(1) An *RCWA-solution* for S under M is a ground target instance T for M such that $S \cup T$ is a RCWA-model of $D_{M,S}$ (recall the definition of $D_{M,S}$ from (4.1)),
(2) We call $cert_{\mathrm{RCWA}}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of all RCWA-solutions for S under M, the *RCWA-answers to $q(\bar{x})$ on M and S.*

Note that RCWA-solutions are *ground* (i.e., contain no nulls), in contrast to other notions of solutions such as plain solutions, universal solutions, or CWA-solutions presented in previous sections.

The RCWA is a very strong assumption. For example, if an RCWA-solution for S under M exists, it is the unique minimal (ground) solution for S under M:

Proposition 4.2. *Let M be a schema mapping, and let S be a source instance for M. Then a solution T for S under M is an RCWA-solution for S under M if and only if T is contained in every ground solution for S under M.*

Proof. Suppose T is an RCWA-solution for S under M, and let T' be a ground solution for S under M. If there is an atom $R(\bar{t}) \in T \setminus T'$, then $D_{M,S} \not\models R(\bar{t})$, so that $\neg R(\bar{t}) \in \overline{D_{M,S}}$, and hence T is no RCWA-solution for S under M. Therefore, $T \subseteq T'$. To prove the other direction, suppose that T is contained in every ground solution for S under M. Then, for all atoms $R(\bar{t}) \in T$ we have $D_{M,S} \models R(\bar{t})$, so that T is an RCWA-solution for S under M. $\square$

_____
[5]We write RCWA instead of CWA to avoid confusion with Libkin's formalization of the CWA.

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

Proposition 4.3. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of full st-tgds, let S be a source instance for M, and let $q(\bar{x})$ be a query over τ. Then, $cert_{RCWA}(q, M, S) = cert_{CWA}(q, M, S)$.*

Proof. Since Σ consists of full st-tgds, there is a unique minimal ground solution T_0 for S under M, which is also the unique CWA-solution for S under M, and, by Proposition 4.2, the unique RCWA-solution for S under M. Consequently, $cert_{\text{RCWA}}(q, M, S) = cert_{\text{CWA}}(q, M, S) = q(T_0)$. $\square$

However, for schema mappings that contain non-full st-tgds, $cert_{\text{RCWA}}$ may lead to answers that are inconsistent with M and S. This is illustrated by the following example, which is based on Example 8 in [36].

Example 4.4. Let $M = (\{P\}, \{E\}, \Sigma)$, where $\Sigma := \{\forall x (P(x) \rightarrow \exists z\, E(x, z))\}$, and let S be the source instance for M with $P^S = \{a\}$. Since there is no unique minimal ground solution, Proposition 4.2 implies that there is no RCWA-solution for S under M. Consequently, the RCWA-answers to $q(x) := \exists z\, E(x, z)$ on M and S are empty. In other words, $cert_{\text{RCWA}}(q, M, S)$ tells us that there is no value z satisfying $E(a, z)$. This is clearly inconsistent with M and S, which tell us that there is a value z satisfying $E(a, z)$, and hence that the set of answers should be $\{a\}$.

4.2. **The Generalized Closed World Assumption (GCWA).** Minker [35] extended Reiter's CWA to the *generalized closed world assumption (GCWA)* as follows. Recall the definition of *minimal instance possessing some property* from Section 2. Let D be a deductive database over a schema σ. A *minimal model* of D is a minimal instance with the property of being a model of D. Let

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

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

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

Translated into the relational data exchange framework, we obtain:

Definition 4.5 (GCWA-solution, GCWA-answers)**.** Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let $q(\bar{x})$ be a query over τ.
(1) A *GCWA-solution* for S under M is a ground target instance T for M such that $S \cup T$ is a GCWA-model of $D_{M,S}$.
(2) We call $cert_{\text{GCWA}}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of the GCWA-solutions for S under M, the *GCWA-answers to $q(\bar{x})$ on M and S*.

We have the following characterization of GCWA-solutions:

Proposition 4.6. *Let M be a schema mapping, and let S be a source instance for M. Then a solution T for S under M is an GCWA-solution for S under M if and only if there is a set $\mathcal{T}$ of minimal ground solutions for S under M such that $T \subseteq \bigcup \mathcal{T}$.*

Proof. Suppose T is a GCWA-solution for S under M. Then for each atom A in T, there is a minimal ground solution T_A for S under M with $A \in T_A$ (otherwise, $\neg A \in \overline{\overline{D_{M,S}}}$, so that T would not be a GCWA-solution for S under M). But then we have $T \subseteq \bigcup_{A \in T} T_A$.

Suppose now that $\mathcal{T}$ is a set of minimal ground solutions for S under M such that $T \subseteq \bigcup \mathcal{T}$. Then, for all atoms $A \in T$ we have $\neg A \notin \overline{\overline{D_{M,S}}}$, and it follows that $\mathcal{T}$ is a GCWA-solution for S under M. $\qquad\square$

Similar to the RCWA-answers semantics, it can be shown that $cert_{\text{GCWA}}$ coincides with $cert_{\text{CWA}}$ on schema mappings defined by full st-tgds. Moreover, $cert_{\text{GCWA}}$ leads to the desired answers to the query in Example 4.4:

Example 4.7. Recall the schema mapping M, the source instance S, and the query q from Example 4.4. We now have

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

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

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

Example 4.8. Consider a slight extension of the schema mapping from Example 4.4, namely $M = (\{P\}, \{E, F\}, \Sigma)$, where Σ consists of the st-tgd

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

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

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

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

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

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

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

then $F(b, c)$ should be present for some $c \in Const$. Similarly, whenever $F(b, c)$ is present for some $b, c \in Const$, then $E(a, b)$ should be present.

4.3. **Extensions of the GCWA.** Various extensions of the GCWA have been proposed. One of these extensions is the *extended GCWA (EGCWA)* by Yahya and Henschen [38], which restricts the set of models of a deductive database D to the minimal models of D. So, given a schema mapping $M = (\sigma, \tau, \Sigma)$ and a source instance S for M, an *EGCWA-solution* for S under M can be defined as a ground minimal solution for S under M, and given a query $q(\bar{x})$ we can define

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

where $\mathcal{I}$ is the set of all EGCWA-solutions for S under M. Then, for the schema mapping M, the source instance S for M, and the query q in Example 4.7, $cert_{\text{EGCWA}}(q, M, S) = cert_{\text{GCWA}}(q, M, S)$, and for the schema mapping M, the source instance S for M, and the query q in Example 4.8, $cert_{\text{EGCWA}}(q, M, S) \neq \emptyset$, as desired. However, the EGCWA seems to be too strong in the sense that it removes too many solutions from the set of all solutions. More precisely, it interprets existential quantifiers (when viewed as disjunctions) exclusively rather than inclusively. We illustrate this by the following example.[6]

Example 4.9. Let $M = (\{P\}, \{E\}, \Sigma)$ be a schema mapping, where Σ consists of

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

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

$$q(x) := \exists z_1 \exists z_2 \left(E(x, z_1) \land E(x, z_2) \land \forall z_3 \left(E(x, z_3) \to (z_3 = z_1 \lor z_3 = z_2) \right) \right),$$

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

To conclude this section, let us consider the *possible worlds semantics (PWS)* by Chan [7]. A natural translation of the PWS for the case of schema mappings defined by st-tgds is as follows: Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds, and let S be a source instance for M. The definition of a PWS-solution for S under M can be given in terms of *justifications*, as in [22]. Given a target instance T for M and an atom $R(\bar{t}) \in T$, we say that $R(\bar{t})$ is *justified* in T under M and S if and only if there is a st-tgd $\forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$ in Σ, tuples $\bar{a}, \bar{b}$ over $\text{dom}(S)$ with $S \models \varphi(\bar{a}, \bar{b})$, and a tuple $\bar{u}$ over $\text{dom}(T)$ such that $T \models \psi(\bar{a}, \bar{u})$, and $R(\bar{t})$ is one of the atoms in $\psi(\bar{a}, \bar{u})$. A PWS-solution for S under M is then a ground solution T for S under M such that all atoms in T are justified in T under M and S. For a query q over τ, we let

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

---

[6]Example 3.3 illustrates this as well, but Example 4.9 seems to make it more clear why it may be desirable to interpret existential quantifiers inclusively.

where $\mathcal{I}$ is the set of all PWS-solutions for S under M. However, $cert_{\mathrm{PWS}}$ does not respect logical equivalence of schema mappings which can be easily verified using the schema mapping, the source instance and the query from Example 3.1.

5. The GCWA*-Semantics

We now introduce the GCWA*-semantics, and argue that it has the desired properties – being invariant under logically equivalent schema mappings, and being just "open enough" to interpret existential quantifiers inclusively.

Among the semantics considered in the previous sections, the GCWA-semantics is closest to the desired semantics. For instance, consider the schema mapping $M = (\{P\}, \{E\}, \Sigma)$, where Σ consists of $\forall x\,(P(x) \to \exists z\,E(x,z))$, and the source instance S for M with $P^S = \{a\}$ from Example 4.4. Let $\mathcal{T}$ be the set of all GCWA-solutions for S under M. As shown in Example 4.7, $\mathcal{T}$ consists of all target instances T for M such that there is a nonempty finite set $B \subseteq Const$ with $T = T_B$, where $E^{T_B} = \{(a,b) \mid b \in B\}$. The set $\mathcal{T}$ is precisely as we would like the set of solutions to be. Intuitively, it precisely captures what is expressed by M and S: there is one $b \in Const$ satisfying $E(a,b)$, or there are two distinct $b_1, b_2 \in Const$ satisfying $E(a,b_1)$ and $E(a,b_2)$, or there are three distinct $b_1, b_2, b_3 \in Const$ satisfying $E(a,b_1)$, $E(a,b_2)$ and $E(a,b_3)$, and so on. The case that there are n distinct $b_1, \ldots, b_n \in Const$ such that $E(a,b_i)$ holds for each $i \in \{1, \ldots, n\}$ is captured precisely by T_B, where $B := \{b_1, \ldots, b_n\}$. However, as we have argued in Example 4.8, with respect to other schema mappings the GCWA is still "too open".

Note that the set $\mathcal{T}$ in the above example is the set of all ground solutions for S under M that are unions of minimal solutions. Indeed, it seems to be a good idea to use the set of all such solutions as the set of "valid solutions". As we have done in Remark 3.6, we can express the existential quantifier in the st-tgd from Example 4.4 equivalently by an infinite disjunction, resulting in the following $L_{\infty\omega}$-sentence:

$$\theta' := \forall x\left(P(x) \to \bigvee_{c \in Const} E(x,c)\right).$$

Then the ground minimal solutions for a source instance S under $M' = (\{P\}, \{E\}, \{\theta'\})$ correspond to the disjuncts of the disjunction in θ', and an inclusive interpretation of this disjunction is guaranteed by taking all ground solutions that are unions of minimal solutions as "valid solutions".

This can be generalized to other schema mappings defined by st-tgds. For instance, recall the schema mapping M, and the source instance S for M from Example 4.8, where the GCWA is "too open". Again, we can express the st-tgd θ that defines M by an $L_{\infty\omega}$-sentence where the existential quantifier in θ is replaced by an infinite disjunction:

$$\theta' := \forall x\left(P(x) \to \bigvee_{c_1,c_2 \in Const}\big(E(x,c_1) \wedge F(c_1,c_2)\big)\right).$$

Then the ground minimal solutions for S under M correspond to the disjuncts of the disjunction in θ', and an inclusive interpretation of this disjunction is guaranteed by taking the set $\mathcal{T}$ of all ground solutions for S under M that are unions of minimal solutions as "valid solutions". That is, we take the set $\mathcal{T}$ of all ground target instances T for M such that

$E^T = \{(a, b) \mid (b, c) \in F^T \text{ for some } c \in Const\}$ and $F^T \neq \emptyset$. Indeed, the set of the certain answers to the query q from Example 4.8 on $\mathcal{T}$ is nonempty, as desired.

The preceding two examples suggest to answer queries by the certain answers on the set of all ground solutions that are unions of minimal solutions. Let us call such solutions *GCWA*-solutions* for the moment:

Definition 5.1 (working definition)**.** Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let $q(\bar{x})$ be a query over τ.

(1) A *GCWA*-solution* for S under M is a ground solution for S under M that is a union of minimal solutions for S under M.

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

The definition of GCWA*-solutions and GCWA*-answers already seems to be a good approximation to the concept of solutions, and query answering semantics, respectively, that we would like to have. Immediately from the definitions, we obtain that the GCWA*-answers are invariant under logically equivalent schema mappings:

Proposition 5.2. *If $M_1 = (\sigma, \tau, \Sigma_1)$ and $M_2 = (\sigma, \tau, \Sigma_2)$ are logically equivalent schema mappings, S is a source instance for M_1 and M_2, respectively, and $q(\bar{x})$ is a query over τ, then $cert_{GCWA^*}(q, M_1, S) = cert_{GCWA^*}(q, M_2, S)$.*

Furthermore, let us generalize the discussion from the beginning of this section to argue that GCWA*-solutions and the GCWA*-answers as defined above are suitable for schema mappings defined by st-tgds. Let $M = (\sigma, \tau, \Sigma)$ be such a schema mapping. Given a source instance S for M, let

$$\Psi_{M,S} := \big\{ \exists \bar{z}\, \psi(\bar{u}, \bar{z}) \mid \text{there are a tgd } \forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})) \text{ in } \Sigma \text{ and tuples}$$

$$\bar{u} \in Const^{|\bar{x}|}, \bar{v} \in Const^{|\bar{y}|} \text{ such that } S \models \varphi(\bar{u}, \bar{v})\big\}.$$

For each ground target instance T for M, it holds that T is a solution for S under M if and only if T satisfies all sentences in $\Psi_{M,S}$. Let $\mathcal{T}_0$ be the set of all ground minimal solutions for S under M. Since all sentences in $\Psi_{M,S}$ are monotonic, $\Psi_{M,S}$ is logically equivalent (on the set of all ground instances over τ) to the sentence

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

that is, for all ground instances T over τ, we have $T \models \psi_{M,S}$ if and only if T satisfies all sentences in $\Psi_{M,S}$. Now, $\psi_{M,S}$ tells us that there is one $T_0 \in \mathcal{T}_0$ such that all $R(\bar{t}) \in T_0$ are satisfied, or there are two $T_0 \in \mathcal{T}_0$ such that all $R(\bar{t}) \in T_0$ are satisfied, and so on. So, intuitively, the set of all solutions that are unions of solutions from $\mathcal{T}_0$ (namely, the set of all GCWA*-solutions for S under M) captures what is expressed by M and S in the sense as explained in the two motivating examples from the beginning of this section.

Remark 5.3. The above argumentation can be generalized to more general classes of schema mappings. For example, let us consider schema mappings defined by a certain kind of $L_{\infty\omega}$ sentences. Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of *right-monotonic $L_{\infty\omega}$-st-tgds*, which are $L_{\infty\omega}$ sentences of the form

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

where φ is a $L_{\infty\omega}$ formula over σ, and ψ is a *monotonic* $L_{\infty\omega}$ formula over τ. We assume that for each instance S over σ, and for each instance T over τ, we have $S \cup T \models \theta$ if and only if for all $\bar{a} \in (\mathrm{dom}(S) \cup \mathrm{dom}(\theta))^{|\bar{x}|}$, where $\mathrm{dom}(\theta)$ is the set of all constants that occur in θ, $S \models \varphi(\bar{a})$ implies $T \models \psi(\bar{a})$. This can be enforced, for example, by relativizing the universal quantifiers, and the quantifiers in φ to the active domain over σ, and by relativizing the quantifiers in ψ to the active domain over τ. Note that right-monotonic $L_{\infty\omega}$-st-tgds capture st-tgds.

Now, the above argumentation for schema mappings defined by st-tgds goes through for M. The only difference is that, given a source instance S for M, we let

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

The remaining part goes through unchanged.

The following example shows that the GCWA*-answers can be appropriate beyond schema mappings defined by right-monotonic $L_{\infty\omega}$-st-tgds.

Example 5.4. Recall the schema mapping M, the source instance S for M, and the query q from Example 4.9. For each ground target instance T for M that is the union of minimal solutions for S under M, there exists a nonempty finite set $C \subseteq Const$ with $E^T = \{(a,b) \mid b \in C\}$. T is a GCWA*-solution for S under M if and only if $2 \le |C| \le 3$, as desired. Note that the GCWA*-answers to q on M and S are empty, as intuitively expected.

However, let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ does *not* entirely consist of right-monotonic $L_{\infty\omega}$-st-tgds, and let S be a source instance for M. Then the set of the GCWA*-solutions for S under M may suppress information that should intuitively be taken into account when answering queries:

Example 5.5. Consider the schema mapping $M = (\{P\}, \{E, F\}, \{\theta_1, \theta_2\})$, where

$$\theta_1 := \forall x \left(P(x) \to \exists z \exists z'\, E(z, z')\right),$$
$$\theta_2 := \forall x \forall y \left(E(x,y) \wedge E(x',y) \to F(x,x')\right),$$

and let S be the source instance for M with $P^S = \{a\}$. Furthermore, let c_1, c_2, c be constants with $c_1 \ne c_2$. Then the target instance

$$T := \{E(c_1, c), E(c_2, c)\} \cup \{F(c_i, c_j) \mid 1 \le i, j \le 2\}$$

for M is a solution for S under M. However, it is not a GCWA*-solution for S under M, since every ground minimal solution for S under M has the form $\{E(d, d'), F(d, d)\}$ for $d, d' \in Const$.

Nevertheless, it seems natural to take into account T when answering queries. Intuitively, under an inclusive interpretation of the existential quantifiers in θ_1, the st-tgd θ_1 and the atom $P(a)$ in S tell us that it is possible that both $E(c_1, c)$ and $E(c_2, c)$ hold. In combination with θ_2, this tells us that it is possible that a solution contains $E(c_1, c)$, $E(c_2, c)$ and $F(c_i, c_j)$ for $i, j \in \{1, 2\}$. Therefore, T should be a possible solution.

We now extend the set of GCWA*-solutions so that solutions like T are included as well. We do this using the following closure operation. Let $\mathcal{T}$ be a nonempty finite set of ground minimal solutions for S under M, say $\mathcal{T}$ consists of instances $T_i = \{E(d_i, e_i), F(d_i, d_i)\}$ for $i = 1, \ldots, n$. $\mathcal{T}$ represents the information that $\bigwedge_{i=1}^{n} E(d_i, e_i)$ holds, and S and θ_1 intuitively tell us that this is possible. In general, $T_0 := \bigcup \mathcal{T}$ is not a solution for S under M, since in general it does not satisfy θ_2 (it does if the constants $e_1, \ldots, e_n$ are distinct). However,

we can extend T_0 to a solution T_0' by adding atoms to T_0 so that θ_2 is satisfied. We pick the minimal set of such atoms, namely $\{F(d_i, d_j) \mid 1 \le i, j \le n, e_i = e_j\}$, since we do not want to add any atoms that are not needed to satisfy θ_2. Note that T_0' is minimal among all ground solutions T' for S under M with $T' \supseteq T_0$. We add T_0' to the set of "valid solutions".

The set of solutions that results from applying the closure operator contains all solutions T for S under M of the form

$$T_0' = \{E(d_i, e_i) \mid 1 \le i \le n\} \cup \{F(d_i, d_j) \mid 1 \le i, j \le n, e_i = e_j\},$$

where $n \ge 1$ and $d_1, \ldots, d_n, e_1, \ldots, e_n$ are arbitrary constants. Intuitively, this set precisely captures what is expressed by M and S.

In general, we iterate the (appropriately generalized) closure operator until a fixed point is reached. We start with the set

$$\mathcal{T}_{M,S}^0 := \{T \mid T \text{ is a ground minimal solution for } S \text{ under } M\}.$$

For each set $\mathcal{I}$ of instances, let

$$\langle \mathcal{I} \rangle := \left\{ \bigcup \mathcal{I}' \mid \mathcal{I}' \text{ is a nonempty finite subset of } \mathcal{I} \right\},$$

where $\bigcup \mathcal{I}'$ denotes the union of all instances in $\mathcal{I}'$. For every $i \ge 0$, let

$$\mathcal{T}_{M,S}^{i+1} := \mathcal{T}_{M,S}^i \cup \big\{ T_0' \mid T_0' \notin \langle \mathcal{T}_{M,S}^i \rangle, \text{ and there is a } T_0 \in \langle \mathcal{T}_{M,S}^i \rangle \text{ such that } T_0' \text{ is minimal}$$
$$\text{among all ground solutions } T' \text{ for } S \text{ under } M \text{ with } T_0 \subseteq T' \big\}.$$

Intuitively, each instance $T_0' \in \mathcal{T}_{M,S}^{i+1} \setminus \mathcal{T}_{M,S}^i$ is a "minimal consequence" of some "fact" $T_0 \in \langle \mathcal{T}_{M,S}^i \rangle$ mentioned by M and S. In Example 5.5, the instance T belongs to $\mathcal{T}_{M,S}^1 \setminus \mathcal{T}_{M,S}^0$.

Note that if Σ contains only st-tgds, or more generally, right-monotonic $L_{\infty\omega}$-st-tgds, we have $\mathcal{T}_{M,S}^0 = \mathcal{T}_{M,S}^i$ for all $i \ge 0$, and $\langle \mathcal{T}_{M,S}^0 \rangle$ is precisely the set of all GCWA*-solutions for S under M. So, for schema mappings defined by st-tgds or right-monotonic $L_{\infty\omega}$-st-tgds, we have to take into account only the GCWA*-solutions as defined earlier. For more general schema mappings, we take into account all solutions for S under M that are unions of one or more instances in

$$\mathcal{T}_{M,S}^* := \bigcup_{i \ge 0} \mathcal{T}_{M,S}^i.$$

Definition 5.6 (GCWA*-solution, GCWA*-answers)**.** Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let q be a query over τ.
(1) A *GCWA*-solution* for S under M is a ground solution T for S under M that is the union of one or more instances in $\mathcal{T}_{M,S}^*$.
(2) We call $cert_{\text{GCWA}^*}(q, M, S) := cert(q, \mathcal{I})$, where $\mathcal{I}$ is the set of the GCWA*-solutions for S under M, the *GCWA*-answers to q on M and S*.

As before, immediately from the definitions, we obtain that the GCWA*-answers are invariant under logically equivalent schema mappings:

Proposition 5.7. *If $M_1 = (\sigma, \tau, \Sigma_1)$ and $M_2 = (\sigma, \tau, \Sigma_2)$ are logically equivalent schema mappings, S is a source instance for M_1 and M_2, respectively, and q is a query over τ, then $cert_{GCWA^*}(q, M_1, S) = cert_{GCWA^*}(q, M_2, S)$.*

Furthermore, it is easy to prove:

Proposition 5.8. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of st-tgds and egds, and let S be a source instance for M. Then a target instance T for M is a GCWA*-solution for S under M if and only if T is the union of one or more ground minimal solutions for S under M, and T satisfies all egds in Σ.*

To conclude this section, we show that, with respect to schema mappings defined by st-tgds and egds, GCWA*-solutions can be defined in a way similar to the definition of GCWA-solutions. This characterization also shows that, with respect to schema mappings defined by st-tgds and egds, GCWA*-solutions are special GCWA-solutions.

Definition 5.9. For every schema mapping $M = (\sigma, \tau, \Sigma)$ and every source instance S for M, define the following set of $L_{\infty\omega}$ sentences over $\sigma \cup \tau$:

$$D_{M,S}^* := \{ R(\bar{t}) \to \varphi \mid R \in \sigma \cup \tau, \bar{t} \in \mathit{Const}^{\mathrm{ar}(R)}, \text{ and } \varphi \text{ is a monotonic } L_{\infty\omega} \text{ sentence}$$
$$\text{over } \sigma \cup \tau \text{ that is satisfied in every minimal model } I \text{ of } D_{M,S}$$
$$\text{with } \bar{t} \in R^I \}.$$

Proposition 5.10. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ is a set of st-tgds and egds, and let S be a source instance for M. Then for all ground target instances T for M, the following statements are equivalent:*
(1) T is a GCWA*-solution for S under M.
(2) $S \cup T$ is a model of $D_{M,S} \cup D_{M,S}^*$.

Proof. $1 \implies 2$: Suppose that T is a GCWA*-solution for S under M. By Proposition 5.8, T is a ground solution for S under M, and there is a set $\mathcal{T}_0$ of minimal solutions for S under M such that $T = \bigcup \mathcal{T}_0$. We have to show that $I := S \cup T$ satisfies $D_{M,S} \cup D_{M,S}^*$.

Since T is a solution for S under M, we have $I \models D_{M,S}$. Thus, it remains to show that $I \models D_{M,S}^*$.

To this end, consider an arbitrary sentence $\psi := R(\bar{t}) \to \varphi$ in $D_{M,S}^*$, and assume that $I \models R(\bar{t})$. Since $I = S \cup T$ and $T = \bigcup \mathcal{T}_0$, there is some $T_0 \in \mathcal{T}_0$ with $\bar{t} \in R^{S \cup T_0}$. Note that $I_0 := S \cup T_0$ is a minimal model of $D_{M,S}$. By Definition 5.9, we thus have $I_0 \models \varphi$. Since $I_0 \subseteq I$ and φ is monotonic, it follows that $I \models \varphi$. Consequently, I satisfies ψ.

$2 \implies 1$: Suppose that $I := S \cup T$ is a model of $D_{M,S} \cup D_{M,S}^*$. Since models are ground instances by definition, it follows that T is a ground solution for S under M. To show that T is a GCWA*-solution for S under M, it remains to construct, by Proposition 5.8, a set $\mathcal{T}_0$ of minimal solutions for S under M such that $T = \bigcup \mathcal{T}_0$.

Let $\mathcal{T}_0$ be the set of all minimal solutions T_0 for S under M with $T_0 \subseteq T$. We claim that $T = \bigcup \mathcal{T}_0$. By construction, we have $\bigcup \mathcal{T}_0 \subseteq T$. Thus it remains to show that it is not the case that $\bigcup \mathcal{T}_0 \subsetneq T$.

Suppose, to the contrary, that $\bigcup \mathcal{T}_0 \subsetneq T$. Then there are $R \in \tau$ and $\bar{t} \in \mathit{Const}^{\mathrm{ar}(R)}$ such that

$$\bar{t} \in R^T \quad \text{and} \quad \bar{t} \notin R^{T_0} \text{ for all } T_0 \in \mathcal{T}_0. \tag{5.1}$$

On the other hand, there is at least one minimal model I_0 of $D_{M,S}$ with $\bar{t} \in R^{I_0}$. Otherwise, $R(\bar{t}) \to \bigvee \emptyset$, which is equivalent to $\neg R(\bar{t})$, would be in $D_{M,S}^*$, so that $\bar{t} \notin R^I \supseteq R^T$ would contradict (5.1). Let

$$\mathcal{I}_0 := \{ I_0 \mid I_0 \text{ is a minimal model of } D_{M,S} \text{ with } \bar{t} \in R^{I_0} \}.$$

Then,

$$\psi := R(\bar{t}) \to \varphi \quad \text{with} \quad \varphi := \bigvee_{I_0 \in \mathcal{I}_0} \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}')$$

is satisfied in *every* minimal model of $D_{M,S}$. Since φ is monotonic, we thus have $\psi \in D^*_{M,S}$. Furthermore, since $I \models D^*_{M,S}$ and $I \models R(\bar{t})$, it follows that $I \models \varphi$. In particular, there must be some $I_0 \in \mathcal{I}_0$ such that $I \models \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}')$, and thus, $I_0 \subseteq I$. Note that $I_0 = S \cup T_0$ for some $T_0 \in \mathcal{T}_0$. Together with $\bar{t} \in R^{I_0}$ and $R \in \tau$, this implies that $\bar{t} \in R^{T_0}$. However, this contradicts (5.1). Consequently, $\bigcup \mathcal{T}_0 = T$. $\qquad\square$

Moreover, the following result translates [35, Theorem 5] from GCWA-solutions to GCWA*-solutions, and shows that for a given schema mapping M and a source instance S for M, the set $D_{M,S} \cup D^*_{M,S}$ is maximally consistent in the sense that the addition of any sentence ψ of the form $R(\bar{t}) \to \varphi$, where φ is a monotonic $L_{\infty\omega}$ sentence and $D_{M,S} \cup D^*_{M,S} \not\models \psi$, leads to a set of formulas that is inconsistent with $D_{M,S} \cup D^*_{M,S}$.

Proposition 5.11. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a nonempty source instance for M, let $D := D_{M,S}$ and let $D' := D \cup D^*$.*
(1) *For all monotonic $L_{\infty\omega}$-sentences φ over $\sigma \cup \tau$, we have $D \models \varphi$ if and only if $D' \models \varphi$.*
(2) *For all $\psi := R(\bar{t}) \to \varphi$, where $R(\bar{t})$ is a ground atom over $\sigma \cup \tau$, φ is a $L_{\infty\omega}$ sentence over $\sigma \cup \tau$, and $D' \not\models \psi$:*
   (a) *$D' \cup \{\psi\}$ has no model, or*
   (b) *there is a monotonic $L_{\infty\omega}$-sentence χ over $\sigma \cup \tau$ such that $D' \cup \{\psi\} \models \chi$, but $D' \not\models \chi$.*

Proof. Statement 1 is obvious, so in the following we prove 2. Let ψ be given. If $D' \cup \{\psi\}$ has no model, then we are done. So assume that $D' \cup \{\psi\}$ has a model. Let $\mathcal{I}_0$ be the set of all minimal models of $D' \cup \{\psi\}$, and consider the monotonic $L_{\infty\omega}$ sentence

$$\chi := \bigvee_{I_0 \in \mathcal{I}_0} \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}').$$

Clearly, we have $D' \cup \{\psi\} \models \chi$. Indeed, if I is a model of $D' \cup \{\psi\}$, let $I_0 \in \mathcal{I}_0$ be such that $I_0 \subseteq I$. Then, $I_0 \models \bigwedge_{R'(\bar{t}') \in I_0} R'(\bar{t}')$, and therefore, $I_0 \models \chi$. Since χ is monotonic and $I_0 \subseteq I$, this leads to $I \models \chi$.

Furthermore, we have $D' \not\models \chi$. For a contradiction suppose that $D' \models \chi$. Note that there is a minimal model I_0 of D with $I_0 \not\models \psi$. (This follows immediately from $D^* \subseteq D'$ and $D' \not\models \psi$, which imply $\psi \notin D^*$.) Since I_0 is a minimal model of D' as well, and $D' \models \chi$, we have $I_0 \models \chi$. Thus, there is some $I'_0 \in \mathcal{I}_0$ such that $I_0 \models \bigwedge_{R'(\bar{t}') \in I'_0} R'(\bar{t}')$. In other words, $I'_0 \subseteq I_0$, which implies $I'_0 = I_0$, because I_0 is a minimal model of D', and $I'_0 \models D'$. But this is impossible, since $I'_0 \models \psi$ and $I_0 \not\models \psi$. Consequently, $D' \not\models \chi$. $\qquad\square$

6. Data Complexity of Query Evaluation under the GCWA*-Semantics

In this section, we study the data complexity of computing GCWA*-answers, where data complexity means that the schema mapping and the query to be evaluated are fixed (i.e., not part of the input). We concentrate on schema mappings defined by st-tgds only.

Since in data exchange, the goal is to answer queries based on some materialized solution, given a schema mapping $M = (\sigma, \tau, \Sigma)$ and a query language L, we are particularly interested in whether there are algorithms $\mathbb{A}_1, \mathbb{A}_2$ with the following properties:

(1) $\mathbb{A}_1$ takes a source instance S for M as input and computes a solution T for S under M, and

(2) $\mathbb{A}_2$ takes a solution T computed by $\mathbb{A}_1$ and a query $q \in L$ over τ as input and computes $cert_{\mathrm{GCWA}^*}(q, M, S)$.

In particular, $\mathbb{A}_1$ takes care of the actual data exchange (dependent on the query language, but independent of any concrete query), while $\mathbb{A}_2$ answers queries based on some materialized solution. At best, both $\mathbb{A}_1$, and $\mathbb{A}_2$ for fixed $q \in L$, run in polynomial time.

For proving complexity lower bounds, we consider, for fixed schema mappings $M = (\sigma, \tau, \Sigma)$ and queries $q(\bar{x})$ over τ, the decision problem

---

$\mathrm{EVAL}_{\mathrm{GCWA}^*}(M, q)$

Input: a source instance S for M, and a tuple $\bar{t} \in Const^{|\bar{x}|}$

Question: Is $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$?

The complexity of this problem can be seen as a lower bound on the joint complexity of finding a solution T as in step 1 above, and obtaining $cert_{\mathrm{GCWA}^*}(q, M, S)$ from T as in step 2. If, for example, $\mathrm{EVAL}_{\mathrm{GCWA}^*}(M, q)$ is co-NP-complete, then finding T is intractable, or computing $cert_{\mathrm{GCWA}^*}(q, M, S)$ from T is intractable.

We first consider the complexity of computing the GCWA*-answers to monotonic queries and existential queries in Sections 6.1 and 6.2, and deal with the present section's main result concerning universal queries in Section 6.3.

6.1. **Monotonic Queries.** For monotonic queries, all results obtained for the certain answers semantics (see, e.g., [10, 32, 3, 26, 29, 8, 5, 6]) carry over to the GCWA*-answers semantics:

Proposition 6.1. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, let S be a source instance for M, and let $q(\bar{x})$ be a monotonic query over τ. Then, $cert_{GCWA^*}(q, M, S) = cert_{OWA}(q, M, S)$.*

Proof. Since every GCWA*-solution for S under M is a solution for S under M, we have $cert_{\mathrm{OWA}}(q, M, S) \subseteq cert_{\mathrm{GCWA}^*}(q, M, S)$. To show $cert_{\mathrm{GCWA}^*}(q, M, S) \subseteq cert_{\mathrm{OWA}}(q, M, S)$, consider a tuple $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$. We have to show that $\bar{t} \in q(T)$ for all solutions T for S under M. To this end, it suffices to show that $\bar{t} \in q(T)$ for all *ground* solutions T for S under M, since nulls can be seen as special constants. Let T be a ground solution for S under M, and let T_0 be a minimal solution for S under M with $T_0 \subseteq T$. By Definition 5.6, T_0 is a GCWA*-solution for S under M, and since $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$, we have $\bar{t} \in q(T_0)$. Since q is monotonic and $T_0 \subseteq T$, we conclude $\bar{t} \in q(T)$. $\qquad\square$

In particular, if M is a schema mapping defined by st-tgds, and $q(\bar{x})$ is a union of conjunctive queries over M's target schema, then Proposition 2.2 implies that there is a polynomial time algorithm that takes a universal solution for a source instance S for M as input and outputs the GCWA*-answers to $q(\bar{x})$ on M and S. Note that by Theorem 2.1, a universal solution can be computed in polynomial time (for fixed M) from a given source instance for M.

6.2. **Existential Queries and Beyond.** We now turn to *existential queries*, which are FO queries of the form $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$, where φ is quantifier-free. A particular class of existential queries are *conjunctive queries with negation* (CQ$^\neg$ queries, for short), which are queries of the form $q(\bar{x}) = \exists \bar{y}\,(L_1 \wedge \cdots \wedge L_k)$ where each L_i is either a relational atom $R(\bar{u})$ or the negation of a relational atom. A simple reduction from the CLIQUE problem [14] shows that $\text{EVAL}_{\text{GCWA}^*}(M, q)$ can be co-NP-hard for schema mappings M defined by LAV tgds and CQ$^\neg$ queries with only one negated atom:

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

Proof. Let $M = (\sigma, \tau, \Sigma)$, where σ consists of binary relation symbols E_0, C_0, τ consists of binary relation symbols E, C, A, and Σ consists of the following st-tgds:

$$\theta_1 := \forall x \forall y\, \big(E_0(x, y) \rightarrow E(x, y)\big),$$
$$\theta_2 := \forall x \forall y\, \big(C_0(x, y) \rightarrow \exists z_1 \exists z_2\, (C(x, y) \wedge A(x, z_1) \wedge A(y, z_2))\big).$$

Furthermore, let

$$q := \exists x \exists y \exists z_1 \exists z_2\, \big(C(x, y) \wedge A(x, z_1) \wedge A(y, z_2) \wedge \neg E(z_1, z_2)\big).$$

We show that $\text{EVAL}_{\text{GCWA}^*}(M, q)$ is co-NP-complete by showing that the complement of $\text{EVAL}_{\text{GCWA}^*}(M, q)$ is NP-complete.

Membership: The complement of $\text{EVAL}_{\text{GCWA}^*}(M, q)$ is solved by a nondeterministic Turing machine as follows. Given a source instance S for M, the machine needs to decide whether $cert_{\text{GCWA}^*}(q, M, S) = \emptyset$, that is, whether there is a GCWA*-solution T for S under M such that $T \models \neg q$.

Note that

$$\neg q \equiv \forall x \forall y \forall z_1 \forall z_2\, \big(C(x, y) \wedge A(x, z_1) \wedge A(y, z_2) \rightarrow E(z_1, z_2)\big), \qquad (6.1)$$

and that the minimal ground solutions for S under M are all solutions of the form

$$T_f := \big\{E(a, b) \mid (a, b) \in E_0^S\big\} \cup \big\{C(c, c') \mid (c, c') \in C_0^S\big\} \cup \big\{A(c, f(c)) \mid c \in \text{dom}(C_0^S)\big\},$$

for some mapping $f \colon \text{dom}(C_0^S) \rightarrow \textit{Const}$. Hence, if T is a GCWA*-solution for S under M with $T \models \neg q$, there is a minimal ground solution $T' \subseteq T$ for S under M with $T' \models \neg q$. In particular, it suffices to decide whether there is a minimal ground solution T for S under M such that $T \models \neg q$.

By (6.1), if $T_f \models \neg q$ for some $f \colon \text{dom}(C_0^S) \rightarrow \textit{Const}$, then for all $c \in \text{dom}(C_0^S)$ we have $f(c) \in \text{dom}(E_0^S)$. Hence, in order to check whether there is a minimal ground solution T for S under M such that $T \models \neg q$, it suffices to guess a mapping $f \colon \text{dom}(C_0^S) \rightarrow \text{dom}(S)$, and to check whether $T_f \models \neg q$. Clearly, this can be done by a nondeterministic Turing machine in time polynomial in the size of S.

Hardness: To show that the complement of $\text{EVAL}_{\text{GCWA}^*}(M, q)$ is NP-hard, we present a reduction from the NP-complete CLIQUE problem [14]. The CLIQUE problem is to decide, given an undirected graph $G = (V, E)$ without loops and a positive integer k, whether G contains a *clique* of size k. Here, a clique in G is a set $C \subseteq V$ such that for all $u, v \in C$ with $u \neq v$ we have $\{u, v\} \in E$.

Let $G = (V, E)$ be an undirected graph without loops, and let $k \geq 1$ be an integer. If $k = 1$, then G has a clique of size k if and only if V is nonempty, and we can reduce (G, k)

to some predefined fixed source instance S for M such that $cert_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$ if and only if V is nonempty (i.e., a source instance S with $C_0^S = \emptyset$ if V is nonempty, and a source instance S with $C_0^S = \{(c, c)\}$ for some $c \in Const$ if V is empty).

If $k \geq 2$, we reduce (G, k) to the source instance S for M with $E_0^S = E$ and $C_0^S = \{(c_i, c_j) \mid 1 \leq i, j \leq k, \, i \neq j\}$, where $c_1, \ldots, c_k$ are pairwise distinct constants that do not occur in V. We claim that G has a clique of size k if and only if $cert_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$.

"Only if" direction: Let $C = \{v_1, \ldots, v_k\}$ be a clique of size k in G, and let T be the target instance for M with $E^T = E$, $C^T = C_0^S$ and $A^T = \{(c_i, v_i) \mid 1 \leq i \leq k\}$. Then T is a minimal solution for S under M, and, by Definition 5.6, a GCWA*-solution for S under M. Furthermore, we have $T \not\models q$. To see this, note that for all $u, v, w_1, w_2 \in \mathrm{dom}(T)$ with $T \models C(u, v) \wedge A(u, w_1) \wedge A(v, w_2)$, there are distinct $i, j \in \{1, \ldots, k\}$ with $u = c_i$ and $v = c_j$, so that $w_1 = v_i$ and $w_2 = v_j$. Since $v_i, v_j \in C$ and $E^T = E$, we thus have $T \models E(w_1, w_2)$ for all such u, v, w_1, w_2. Since T is a GCWA*-solution for S under M, and $T \not\models q$, we have $cert_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$.

"If" direction: Suppose that $cert_{\mathrm{GCWA}^*}(q, M, S) = \emptyset$. Then there is a GCWA*-solution T for S under M with $T \not\models q$. For all $i \in \{1, \ldots, k\}$, let

$$V_i := \left\{ v \in \mathrm{dom}(T) \mid (c_i, v) \in A^T \right\}.$$

Since $S \cup T \models \theta_2$, each V_i is nonempty. Thus, there is a set $C = \{v_1, \ldots, v_k\}$ such that $v_i \in V_i$ for each $i \in \{1, \ldots, k\}$. Moreover, for all $i, j \in \{1, \ldots, k\}$ with $i \neq j$, we have $(v_i, v_j) \in E$. To see this, observe that $T \models C(c_i, c_j) \wedge A(c_i, v_i) \wedge A(c_j, v_j)$, so that $T \not\models q$ implies $T \models E(v_i, v_j)$. It follows that C is a clique in G of size k (since $k \geq 2$ and G has no loops). $\qquad \square$

Adding only one universal quantifier can make the problem undecidable. Specifically, let us consider $\exists^* \forall$ FO queries, which are FO queries of the form $\exists x_1 \cdots \exists x_k \forall y \, \varphi$, where φ is quantifier-free. Then we have:

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

Proof. Let $M = (\{R\}, \{R_p, R_f\}, \Sigma)$, where R, R_p, R_f are ternary relation symbols, and Σ consists of the st-tgds $\theta_c := \forall \bar{x}(R(\bar{x}) \to R_p(\bar{x}))$ and $\theta_d := \forall \bar{x}(R(\bar{x}) \to \exists \bar{y} \, R_f(\bar{y}))$. Let $\hat{q}$ be a FO query that is true in a target instance T for M precisely if $R_p^T \subseteq R_f^T$, and R_f^T encodes the graph of a total associative function $f \colon B \times B \to B$ for some set B:

$$
\begin{aligned}
\hat{q} \; := \; & \forall \bar{x} \left(R_p(\bar{x}) \to R_f(\bar{x}) \right) \\
& \wedge \; \forall \bar{x} \, \forall y_1 \forall y_2 \left(R_f(\bar{x}, y_1) \wedge R_f(\bar{x}, y_2) \to y_1 = y_2 \right) \\
& \wedge \; \forall x \forall y \left(\varphi_{\mathrm{dom}}(x) \wedge \varphi_{\mathrm{dom}}(y) \to \exists z \, R_f(x, y, z) \right) \\
& \wedge \; \forall x \forall y \forall z \forall u \forall v \forall w \left(R_f(x, y, u) \wedge R_f(u, z, v) \wedge R_f(y, z, w) \to R_f(x, w, v) \right),
\end{aligned}
$$

where, in a target instance T for M,

$$\varphi_{\mathrm{dom}}(x) := \exists x_1 \exists x_2 \exists x_3 \left(R_f(x_1, x_2, x_3) \wedge \bigvee_{i=1}^{3} x = x_i \right)$$

defines the set of all values that occur in R_f^T. Note that the last three lines in the definition of $\hat{q}$ are essentially the target constraints of the schema mapping in [27, Theorem 3.6]. Note

also that the negation of $\hat{q}$ is equivalent to a Boolean $\exists^*\forall$ FO query. Let q be this $\exists^*\forall$ FO query.

To show that $\text{EVAL}_{\text{GCWA}^*}(M, q)$ is undecidable, we reduce the

---

EMBEDDING PROBLEM FOR FINITE SEMIGROUPS

Input: a partial function $p\colon A^2 \to A$, where A is a finite set

Question: Is there a finite set $B \supseteq A$ and a total function $f\colon B^2 \to B$ such that f is associative, and f extends p (i.e., $p(x, y)$ defined implies $f(x, y) = p(x, y)$)?

---

that is known to be undecidable [27], to $\text{EVAL}_{\text{GCWA}^*}(M, q)$. Let $p\colon A^2 \to A$ be a partial function, where A is a finite set. Construct the source instance S for M, where R^S is the graph of p, that is, $R^S = \{(a, b, c) \mid p(a, b) = c\}$. We claim that $cert_{\text{GCWA}^*}(q, M, S) = \emptyset$ if and only if p is a "yes"-instance of the embedding problem for finite semigroups.

Note that the GCWA*-solutions for S under M are all the target instances T for M such that $R_p^T = R^S$, and either (1) $R^S = R_p^T = R_f^T = \emptyset$, or (2) R_f^T is a nonempty finite subset of $Const^3$. Therefore, $cert_{\text{GCWA}^*}(q, M, S) = \emptyset$ if and only if there is a GCWA*-solution T for S under M such that R_f^T is the graph of a total function $f\colon \text{dom}(T)^2 \to \text{dom}(T)$ that is associative and extends p. This is the case precisely if p is a "yes"-instance of the embedding problem for finite semigroups. $\square$

6.3. **Universal Queries.** As we have seen in Section 6.2, computing GCWA*-answers to existential queries may be a difficult task, and even more difficult (if possible at all) if the query additionally contains universal quantifiers.

We now turn to *universal queries*, which are FO queries of the form $q(\bar{x}) = \forall\bar{y}\,\varphi(\bar{x}, \bar{y})$, where φ is quantifier-free. As a general upper bound for such queries with respect to schema mappings defined by st-tgds we obtain:

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

The proof of Proposition 6.4 uses basic ideas from the proof of this section's main result, Theorem 6.6, and is deferred to Section 6.3.4.

In what follows, we prove that for schema mappings defined by st-tgds which are *packed* as defined below, the GCWA*-answers to universal queries can even be computed in polynomial time.

Definition 6.5 (packed st-tgd). An st-tgd $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$ is *packed* if for all distinct atoms $R_1(\bar{u}_1), R_2(\bar{u}_2)$ in ψ, there is a variable in $\bar{z}$ that occurs both in $\bar{u}_1$ and in $\bar{u}_2$.

Notice that the schema mapping defined in the proof of Proposition 6.3 is defined by packed st-tgds.

Although schema mappings defined by packed st-tgds are not as expressive as schema mappings defined by st-tgds, they seem to form an interesting class of schema mappings. Packed st-tgds still allow for non-trivial use of existential quantifiers in the heads of st-tgds. For example, consider a schema mapping M defined by st-tgds $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists\bar{z}\,\psi(\bar{x}, \bar{z}))$, where ψ contains at most two atoms that contain variables from $\bar{z}$. Then M is logically equivalent to a schema mapping defined by packed st-tgds. To see this, let

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

be an st-tgd in M, and let G be the graph whose vertices are the atoms in ψ, and which has an edge between two distinct atoms if they share a variable from $\bar{z}$. Let $C_1, \ldots, C_k$ be the connected components of G, and for every $i \in \{1, \ldots, k\}$ let

$$\theta_i := \forall \bar{x} \forall \bar{y} \left(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z} \, \psi_i \right),$$

where ψ_i is the conjunction of all atoms in C_i. Then θ is logically equivalent to $\{\theta_1, \ldots, \theta_k\}$. Using that ψ contains at most two atoms with variables from $\bar{z}$, it is easy to see that each θ_i is a packed st-tgd. As a special case, it follows that each full st-tgd is equivalent to a set of packed st-tgds. An example of a st-tgd that is *not* packed is $\forall x (P(x) \to \exists z_1 \exists z_2 \exists z_3 (E(x, z_1) \land E(z_1, z_2) \land E(z_2, z_3)))$.

We are now ready to state this section's main result:

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

Note that Theorem 6.6 and Theorem 2.1 immediately imply that for every schema mapping M specified by packed st-tgds, and for every universal query $q(\bar{x})$ over M's target schema, there is a polynomial time algorithm that takes a source instance S for M as input, and outputs $cert_{\mathrm{GCWA}^*}(q, M, S)$. In particular:

Corollary 6.7. *If M is a schema mapping defined by packed st-tgds, and $q(\bar{x})$ is a universal query over M's target schema, then $\mathrm{EVAL}_{GCWA^*}(M, q)$ is in PTIME.*

An interesting consequence of Theorem 6.6 is the following. Let M be a schema mapping defined by packed st-tgds, and let S be a source instance for M. Recall from Section 2 that the OWA-answers to unions of conjunctive queries on M and S can be computed in polynomial time from $\mathrm{Core}(M, S)$ (assuming M and the query are fixed). In other words, we only need to compute $\mathrm{Core}(M, S)$ in order to answer both unions of conjunctive queries, and universal queries. As mentioned above, $\mathrm{Core}(M, S)$ can be computed in polynomial time if M is fixed.

Let us now turn to the proof of Theorem 6.6. Observe that Theorem 6.6 is an immediate consequence of:

Theorem 6.8. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, and let $q(\bar{x})$ be a universal query over τ. Then there is a polynomial time algorithm that, given $\mathrm{Core}(M, S)$ for some source instance S for M, and a tuple $\bar{t} \in \mathrm{Const}^{|\bar{x}|}$, decides whether $\bar{t} \in cert_{GCWA^*}(q, M, S)$.*

The remaining part of this section is devoted to the proof of Theorem 6.8.

6.3.1. *GCWA*-Answers and the Core.* Let us first see how we can decide membership of tuples in $cert_{\mathrm{GCWA}^*}(q, M, S)$ using $\mathrm{Core}(M, S)$. Consider a schema mapping $M = (\sigma, \tau, \Sigma)$, where Σ is a set of packed st-tgds, and let $q(\bar{x})$ be a universal query over τ. Given $\mathrm{Core}(M, S)$ and an $|\bar{x}|$-tuple $\bar{t}$ of constants, how can we decide whether $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$?

First observe that if $\bar{t}$ is not a tuple over $\mathrm{const}(\mathrm{Core}(M, S)) \cup \mathrm{dom}(q)$, then by the definition of $cert_{\mathrm{GCWA}^*}(q, M, S)$ we have $\bar{t} \notin cert_{\mathrm{GCWA}^*}(q, M, S)$. Therefore, in the following we assume that $\bar{t}$ is a tuple over $\mathrm{const}(\mathrm{Core}(M, S)) \cup \mathrm{dom}(q)$. In this case, we have $\bar{t} \notin cert_{\mathrm{GCWA}^*}(q, M, S)$ if and only if there is a GCWA*-solution $\tilde{T}$ for S under M such that $\tilde{T} \models \neg q(\bar{t})$. By the definition of GCWA*-solution and the fact that Σ consists of st-tgds, the

latter is the case precisely if there is a nonempty finite set $\mathcal{T}$ of ground minimal solutions for S under M with $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Using the following lemma, we can reformulate the last condition in terms of $\mathrm{Core}(M, S)$.

Recall the definition of a valuation of an instance T, and the definition of $poss(T)$ from Section 3. Then:

Lemma 6.9. *Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let S be a source instance for M. Then the set of all ground minimal solutions for S under M is precisely the set of all minimal instances in $poss(\mathrm{Core}(M, S))$.*

Proof. Let $T := \mathrm{Core}(M, S)$. We first show that every instance in $poss(T)$ is a ground solution for S under M. Let $\hat{T}$ be an instance in $poss(T)$. Then there is a valuation v of T with $v(T) = \hat{T}$. This shows that $\hat{T}$ is ground. To see that $\hat{T}$ satisfies all st-tgds in Σ, let $\theta := \forall \bar{x} \forall \bar{y}(\varphi(\bar{x}, \bar{y}) \to \exists \bar{z} \psi(\bar{x}, \bar{z}))$ be a st-tgd in Σ, and let $\bar{a}, \bar{b}$ be tuples with $S \models \varphi(\bar{a}, \bar{b})$. Since $S \cup T \models \theta$, there is a tuple $\bar{t}$ with $T \models \psi(\bar{a}, \bar{t})$, and thus $\hat{T} \models \psi(\bar{a}, v(\bar{t}))$. Altogether, $\hat{T}$ is a ground solution for S under M.

It remains to show that every ground minimal solution for S under M is in $poss(T)$. Let T_0 be a ground minimal solution for S under M. It is not hard to verify that there is a valuation v_0 of $T^* := \mathrm{CanSol}(M, S)$ with $v_0(T^*) = T_0$ (see also [29]). Since T^* is a universal solution for S under M, we have $T = \mathrm{Core}(T^*)$, and thus $\iota(T) \subseteq T^*$ for some injective mapping $\iota\colon \mathrm{dom}(T) \to \mathrm{dom}(T)$ that is legal for T. Let $v := v_0 \circ \iota$. Then,

$$v(T) = v_0(\iota(T)) \subseteq v_0(T^*) = T_0. \tag{6.2}$$

Note that $v(T) \in poss(T)$. Therefore, as shown above, $v(T)$ is a solution for S under M. Since T_0 is a minimal solution for S under M, (6.2) implies $v(T) = T_0$. Thus, v is a valuation of T with $v(T) = T_0$, which proves that $T_0 \in poss(T)$. $\qquad \square$

Given $\mathrm{Core}(M, S)$ and $\bar{t}$, it remains to decide whether there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(\mathrm{Core}(M, S))$ such that $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Note that, since q is a universal query, $\neg q$ is logically equivalent to a query of the form $\exists \bar{y}\, \varphi(\bar{x}, \bar{y})$. Before we consider the general case (where φ is an arbitrary quantifier-free query) in Section 6.3.3, the following section deals with the case that $\bar{y}$ contains no variable and φ consists of a single atom $R(\bar{u})$, where $\bar{u}$ is a tuple of constants. In this case, the problem simplifies to: Is there a minimal instance in $poss(\mathrm{Core}(M, S))$ that contains $R(\bar{u})$?

6.3.2. *Finding Atoms in Minimal Instances.* Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of packed st-tgds, let $T := \mathrm{Core}(M, S)$ for some source instance S for M, and let $R(\bar{t})$ be an atom over τ. In the following, we consider the problem of testing whether there is a minimal instance T_0 in $poss(T)$ with $R(\bar{t}) \in T_0$. We will often state results in a more general form than necessary, so that we can apply those results later in the more general setting considered in Section 6.3.3.

First note that there may be infinitely many minimal instances in $poss(T)$, so that it is impossible to check out all these instances. However, it suffices to consider representatives of the minimal instances in $poss(T)$, where constants that do not occur in T or $R(\bar{t})$ are represented by nulls in T. Denoting by C the set of all constants in $\bar{t}$, the set $min_C(T)$ of all such representatives is formally defined as follows:

Definition 6.10 ($val_C(T)$, $min_C(T)$)**.** Let T be an instance, and let $C \subseteq Const$.

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

Throughout this section, C will usually be the set of constants in $\bar{t}$.

Proposition 6.11. *Let T be an instance, and let $C \subseteq Const$.*
(1) *For each $T_0 \in poss(T)$, the following are equivalent:*
    (a) *T_0 is a minimal instance in $poss(T)$.*
    (b) *There is an instance $T_0' \in min_C(T)$ and an injective valuation v of T_0' such that $v(T_0') = T_0$, and $v^{-1}(c) = c$ for all $c \in \mathrm{dom}(T_0) \cap C$.*
(2) *If T is a core, then $T \in min_C(T)$.*
(3) *Each instance in $min_C(T)$ is a core.*

Proof. Ad 1: We first prove that 1a implies 1b. Suppose that T_0 is a minimal instance in $poss(T)$, and let v_0 be a valuation of T with $v_0(T) = T_0$. Furthermore, let $\bar{v}\colon \mathrm{dom}(T_0) \to \mathrm{dom}(T) \cup C$ be an injective mapping with

$$\bar{v}(c) = c \quad \text{for each } c \in \mathrm{dom}(T_0) \cap (\mathrm{const}(T) \cup C), \tag{6.3}$$

and

$$\bar{v}(c) \in \mathrm{nulls}(T) \quad \text{for each } c \in \mathrm{dom}(T_0) \setminus (\mathrm{const}(T) \cup C). \tag{6.4}$$

Then $f := \bar{v} \circ v_0 \in val_C(T)$, and

$$T_0' := f(T) = \bar{v}(v_0(T)) = \bar{v}(T_0). \tag{6.5}$$

Let v be the inverse of $\bar{v}$ on $\mathrm{dom}(T_0')$. Then, by (6.3)–(6.5), v is an injective valuation of T_0' such that $v(T_0') = T_0$, and $v^{-1}(c) = \bar{v}(c) = c$ for every $c \in \mathrm{dom}(T_0) \cap C$.

It remains to show that $T_0' \in min_C(T)$. By (6.5), we have $T_0' = f(T)$, where $f \in val_C(T)$. Suppose, for a contradiction, that there is an $f' \in val_C(T)$ with $f'(T) \subsetneqq T_0'$. Since v is injective and $v(T_0') = T_0$, we then have $v(f'(T)) \subsetneqq v(T_0') = T_0$, which is impossible, since $v(f'(T)) \in poss(T)$, and T_0 is a minimal instance in $poss(T)$.

We next prove that 1b implies 1a. Suppose that $T_0' \in min_C(T)$ and that v is an injective valuation of T_0' with $v(T_0') = T_0$ (we will not need the restriction that $v^{-1}(c) = c$ for all $c \in \mathrm{dom}(T_0) \cap C$). We show that T_0 is a minimal instance in $poss(T)$. To this end, let $f \in val_C(T)$ be such that $f(T) = T_0'$. Then $v_0 := v \circ f$ is a valuation of T, so that

$$T_0 = v(T_0') = v(f(T)) = v_0(T) \in poss(T).$$

It remains, therefore, to show that there is no $\tilde{T}_0 \in poss(T)$ with $\tilde{T}_0 \subsetneqq T_0$.

Suppose, to the contrary, that there is such a $\tilde{T}_0$. Let $\tilde{v}_0$ be a valuation of T with $\tilde{v}_0(T) = \tilde{T}_0$, and let $\tilde{f} := v^{-1} \circ \tilde{v}_0$, where v^{-1} is the inverse of v on $\mathrm{dom}(T_0)$. Since v^{-1} is an injective mapping on $\mathrm{dom}(T_0)$, we have $\tilde{f}(T) = v^{-1}(\tilde{v}_0(T)) = v^{-1}(\tilde{T}_0) \subsetneqq v^{-1}(T_0) = T_0'$, which is impossible, since $\tilde{f} \in val_C(T)$ and $T_0' \in min_C(T)$.

Ad 2: Clearly, the identity f on $\mathrm{dom}(T)$ belongs to $val_C(T)$ and satisfies $f(T) = T$. Let $f' \in val_C(T)$ be such that $f'(T) \subseteq T$. Then f' is a homomorphism from T to T, and since T is a core, we cannot have $f'(T) \subsetneqq T$.

Ad 3: Let $f \in val_C(T)$ be such that $T_0 := f(T) \in min_C(T)$. For a contradiction, suppose that T_0 is not a core. Let h be a homomorphism from T_0 to T_0 such that $h(T_0)$ is a

core of T_0. Since T_0 is not a core, we have $h(T_0) \subsetneq T_0$. Thus, for $f' := h \circ f$, we have $f'(T) = h(f(T)) = h(T_0) \subsetneq T_0$, which contradicts $T_0 \in min_C(T)$. Hence, T_0 is a core. $\qquad\square$

The converse of Proposition 6.11(3) is not true, as shown by the following example:

Example 6.12. Let T be an instance over $\sigma = \{E, P\}$, where $E^T = \{(a, \bot), (\bot, \bot')\}$ and $P^T = \{b\}$. The mapping $f \in val_C(T)$ with $f(\bot) = a$ and $f(\bot') = b$ then yields the instance $f(T)$, where $E^{f(T)} = \{(a, a), (a, b)\}$ and $P^{f(T)} = \{b\}$. Hence, $f(T)$ is a core. However, $f(T)$ does not belong to $min_C(T)$, since the mapping $f' \in val_C(T)$ with $f'(\bot) = f'(\bot') = a$ yields the instance $f'(T)$ with $E^{f'(T)} = \{(a, a)\}$ and $P^{f'(T)} = \{b\}$, which is a proper subinstance of $f(T)$.

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

Definition 6.13 (atom block [16]). Let T be an instance.
- The *Gaifman graph of the atoms of* T is the undirected graph whose vertices are the atoms of T, and which has an edge between two atoms $A, A' \in T$ if and only if $A \neq A'$, and there is a null that occurs both in A and A'.
- An *atom block* of T is the set of atoms in a connected component of the Gaifman graph of the atoms of T.

Note that each atom block of T is a subinstance of T. Furthermore, for each atom block B of T that contains at least one null, nulls(B) is a *block* as considered in [12]. The crucial property of T is:

Lemma 6.14 ([12]). *For every schema mapping $M = (\sigma, \tau, \Sigma)$, where Σ consists of st-tgds, there is a positive integer bs such that if S is a source instance for M, and B is an atom block of* $\mathrm{Core}(M, S)$*, then* $|\mathrm{nulls}(B)| \leq bs$.

Let us come back to our initial problem – to decide whether there is a minimal instance in $poss(\mathrm{Core}(M, S))$ that contains the ground atom $R(\bar{t})$. Let $T := \mathrm{Core}(M, S)$, and let C be the set of constants in $\bar{t}$. By Proposition 6.11(1) it is enough to decide whether there is a $T_0 \in min_C(T)$ with $R(\bar{t}) \in T_0$. The following algorithm seems to accomplish this task:
(1) Compute the atom blocks of T.
(2) Consider the atom blocks B of T in turn, and
(3) if there is an instance $B_0 \in min_C(B)$ with $R(\bar{t}) \in B_0$, accept the input; otherwise reject it.
Since, by Lemma 6.14, there is a constant bs with $|\mathrm{nulls}(B)| \leq bs$ for each atom block B of T, we have to consider at most $|val_C(B)| = |\mathrm{dom}(B) \cup C|^{bs}$ mappings in step 3 to find all the instances $B_0 \in min_C(B)$. Thus, the whole algorithm runs in polynomial time.

Example 6.15 below shows that this algorithm is incorrect. In particular, the example exhibits an instance T that is a core, and an atom block B of T such that there is an atom A of some minimal instance $B_0 \in poss(B)$ that is not an atom of any minimal instance in $poss(T)$. Letting C be the set of all constants in A, this implies that there is an atom of some instance $B_0 \in min_C(B)$ that is not an atom of any instance in $min_C(T)$.

Example 6.15. Let T be the instance over $\{E\}$ with

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

and consider the atom block

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

of T; see Figure 1 for a graph representation of T and B. Note that T is a core. It is not

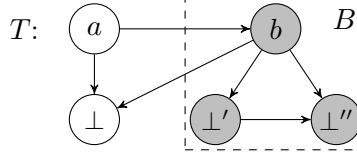Figure 1: The instance T. The subinstance induced by the gray vertices is B.

hard to see that every minimal instance in $poss(B)$ has one of the following forms:

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

Thus, there is a minimal instance in $poss(B)$ of the third form that contains $E(c,a)$ for some $c \in Const \setminus \{b\}$ (replace c' in 3 with a).

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

$$v'(T) = \{E(a,b), E(b,a), E(a,v(\perp)), E(b,v(\perp))\}$$
$$\subsetneq \{E(a,b), E(b,a), E(a,v(\perp)), E(b,v(\perp)), E(b,c), E(c,a)\} = v(T).$$

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

It is nevertheless possible to solve our initial problem using the following approach. Let $T = \mathrm{Core}(M, S)$, let $R(\bar{t})$ be a ground atom, and let C be the set of all constants in $\bar{t}$. Our goal is to decide whether there is an instance $T_0 \in min_C(T)$ with $R(\bar{t}) \in T_0$. To this end, we identify a set $\mathcal{S} \subseteq min_C(T)$ of size polynomial in the size of T such that $R(\bar{t})$ occurs in an instance in $min_C(T)$ if and only if $R(\bar{t})$ occurs in an instance in $\mathcal{S}$. Furthermore, we ensure that $\mathcal{S}$ can be computed in polynomial time from T and C. To define $\mathcal{S}$, we need a few definitions.

In the following, we fix, for each instance I, a core $\mathrm{Core}(I) \subseteq I$, namely the output of the algorithm provided by the following lemma:

Lemma 6.16 (implicit in [12]). *There is an algorithm that takes an instance I as input, and outputs a core $J \subseteq I$ of I in time $O(n^{b+3})$, where n is the size of I and b is the maximum number of nulls in an atom block of I.*

Proof. Just omit the first step of the *blocks algorithm* from [12]. That is, given an instance I, proceed as follows:

(1) Compute a list $B_1, \ldots, B_m$ of all atom blocks of I, and initialize J to be I.
(2) Check whether there is a homomorphism h from J to J such that h is not injective, and there is some $i \in \{1, \ldots, m\}$ such that $h(u) = u$ for each $u \in \mathrm{dom}(J) \setminus \mathrm{nulls}(B_i)$.

(3) If such a h exists, replace J by $h(J)$, and go to step 2.

(4) Output J.

Now the lemma follows from the proof of [12, Theorem 5.9]. □

Given $T = \mathrm{Core}(M, S)$ and C as above, we define the set $\mathcal{S}$ to be the union of the following sets $min_C(T, B)$ over all atom blocks B of T.

Definition 6.17 ($minval_C(T, B)$, $min_C(T, B)$)**.** Let T be an instance, let B be an atom block of T, let $\overline{B} := T \setminus B$, and let $C \subseteq \mathit{Const}$.

(1) Let $val_C(T, B)$ be the set of all mappings $f \in val_C(T)$ such that
   - $f(\bot) = \bot$ for all $\bot \in \mathrm{nulls}(\overline{B})$, and
   - all nulls that occur in $f(B) \setminus \overline{B}$ belong to $\mathrm{nulls}(B)$.

(2) Let $minval_C(T, B)$ be the set of all $f \in val_C(T, B)$ such that there is no $f' \in val_C(T, B)$ with $f'(T) \subsetneq f(T)$.

(3) Let $min_C(T, B) := \{\mathrm{Core}(f(T)) \mid f \in minval_C(T, B)\}$.

Using Lemma 6.16, we obtain:

Proposition 6.18. *For each positive integer bs, there is a polynomial time algorithm that, given an instance T such that the number of nulls in each atom block of T is at most bs, and a set $C \subseteq \mathit{Dom}$, outputs a list of all instances that occur in $min_C(T, B)$ for some atom block B of T.*

Proof. The algorithm is as follows: Given T and C, first compute a list $f_1, \ldots, f_m$ of all mappings f such that there is an atom block B of T with $f \in minval_C(T, B)$. This can be done in time polynomial in the size of T. Then, compute and output $\mathrm{Core}(f_i(T))$ for each $i \in \{1, \ldots, m\}$. By Lemma 6.16, this can be done in time $O(n^{bs+3})$, where n is the size of T (note that the number of nulls in each atom block of $f_i(T)$ is bounded by bs). □

The following Lemma 6.19 tells us that the instances in $min_C(T, B)$ indeed belong to $min_C(T)$. Before stating the lemma, let us introduce retractions. Given an instance I, a *retraction* of I is a homomorphism h from I to I such that $h(u) = u$ for all elements u in the range of h. In particular, for all atoms $A \in h(I)$, we have $A \in I$ and $h(A) = A$. It is known that a core of I is an instance J for which there is a retraction h of I with $h(I) = J$, and there is no retraction of J to a proper subinstance of J ([19]). A *retraction of I over a set* $X \subseteq \mathit{Dom}$ is a retraction h of I such that $h(u) = u$ for each $u \in X \cap \mathrm{dom}(I)$.

Lemma 6.19. *Let T be an instance, let B be an atom block of T, let $\overline{B} := T \setminus B$, and let $C \subseteq \mathit{Const}$. Then for each $f \in minval_C(T, B)$, there is a retraction h of $\hat{T} := f(T)$ over the set of the nulls of $f(B) \setminus \overline{B}$ such that*

(1) $h(\hat{T})$ *is a core of $\hat{T}$, and*

(2) $h(\hat{T}) \in min_C(T)$.

In particular, $min_C(T, B) \subseteq min_C(T)$.

Proof. Let $\mathcal{A} := f(B) \setminus \overline{B}$, and let h be a retraction of $\hat{T}$ over $\mathrm{nulls}(\mathcal{A})$ such that for

$$\hat{T}_0 := h(\hat{T}) = h(f(T)) \tag{6.6}$$

we have:

$$\text{There is no retraction } h' \text{ of } \hat{T}_0 \text{ over } \mathrm{nulls}(\mathcal{A}) \text{ with } h'(\hat{T}_0) \subsetneq \hat{T}_0. \tag{6.7}$$

We show that $\hat{T}_0$ is a core of $\hat{T}$, and that $\hat{T}_0 \in min_C(T)$.

Step 1: $\hat{T}_0$ is a core of $\hat{T}$.
Suppose, for a contradiction, that $\hat{T}_0$ is not a core of $\hat{T}$. Then there is a retraction h' of $\hat{T}_0$ with $h'(\hat{T}_0) \subsetneq \hat{T}_0$. By (6.7), there is some $\perp \in \text{nulls}(\mathcal{A})$ with $h'(\perp) \neq \perp$. Let A be an atom in $\mathcal{A}$ that contains $\perp$. Since $h'(\perp) \neq \perp$ and h' is a retraction, $\perp$ does not occur in the range of h', and therefore A does not occur in $h'(\mathcal{A})$. Together with $h'(\mathcal{A}) \subseteq \mathcal{A} \cup \overline{B}$ and $A \in \mathcal{A}$, this implies

$$h'(\mathcal{A}) \setminus \overline{B} \subsetneq \mathcal{A}. \tag{6.8}$$

Consider the mapping $f' \colon \text{dom}(T) \to \text{dom}(T) \cup C$ defined for each $u \in \text{dom}(T)$ by

$$f'(u) := \begin{cases} h'\big(h(f(u))\big), & \text{if } u \in \text{nulls}(B), \\ u, & \text{otherwise.} \end{cases}$$

Then $f' \in val_C(T, B)$, since $f \in val_C(T, B)$ and h, h' are retractions. Moreover,

$$f'(B) \subseteq h'(\hat{T}_0) = h'(\hat{T}_0 \setminus \overline{B}) \cup h'(\overline{B}) \subseteq h'(\hat{T}_0 \setminus \overline{B}) \cup \overline{B}, \tag{6.9}$$

where the first inclusion holds due to $f'(B) = h'(h(f(B))) \subseteq h'(h(f(T)))$ and (6.6), and the last inclusion holds, because h' is a retraction of $\hat{T}_0$.

Observe also that

$$\hat{T}_0 \setminus \overline{B} \subseteq \mathcal{A}. \tag{6.10}$$

Indeed, let A be an atom of $\hat{T}_0$ with $A \notin \overline{B}$. By (6.6), we have $\hat{T}_0 = h(f(T))$. Together with $f(T) = f(B) \cup f(\overline{B})$, and the fact that h is a retraction of $f(T)$, this implies that $A \in f(B)$ or $A \in f(\overline{B})$. Note that $A \notin f(\overline{B})$, because $f(\overline{B}) = \overline{B}$ and $A \notin \overline{B}$. Hence, $A \in f(B)$, which, together with $A \notin \overline{B}$, implies that $A \in \mathcal{A}$.

Consequently, we have

$$
\begin{aligned}
f'(T) &= f'(B) \cup \overline{B} && \text{since } T = B \cup \overline{B} \text{ and } f' \in val_C(T, B) \\
&\subseteq h'(\mathcal{A}) \cup \overline{B} && \text{by (6.9) and (6.10)} \\
&\subsetneq \mathcal{A} \cup \overline{B} && \text{by (6.8)} \\
&= f(B) \cup f(\overline{B}) && \text{by definition of } \mathcal{A}, \text{ and } f \in val_C(T, B) \\
&= f(T),
\end{aligned}
$$

which contradicts $f \in minval_C(T, B)$. Thus, $\hat{T}_0$ is a core of $\hat{T}$.

Step 2: $\hat{T}_0 \in min_C(T)$.
First observe that $\hat{T}_0 = f_0(T)$, where $f_0 := h \circ f$ and $f_0 \in val_C(T)$. It remains, therefore, to show that there is no $f' \in val_C(T)$ with $f'(T) \subsetneq \hat{T}_0$.

Suppose, for a contradiction, that there is such a mapping f'. Without loss of generality, we may assume that $f'(T) \in min_C(T)$. Moreover,

$$f'(T) \subseteq \hat{T}_0 \overset{(6.6)}{=} h(f(T)) \subseteq f(T) = f(B) \cup \overline{B}. \tag{6.11}$$

Next observe that

$$f'(B) \setminus \overline{B} = f(B) \setminus \overline{B}. \tag{6.12}$$

Otherwise, we could use f' to construct a mapping $f'' \in val_C(T, B)$ such that $f''(T) \subsetneq f(T)$, which is impossible, because $f \in minval_C(T, B)$. Indeed, assume that (6.12) is not true. Then,

$$f'(B) \setminus \overline{B} \subsetneq f(B) \setminus \overline{B}, \tag{6.13}$$

since by (6.11) we have $f'(B) \subseteq f(B) \cup \overline{B}$. Let $f'' \colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ be such that for each $u \in \mathrm{dom}(T)$, $f''(u) = f'(u)$ if $u \in \mathrm{nulls}(B)$, and $f''(u) = u$ otherwise. Then it is not hard to see that $f'' \in val_C(T, B)$. Moreover, we have

$$f''(T) = (f'(B) \setminus \overline{B}) \cup \overline{B} \overset{(6.13)}{\underset{\subsetneq}{}} (f(B) \setminus \overline{B}) \cup \overline{B} = f(T),$$

as claimed.

Now, (6.12) implies that the mapping $h' \colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ that is defined for each $u \in \mathrm{dom}(T)$ by $h'(u) = u$ if $u \in \mathrm{nulls}(B)$, and $h'(u) = f'(u)$ otherwise, is a homomorphism from $f(T) = f(B) \cup \overline{B}$ to $f'(T)$. Furthermore, by (6.11) we have $f'(T) \subseteq f(T)$, and therefore, the identity on $\mathrm{dom}(f'(T))$ is a homomorphism from $f'(T)$ to $f(T)$. This implies that $f'(T)$ and $f(T)$ are homomorphically equivalent, and therefore, their cores are isomorphic. Since $\hat{T}_0$ is a core of $f(T)$ as shown in Step 1, and $f'(T)$ is a core by Proposition 6.11(3), we have $\hat{T}_0 \cong f'(T)$. However, this is a contradiction to our earlier assumption that $f'(T) \subsetneq \hat{T}_0$. $\square$

Clearly, the union of the sets $min_C(T, B)$ over all atom blocks B of T does not cover the whole set of instances in $min_C(T)$. However, Lemma 6.23 below tells us that for each atom A of some instance $T_0 \in min_C(T)$ there is an atom block B of T and an instance $T_B \in min_C(T, B)$ that contains an atom A' isomorphic to A in the following sense:

Notation 6.20. We say that two atoms A_1, A_2 are *isomorphic*, and we write $A_1 \cong A_2$, if the instances $\{A_1\}$ and $\{A_2\}$ are isomorphic.

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

Lemma 6.23 is based on the following notion of a *packed* atom block:

Definition 6.21 (packed atom block)**.** An atom block B of an instance is called *packed* if for all atoms $A, A' \in B$ with $A \neq A'$, there is a null that occurs both in A and A'.
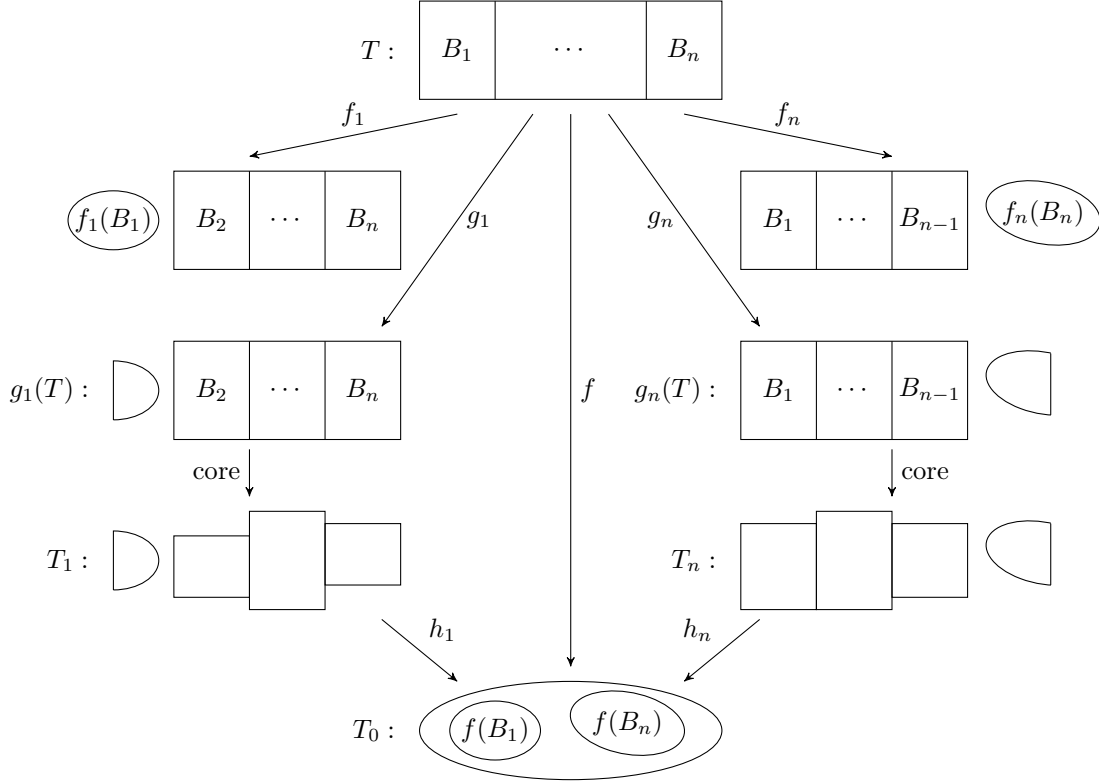
Immediately from the definitions, we obtain:

Proposition 6.22. *If $M = (\sigma, \tau, \Sigma)$ is a schema mapping, where Σ consists of packed st-tgds, and S is a source instance for M, then each atom block of $\mathrm{Core}(M, S)$ is packed.*

We are now ready to state the main result of the present Section 6.3.2:

Lemma 6.23. *Let T be an instance such that T is a core, and each atom block of T is packed. Let $C \subseteq Const$, and let $T_0 \in min_C(T)$. Then for each atom $A \in T_0$, there are*

(1) *an atom block B of T,*
(2) *an instance $T_B \in min_C(T, B)$,*
(3) *an atom $A' \in T_B$ with $A' \cong A$, and*
(4) *a homomorphism h from T_B to T_0 with $h(T_B) = T_0$ and $h(A') = A$.*

Figure 2: The mappings f, f_i, g_i, h_i and their relations.

Proof. Let T be an instance such that T is a core and each atom block of T is packed. Let $C \subseteq Const$, and let $T_0 \in min_C(T)$. Furthermore, let $B_1, \ldots, B_n$ be an enumeration of all the atom blocks of T. We prove the following stronger statement:

> Let $i \in \{1, \ldots, n\}$. Then there is an instance $T_i \in min_C(T, B_i)$ and a homomorphism h_i from T_i to T_0 with $h_i(T_i) = T_0$ such that the following is true: For each atom $A \in T_0$, there is an index $j \in \{1, \ldots, n\}$ and an atom $A' \in T_j$ with $h_j(A') = A$ and $A' \cong A$.   ($\star$)

Idea of the construction.
We start with an $f \in val_C(T)$ such that $f(T) = T_0$. The first step is to find mappings $f_1 \in val_C(T, B_1), \ldots, f_n \in val_C(T, B_n)$ such that each $f_i(B_i)$ is isomorphic to $f(B_i)$. This is easy since we can assign an "unused" null in B_i to any null in B_i that is mapped by f to a null outside B_i.

We then "minimize" each $f_i(T)$ by picking a $g_i \in minval_C(T, B_i)$ with $g_i(T) \subseteq f_i(T)$. The instance T_i is then defined to be the core of $g_i(T)$ (that contains $g_i(B_i) \setminus (T \setminus B_i)$). It is then not hard to define a homomorphism h_i from T_i to T_0 with $h_i(T_i) = T_0$; see Figure 2 for an illustration.

Finally, we have to show that for each atom $A \in T_0$, there is a $j \in \{1, \ldots, n\}$ and an atom $A' \in T_j$ with $h_j(A') = A$ and $A' \cong A$. This is not an immediate consequence of the construction of T_i and g_i. To explain the problem, let us pick an atom $A \in T_0$. This atom must occur in some $f(B_j)$, possibly in more than one such $f(B_j)$. Let $j \in \{1, \ldots, n\}$ be

such that $A \in f(B_j)$. It will be clear from the construction of the f_j and h_j that there is an atom $A' \in f_j(B_j)$ with $A' \cong A$. If $A' \in g_j(B_j)$ and $A' \notin T \setminus B_j$, we are done: in this case we will have $A' \in T_j$ and $h_j(A') = A$. However, if $A' \notin g_j(B_j)$ or $A' \in T \setminus B_j$, it may be that T_j contains no atom isomorphic to A, or – in the case that T contains such an atom, say A'' – that h_j does not map A'' to A.

The extreme case is that for all $j \in \{1, \ldots, n\}$ with $A \in f(B_j)$, there is no atom $A' \in g_j(B_j)$ with $A' \notin T \setminus B_j$ and $A' \cong A$. Using the property that all of the atom blocks $B_1, \ldots, B_n$ are packed, we show that this case does not occur, whereby proving $(\star)$. (Example 6.25 below shows that it may occur if the atom blocks are not packed.)

It is helpful here to think in terms of the following graph G: the nodes of G are the atoms of T, and there is an edge from a node $A' \in B_j$ to a node A'' if $g_j(A') \in T \setminus B_j$ and $g_j(A') = A''$. The core of the proof can then be summarized as follows: We first show that any path in G must eventually reach a node $A' \in B_j$ for some $j \in \{1, \ldots, n\}$ such that $g_j(A') \notin T \setminus B_j$. Otherwise, there would be a cycle in G containing a node A'. This would imply that $A'' := g_j(A')$ is isomorphic to A', and that $A'' \notin B_j$. But since B_j is packed, this implies that g_j is actually a homomorphism from B_j to $T \setminus B_j$, which is impossible since T is a core. Using this property, we then construct – basically by repeated application of the mappings $g_1, \ldots, g_n$ followed by a renaming of the nulls – a mapping $f' \in val_C(T)$ such that $T' := f'(T) \subseteq T_0$. If there are no $j \in \{1, \ldots, n\}$ and $A' \in T_j$ with $h_j(A') = A$ and $A' \cong A$, we will have $A \notin T'$, which would mean that $T' \subsetneq T_0$ and contradict $T_0 \in min_C(T)$. Consequently, there is a $j \in \{1, \ldots, n\}$ and an atom $A' \in T_j$ with $h_j(A') = A$ and $A' \cong A$, which proves $(\star)$.

The details.
Let $f \in val_C(T)$ be such that $f(T) = T_0$. The construction of the instances T_i and the homomorphisms h_i proceeds in three steps. First, we "split" f into mappings $f_1 \in val_C(T, B_1), \ldots, f_n \in val_C(T, B_n)$ such that each $f_i(B_i)$ is isomorphic to $f(B_i)$. Second, we use these mappings to construct the instances T_i and the homomorphisms h_i. Third, we show that for each atom $A \in T_0$, there is a $j \in \{1, \ldots, n\}$ and an atom $A' \in T_j$ with $h_j(A') = A$ and $A' \cong A$.

Step 1: Construction of $f_1, \ldots, f_n$.
Let $i \in \{1, \ldots, n\}$. We construct a mapping $f_i \in val_C(T, B_i)$ and an injective homomorphism r_i from $f_i(B_i)$ to $f(B_i)$ with $r_i(f_i(B_i)) = f(B_i)$ as follows. Pick an injective mapping

$$\bar{r}_i \colon \mathrm{dom}(f(B_i)) \to \mathrm{const}(f(B_i)) \cup \mathrm{nulls}(B_i)$$

such that $\bar{r}_i(c) = c$ for each $c \in \mathrm{const}(f(B_i))$, and $\bar{r}_i(\bot) \in \mathrm{nulls}(B_i)$ for each $\bot \in \mathrm{nulls}(f(B_i))$. Then define $f_i \colon \mathrm{dom}(T) \to \mathrm{dom}(T) \cup C$ such that for each $u \in \mathrm{dom}(T)$,

$$f_i(u) := \begin{cases} \bar{r}_i(f(u)), & \text{if } u \in \mathrm{nulls}(B_i) \\ u, & \text{otherwise.} \end{cases}$$

By construction, we have $f_i \in val_C(T, B_i)$. Furthermore, for each atom A of $f(B_i)$, we have $\bar{r}_i(A) \cong A$. In particular, each atom of $f(B_i)$ is isomorphic to an atom of $f_i(B_i)$, and vice versa. Let r_i be the inverse of $\bar{r}_i$ on $\mathrm{dom}(f_i(B_i))$. Then, r_i is an injective homomorphism from $f_i(B_i)$ to $f(B_i)$ with

$$r_i(f_i(B_i)) = r_i(\bar{r}_i(f(B_i))) = f(B_i). \tag{6.14}$$

In particular,

$$r_i(A) \cong A \quad \text{for all atoms } A \in f_i(B_i). \tag{6.15}$$

Step 2: Construction of the instances T_i and the homomorphisms h_i.
Let $i \in \{1, \ldots, n\}$, and pick $g_i \in minval_C(T, B_i)$ with $g_i(T) \subseteq f_i(T)$. By Lemma 6.19, there is a retraction h_i' of $g_i(T)$ over the set of the nulls of $g_i(B_i) \setminus (T \setminus B_i)$ such that

$$T_i := h_i'(g_i(T)) \in min_C(T). \tag{6.16}$$

Define $h_i \colon \text{dom}(g_i(T)) \to \text{dom}(T_0)$ such that for each $u \in \text{dom}(g_i(T))$,

$$h_i(u) := \begin{cases} r_i(u), & \text{if } u \in \text{dom}(g_i(T) \setminus (T \setminus B_i)) \\ f(u), & \text{otherwise.} \end{cases}$$

Note that r_i is defined for all values that occur in $\text{dom}(g_i(T) \setminus (T \setminus B_i))$, since $g_i(T) \subseteq f_i(T) = f_i(B_i) \cup (T \setminus B_i)$, and therefore,

$$g_i(T) \setminus (T \setminus B_i) \subseteq f_i(B_i). \tag{6.17}$$

Furthermore,

$$h_i(g_i(T) \setminus (T \setminus B_i)) = r_i(g_i(T) \setminus (T \setminus B_i)) \stackrel{(6.17)}{\subseteq} r_i(f_i(B_i)) \stackrel{(6.14)}{=} f(B_i) \subseteq T_0,$$

and

$$h_i(T \setminus B_i) = f(T \setminus B_i) \subseteq f(T) = T_0,$$

which yields $h_i(g_i(T)) \subseteq T_0$. In particular,

$$h_i(h_i'(g_i(T))) \subseteq h_i(g_i(T)) \subseteq T_0.$$

Since $h_i \circ h_i' \circ g_i \in val_C(T)$ and $T_0 \in min_C(T)$, we have $h_i(h_i'(g_i(T))) = T_0$, and hence,

$$h_i(T_i) \stackrel{(6.16)}{=} h_i(h_i'(g_i(T))) = T_0.$$

Let $\tilde{h}_i$ be the restriction of h_i to $\text{dom}(T_i)$. Then, clearly, $\tilde{h}_i$ is a homomorphism from T_i to T_0 with $\tilde{h}_i(T_i) = T_0$.

Step 3: For each $A \in T_0$ there are $j \in \{1, \ldots, n\}$ and $A' \in T_j$ with $h_j(A') = A$ and $A' \cong A$.
Let

$$T^* := \bigcup_{i=1}^{n} \big(g_i(B_i) \setminus (T \setminus B_i)\big).$$

To prove $(\star)$, it suffices to show that there is a mapping $r \colon \text{dom}(T^*) \to \text{dom}(T_0)$ with
(1) $r(T^*) = T_0$,
(2) $r(A') \cong A'$ for each $A' \in T^*$, and
(3) $r(A') = h_i(A')$ for each $i \in \{1, \ldots, n\}$ and each $A' \in g_i(B_i) \setminus (T \setminus B_i)$.
Indeed, let $A \in T_0$. Since $r(T^*) = T_0$ by condition 1, there is some $A' \in T^*$ with $r(A') = A$. So, by the construction of T^*, there is an $i \in \{1, \ldots, n\}$ such that $A' \in g_i(B_i) \setminus (T \setminus B_i) \subseteq T_i$. Condition 3 then yields $h_i(A') = r(A') = A$, and since $r(A') \cong A'$ by condition 2, we have $A' \cong A$.

Define $r \colon \text{dom}(\bigcup_{i=1}^{n} f_i(B_i)) \to \text{dom}(T_0)$ such that

$$r(u) = r_i(u) \quad \text{for all } i \in \{1, \ldots, n\} \text{ and } u \in \text{dom}(f_i(B_i)).$$

This is well-defined, since $\mathrm{nulls}(f_i(B_i)) \cap \mathrm{nulls}(f_j(B_j)) = \emptyset$ for all distinct $i, j \in \{1, \ldots, n\}$, and each r_i is the identity on constants. We claim that r satisfies conditions 1–3 above.

To see that r satisfies condition 2, let $A' \in T^*$. Then there is some $i \in \{1, \ldots, n\}$ with $A' \in g_i(B_i) \setminus (T \setminus B_i)$. By (6.15) and (6.17), we thus have $r(A') = r_i(A') \cong A'$.

To see that r satisfies condition 3, let $i \in \{1, \ldots, n\}$ and $A' \in g_i(B_i) \setminus (T \setminus B_i)$. Then, $r(A') = r_i(A') = h_i(A')$, where the last equality follows from the construction of h_i.

It thus remains to show that r satisfies condition 1, that is, $r(T^*) = T_0$. Note that, by (6.17), we have

$$T^* \subseteq \bigcup_{i=1}^{n} f_i(B_i).$$

Hence,

$$r(T^*) \subseteq r\left(\bigcup_{i=1}^{n} f_i(B_i)\right) = \bigcup_{i=1}^{n} r(f_i(B_i)) = \bigcup_{i=1}^{n} r_i(f_i(B_i)) \overset{(6.14)}{=} \bigcup_{i=1}^{n} f(B_i) = T_0.$$

To show that $r(T^*) = T_0$, we show that there is some $f^* \in val_C(T)$ with $f^*(T) = T^*$. Then, $f' := r \circ f^* \in val_C(T)$. Since $T_0 \in min_C(T)$ and $f'(T) = r(f^*(T)) = r(T^*) \subseteq T_0$, this implies $r(T^*) = T_0$, and the proof is complete.

Thus, it remains to show that there is a mapping $f^* \in val_C(T)$ with $f^*(T) = T^*$. Basically, f^* is obtained by repeated application of the mappings $g_1, \ldots, g_n$.

Let us first modify $g_1, \ldots, g_n$ as follows. Choose an arbitrary "renaming" of the nulls of T. That is, pick an injective mapping $\rho \colon \mathrm{dom}(T) \to \mathrm{const}(T) \cup (Null \setminus \mathrm{nulls}(T))$ such that $\rho(c) = c$ for each constant $c \in \mathrm{const}(T)$. Note that ρ maps each null of T to a unique null that does not occur in T. Let

$$\mathcal{X} := \rho(\mathrm{nulls}(T)).$$

For each $i \in \{1, \ldots, n\}$ then define $\hat{g}_i \colon \mathrm{dom}(T) \cup C \cup \mathcal{X} \to \mathrm{dom}(T) \cup C \cup \mathcal{X}$ such that for each $u \in \mathrm{dom}(T) \cup C \cup \mathcal{X}$,

$$\hat{g}_i(u) := \begin{cases} g_i(\rho^{-1}(u)), & \text{if } u \in \rho(\mathrm{nulls}(B_i)) \text{ and } g_i(\rho^{-1}(u)) \in \mathrm{nulls}(B_i) \\ \rho(g_i(\rho^{-1}(u))), & \text{if } u \in \rho(\mathrm{nulls}(B_i)) \text{ and } g_i(\rho^{-1}(u)) \notin \mathrm{nulls}(B_i) \\ u, & \text{otherwise.} \end{cases}$$

Note that for each $i \in \{1, \ldots, n\}$,

$$\hat{g}_i(\rho(B_i)) \setminus \rho(T) = g_i(B_i) \setminus (T \setminus B_i). \tag{6.18}$$

Now let

$$\hat{g} := \hat{g}_n \circ \cdots \circ \hat{g}_2 \circ \hat{g}_1.$$

Recall the graph G mentioned at the beginning of the proof, on page 36. Then an application of $\hat{g}$ to an atom $\rho(A)$ with $A \in B_{i_1}$ corresponds to following the maximal path in G that starts in A and proceeds to atoms $A' \in B_{i_2}, A'' \in B_{i_3}, \ldots$ with $i_1 < i_2 < i_3 < \cdots$. If $A''' \in B_j$ is the endpoint of this path, then either $g_j(A''') \notin T \setminus B_j$ and $\hat{g}(\rho(A)) = \hat{g}_j(\rho(A''')) = g_j(A''')$, or $g_j(A''') \in T \setminus B_j$ and $\hat{g}(\rho(A)) = \rho(A''')$.

For each $s \geq 0$ let

$$\hat{g}^s := \begin{cases} \rho, & \text{if } s = 0, \\ \hat{g} \circ \hat{g}^{s-1}, & \text{if } s \geq 1. \end{cases}$$

We show by induction that

$$\hat{g}^1(T) \supseteq \hat{g}^2(T) \supseteq \hat{g}^3(T) \supseteq \cdots . \tag{6.19}$$

To prove $\hat{g}^1(T) \supseteq \hat{g}^2(T)$, let $A \in \hat{g}^2(T)$. If $A \in T^*$, then by (6.18), we have $A \in \hat{g}^1(T)$. Otherwise, if $A \in \rho(T)$, there is an $A' \in \hat{g}^1(T)$ with $\hat{g}(A') = A$ and $A' \in \rho(T)$. Since $A' \in \rho(T)$, we have $A \in \hat{g}(\rho(T)) = \hat{g}^1(T)$, as desired. To prove $\hat{g}^{i+2}(T) \supseteq \hat{g}^{i+1}(T)$ for $i \geq 1$, let $A \in \hat{g}^{i+2}(T)$. Then there is an $A' \in \hat{g}^{i+1}(T)$ with $\hat{g}(A') = A$. Since $\hat{g}^{i+1}(T) \subseteq \hat{g}^i(T)$ by the induction hypothesis, we have $A \in \hat{g}(\hat{g}^i(T)) = \hat{g}^{i+1}(T)$, as desired.

By (6.19) and since $\hat{g}^1(T)$ is finite, there is an $s_0 \geq 1$ such that $\hat{g}^{s_0}(T) = \hat{g}^s(T)$ for each $s \geq s_0$. Let $f^* := \hat{g}^{s_0}$. We show that $f^*(T) = T^*$.

First observe that

$$T^* = \bigcup_{i=1}^{n} \big(g_i(B_i) \setminus (T \setminus B_i) \big) = \hat{g}^1(T) \setminus \rho(T) = f^*(T) \setminus \rho(T).$$

To see that T^* is not a proper subinstance of $f^*(T)$, we show that $f^*(T)$ contains no atoms from $\rho(T)$.

For a contradiction, suppose that $f^*(T)$ contains an atom $A \in \rho(T)$. Then, $A \in \rho(B_i)$ for some $i \in \{1, \ldots, n\}$. Since $\hat{g}(f^*(T)) = f^*(T)$, we know that $\hat{g}$ is a bijection on $\mathrm{dom}(f^*(T))$. Furthermore, since $\hat{g}$ is the identity on $\mathrm{dom}(T) \cup C$, we have $\hat{g}(\bot) \in \mathcal{X}$ for each $\bot \in \mathcal{X}$. It follows that

$$\hat{g}_i(A) \cong A, \quad \text{and} \quad \hat{g}_i(A) \in \rho(B_j) \quad \text{for some } j \in \{1, \ldots, n\} \setminus \{i\}.$$

Let $A' := \rho^{-1}(A)$. By the construction of $\hat{g}_i$, we have

$$g_i(A') \cong A', \quad \text{and} \quad g_i(A') \in B_j \quad \text{for some } j \in \{1, \ldots, n\} \setminus \{i\}.$$

Since B_i is packed and g_i maps each null in A' to a null in B_j, each atom in $g_i(B_i)$ contains a null from B_j. Together with $g_i \in val_C(T, B_i)$, this implies $g_i(B_i) \subseteq B_j$. In other words, g_i is a homomorphism from T to $T \setminus B_i$, which contradicts the fact that T is a core. Consequently, we must have $f^*(T) = T^*$. $\qquad\square$

Corollary 6.24. *Let T be an instance such that T is a core, and each atom block of T is packed. Let $C \subseteq Const$, and let A be an atom. Then the following statements are equivalent:*

- *There is an instance in $min_C(T)$ that contains an atom isomorphic to A.*
- *There is an atom block B of T such that some instance in $min_C(T, B)$ contains an atom isomorphic to A.*

The following polynomial time algorithm for deciding whether $R(\bar{t})$ occurs in some minimal instance in $poss(T)$ immediately suggests itself. Let C be the set of all constants in $\bar{t}$. Consider each atom block B of T, and each $T_0 \in min_C(T, B)$ in turn, and accept the input if and only if $R(\bar{t}) \in T_0$ for some T_0. By Proposition 6.18, the instances T_0 can be computed in polynomial time.

The following example shows that the proof of Lemma 6.23 fails if T contains atom blocks that are not packed.

Example 6.25. Let E be a binary relation symbol, and consider the instance T over $\{E\}$ with

$$E^T = \{(\bot_1, a), (\bot_1, b), (\bot_1, \bot_1'), (\bot_1', c), (\bot_2, a), (\bot_2, b), (\bot_2, \bot_2'), (c, \bot_2')\}.$$

Note that T is a core, and that T has the two atom blocks

$$B_1 \;=\; \{E(\bot_1, a), E(\bot_1, b), E(\bot_1, \bot_1'), E(\bot_1', c)\}, \text{ and}$$
$$B_2 \;=\; \{E(\bot_2, a), E(\bot_2, b), E(\bot_2, \bot_2'), E(c, \bot_2')\}.$$

See Figure 3 for a graph representation of T and its two atom blocks B_1 and B_2. Note also
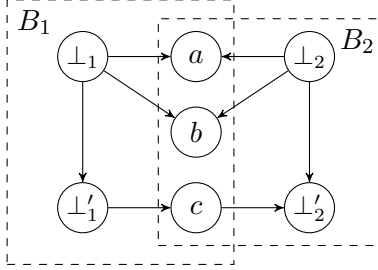

Figure 3: The instance T, and the two atom blocks B_1 and B_2 of T, which are the subin-
stances induced by the vertices in the corresponding dashed rectangles.

that neither B_1 nor B_2 is packed.

Consider $f \in val_\emptyset(T)$ with $f(\bot_1) = f(\bot_2) = a$ and $f(\bot_1') = f(\bot_2') = b$. Then it is not
hard to see that

$$f(T) \;=\; \{E(a, a), E(a, b), E(b, c), E(c, b)\} \in min_\emptyset(T).$$

Furthermore, for the mappings f_i created in the proof of Lemma 6.23, we have

$$f_1(T) \;=\; \{E(a, a), E(a, b), E(b, c), E(\bot_2, a), E(\bot_2, b), E(\bot_2, \bot_2'), E(c, \bot_2')\}$$

and

$$f_2(T) \;=\; \{E(a, a), E(a, b), E(c, b), E(\bot_1, a), E(\bot_1, b), E(\bot_1, \bot_1'), E(\bot_1', c)\}.$$

For $g_i \in val_\emptyset(T, B_i)$ with $g_i(\bot_i) = \bot_{3-i}$ and $g_i(\bot_i') = b$, it holds that $g_i \in minval_\emptyset(T, B_i)$,
and moreover,

$$g_1(T) \;=\; \{E(\bot_2, a), E(\bot_2, b), E(\bot_2, \bot_2'), E(c, \bot_2'), E(b, c)\} \in min_\emptyset(T),$$
$$g_2(T) \;=\; \{E(\bot_1, a), E(\bot_1, b), E(\bot_1, \bot_1'), E(\bot_1', c), E(c, b)\} \in min_\emptyset(T).$$

Note that $E(a, a)$ and $E(a, b)$ occur in $f_i(B_i)$, but neither $g_1(T)$ nor $g_2(T)$ contains $E(a, a)$
or $E(a, b)$.

6.3.3. *Proof of Theorem 6.8.* This section finally proves Theorem 6.8. Let $M = (\sigma, \tau, \Sigma)$ be
a schema mapping, where Σ consists of packed st-tgds, and let $q(\bar{x})$ be a universal query
over τ. We show that there is a polynomial time algorithm that, given as input an instance
$T := \text{Core}(M, S)$ for some source instance S for M, and a tuple $\bar{t} \in Const^{|\bar{x}|}$, decides whether
$\bar{t} \in cert_{\text{GCWA}^*}(q, M, S)$.

As shown in Section 6.3.1, we can assume that $\bar{t}$ is a tuple over $\text{const}(T) \cup \text{dom}(q)$, and
that in this case we have $\bar{t} \notin cert_{\text{GCWA}^*}(q, M, S)$ if and only if there is a nonempty finite set

$\mathcal{T}$ of minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Now observe that $\neg q$ is logically equivalent to a query $\bar{q}$ of the form

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

where each q_i is an existential query of the form

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

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

Fix some constant bs as in Lemma 6.14. Then, for each atom block B of $\mathrm{Core}(M, S)$, we have $|\mathrm{nulls}(B)| \leq bs$. Furthermore, Proposition 6.22 tells us that each atom block of $\mathrm{Core}(M, S)$ is packed. We can now use the following algorithm to decide, given as input an instance $T := \mathrm{Core}(M, S)$ for some source instance S for M, and a tuple $\bar{t} \in Const^{|\bar{x}|}$, whether $\bar{t} \in cert_{\mathrm{GCWA}^*}(q, M, S)$:

(1) Determine the atom blocks of T and check whether each atom block B of T is packed and satisfies $|\mathrm{nulls}(B)| \leq bs$; if not, reject the input.
(2) Check whether T is a core; if not, reject the input.
(3) For each $i \in \{1, \ldots, m\}$:
  (a) Check whether there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models q_i(\bar{t})$.
  (b) If such a $\mathcal{T}$ exists, reject the input.
(4) Accept the input.

Step 1 clearly runs in polynomial time, and step 2 can be implemented in polynomial time using the algorithm from Lemma 6.16 (that algorithm outputs T if and only if T is a core). Lemma 6.26 below tells us that step 3a can be implemented in polynomial time as well. Thus, once Lemma 6.26 is proved, the proof of Theorem 6.8 is complete.

Lemma 6.26. *Let $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ be a FO query over τ, where $\varphi = \bigwedge_{i=1}^{p} \varphi_i$, and each φ_i is an atomic FO formula or the negation of an atomic FO formula. For each positive integer bs, there is a polynomial time algorithm that decides:*

$\mathrm{CoreEval}_{\tau, bs}$

Input: an instance T over τ such that T is a core and each atom block of T is packed and contains at most bs nulls; and a tuple $\bar{t} \in Const^{|\bar{x}|}$

Question: Is there a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models q(\bar{t})$?

The remaining part of this section is devoted to a proof of Lemma 6.26.

Let $q(\bar{x}) = \exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ be as in the hypothesis of Lemma 6.26. Without loss of generality, there is no variable that occurs both in $\bar{x}$ and in $\bar{y}$, and φ has the form $\bigwedge_{i=1}^{p} \varphi_i$, where each

φ_i is a relational atomic FO formula, the negation of a relational atomic FO formula, or the negation of an equality. Let bs be a positive integer.

Suppose we are given an instance T over τ, where T is a core, each atom block of T is packed, and each atom block of T contains at most bs nulls, and a tuple $\bar{t} \in Const^{|\bar{x}|}$. In a first step, we rewrite φ to a formula ψ by replacing each variable x in $\bar{x}$ with the corresponding constant assigned to x by $\bar{t}$. That is, if $\bar{x} = (x_1, \ldots, x_k)$ and $\bar{t} = (t_1, \ldots, t_k)$, then ψ is obtained from φ by replacing, for each $i \in \{1, \ldots, k\}$, each occurrence of the variable x_i in φ by t_i. Let

$$\tilde{q} := \exists \bar{y} \, \psi(\bar{y}).$$

To check whether there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models q(\bar{t})$, it suffices to check whether there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models \tilde{q}$.

Suppose that ψ has the form

$$\psi(\bar{y}) = \bigwedge_{i=1}^{k} R_i(\bar{x}_i) \wedge \bigwedge_{i=1}^{l} \neg Q_i(\bar{w}_i) \wedge \bigwedge_{i=1}^{m} \neg v_i = v_i'.$$

Let C be the set of constants that occur in ψ, and for each $i \in \{1, \ldots, k\}$, let X_i be the set of all variables in $\bar{x}_i$. Given an assignment α for a set X of variables, and a tuple $\bar{t}$ over $X \cup Const$, we sloppily write $\alpha(\bar{t})$ for the tuple obtained from $\bar{t}$ by replacing each occurrence of each variable $x \in X$ in $\bar{t}$ with $\alpha(x)$.

The idea for finding a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ with $\bigcup \mathcal{T} \models \tilde{q}$ is as follows. In the first step, we compute, for each $i \in \{1, \ldots, k\}$, the set of all pairs (T_i, α_i) such that $T_i \in min_C(T, B)$ for some atom block B of T, and $\alpha_i(\bar{x}_i) \in R_i^{T_i}$. Thus, modulo renaming of values that do not occur in $\mathrm{const}(T) \cup C$, we enumerate the possible assignments α_i of $\bar{x}_i$ under which $R_i(\bar{x}_i)$ is satisfied in some minimal instance in $poss(T)$; the instance T_i can then be considered as a witness to this fact. In the second step, we try to join the pairs $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$, where each (T_i, α_i) is a pair computed for i in the first step, to a single pair $(\tilde{T}, \tilde{\alpha})$ such that $\tilde{T}$ satisfies each $R_i(\bar{x}_i)$ under the assignment $\tilde{\alpha}$. The instance $\tilde{T}$ will actually be the union of isomorphic copies $\rho_1(T_1), \ldots, \rho_k(T_k)$ of the instances $T_1, \ldots, T_k$. In particular, the set $\{\rho_1(T_1), \ldots, \rho_k(T_k)\}$ is already close to the desired set $\mathcal{T}$: it is a finite set of instances from $min_C(T)$, and its union satisfies the subformula $\bigwedge_{i=1}^{k} R_i(\bar{x}_i)$ of $\tilde{q}$ under $\tilde{\alpha}$. Not all pairs $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ can be joined together. We will join only pairs that are *compatible* in the sense of Definition 6.27 below. By taking care in how those pairs are joined together, and using Lemma 6.23, we can show that the desired set $\mathcal{T}$ exists if and only if the instance obtained from $\tilde{T}$ by adding a large enough, but constant, number of isomorphic copies of T to $\tilde{T}$ satisfies $\tilde{q}$.

More precisely, the algorithm proceeds as follows. Fix the constant

$$s := k + \sum_{i=1}^{l} |\bar{w}_i| + 2 \cdot m.$$

In the above description, $s - k$ is the number of isomorphic copies of T that will be added to $\tilde{T}$. For each $i \in \{1, \ldots, s\}$, pick an injective mapping

$$\rho_i \colon \mathrm{dom}(T) \cup C \to Dom$$

such that $\rho_i(c) = c$ for each $c \in \mathrm{const}(T) \cup C$, $\rho_i(\bot) \in Null$ for each $\bot \in \mathrm{nulls}(T)$, and such that for all distinct $i, j \in \{1, \ldots, s\}$, we have $\mathrm{nulls}(\rho_i(T)) \cap \mathrm{nulls}(\rho_j(T)) = \emptyset$. Then compute,

for each $i \in \{1, \ldots, k\}$, the set

$$\mathcal{X}_i := \Big\{ (T_0, \alpha) \mid \text{there is some } T_0' \in min_C(T, B) \text{ and an atom block } B \text{ of } T$$
$$\text{such that } T_0 = \rho_i(T_0'), \, \alpha \colon X_i \to \mathrm{dom}(T_0), \text{ and } \alpha(\bar{x}_i) \in R_i^{T_0} \Big\}. \tag{6.20}$$

Now we would like to join pairs $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ into single pairs $(\tilde{T}, \tilde{\alpha})$ according to the above description. However, we would like to do this only if the pairs $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are *compatible* in the following sense. Intuitively, $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible if the nulls in the image of each α_i can be consistently renamed such that the resulting mappings $\tilde{\alpha}_i$ agree on common variables.

Definition 6.27 (compatible)**.** We say that $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ are *compatible* if there is an equivalence relation $\sim$ on $D := \bigcup_{i=1}^k \alpha_i(X_i)$ such that

(1) for all $i, j \in \{1, \ldots, k\}$ and $x \in X_i \cap X_j$, we have $\alpha_i(x) \sim \alpha_j(x)$,
(2) for all $u, u' \in D$, if $u \sim u'$ and $u \in \mathit{Const}$, then $u = u'$, and
(3) for all $i \in \{1, \ldots, k\}$ and $x, x' \in X_i$, we have $\alpha_i(x) \sim \alpha_i(x')$ if and only if $\alpha_i(x) = \alpha_i(x')$.

Proposition 6.28. *There is an algorithm that, given $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ as input, decides in time linear in the size of T whether $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible, and if so, outputs an equivalence relation $\sim$ on $D := \bigcup_{i=1}^k \alpha_i(X_i)$ that satisfies conditions 1–3 of Definition 6.27. In fact, $\sim$ is the smallest such equivalence relation (with respect to set inclusion).*

Proof. Given $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$, the following algorithm computes the desired relation $\sim$ if it exists:

(1) Initialize $\sim$ to be $\{(u, u) \mid u \in \alpha_i(X_i) \text{ for some } i \in \{1, \ldots, k\}\}$.
(2) For all $i, j \in \{1, \ldots, k\}$ and $x \in X_i \cap X_j$, add $(\alpha_i(x), \alpha_j(x))$ to $\sim$.
(3) For all $i \in \{1, \ldots, k\}$ and $x, x' \in X_i$ with $\alpha_i(x) = \alpha_i(x')$, add $(\alpha_i(x), \alpha_i(x'))$ to $\sim$.
(4) Update $\sim$ to be the symmetric and transitive closure of $\sim$.
(5) If $\sim$ satisfies conditions 2 and 3 of Definition 6.27, then output $\sim$; otherwise output "not compatible".

Since k and $X_1, \ldots, X_k$ are constant, it should be clear that each of the steps 1–5 can be accomplished in constant time, after building the necessary data structures from the input in time linear in the size of T (note that each T_i is at most as large as T, so that the length of the input is linear in the size of T).

It is now not hard to see that if the algorithm outputs a relation $\sim$, then $\sim$ is an equivalence relation on $D := \bigcup_{i=1}^k \alpha_i(X_i)$ that satisfies conditions 1–3 of Definition 6.27. In particular, $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible. Even more, $\sim$ is the smallest such equivalence relation, since every equivalence relation $\sim^*$ on D that satisfies conditions 1–3 of Definition 6.27 must contain the pairs put into $\sim$ in steps 1–4 of the algorithm. The same argument shows that the algorithm outputs a relation $\sim$ if there is an equivalence relation $\sim^*$ on D that satisfies conditions 1–3 of Definition 6.27, that is, if $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible. $\qquad\square$

We now define the join of compatible pairs $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$. Given an equivalence relation $\sim$ on $D := \bigcup_{i=1}^k \alpha_i(X_i)$ as in Definition 6.27, the idea of the join is to identify values $u, u' \in D$ with $u \sim u'$, and to "glue" the resulting instances $\tilde{T}_i$ and assignments $\tilde{\alpha}_i$ together to a single instance $\tilde{T}$ and assignment $\tilde{\alpha}$.

Definition 6.29 (Join). Let $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ be compatible, and let $\sim$ be the smallest equivalence relation on $D := \bigcup_{i=1}^k \alpha_i(X_i)$ that satisfies conditions 1–3 of Definition 6.27. Pick some linear order $\preceq$ on the elements of D, and for each $u \in D$, let $\hat{u}$ be the minimal element in $[u] := \{u' \in D \mid u' \sim u\}$ with respect to $\preceq$. For each $i \in \{1, \ldots, k\}$, define $r_i \colon \mathrm{dom}(T_i) \to Dom$ such that for each $u \in \mathrm{dom}(T_i)$,

$$r_i(u) := \begin{cases} \hat{u}, & \text{if } u \in \alpha_i(X_i), \\ u, & \text{otherwise.} \end{cases}$$

Then *the join of* $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ is the pair $(\tilde{T}, \tilde{\alpha})$, where $\tilde{T} := \bigcup_{i=1}^k r_i(T_i)$, and $\tilde{\alpha} \colon \bigcup_{i=1}^k X_i \to Dom$ is such that for each $x \in \bigcup_{i=1}^k X_i$,

$$\tilde{\alpha}(x) := \begin{cases} r_1(\alpha_1(x)), & \text{if } x \in X_1 \\ \quad \vdots \\ r_k(\alpha_k(x)), & \text{if } x \in X_k. \end{cases}$$

Note that different choices of $\preceq$ yield different joins. For definiteness, we can generate $\preceq$ as follows. Initialize $\preceq$ to be the empty relation. For increasing $i = 1, 2, \ldots, k$, consider the variables $x \in X_i$ in some predefined fixed order, and if $u := \alpha_i(x)$ does not already occur in $\preceq$, add u as the new maximal element to $\preceq$. This takes constant time, since k and $X_1, \ldots, X_k$ are fixed. For the following construction, it is not important that the join always yields the same result – the join resulting from any linear ordering $\preceq$ on D is fine. What is important are the properties summarized in Proposition 6.30 below. Note also that modulo the choice of $\preceq$, $\tilde{\alpha}$ is well-defined by the construction of $\sim$ and $r_1, \ldots, r_k$: if $x \in X_i \cap X_j$, then $\alpha_i(x) \sim \alpha_j(x)$, and thus, $r_i(\alpha_i(x)) = r_j(\alpha_j(x))$.

Proposition 6.30. *The join* $(\tilde{T}, \tilde{\alpha})$ *of compatible pairs* $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ *can be computed in time linear in the size of* T *and has the following properties. Let* $r_1, \ldots, r_k$ *be the mappings used in the construction of* $(\tilde{T}, \tilde{\alpha})$. *Then for all* $i, j \in \{1, \ldots, k\}$:

(1) *For all* $c \in \mathrm{const}(T_i)$ *and* $\bot \in \mathrm{nulls}(T_i)$, *we have* $r_i(c) = c$ *and* $r_i(\bot) \in Null$.
(2) *Let* $\sim$ *be an equivalence relation on* $D := \bigcup_{i=1}^k \alpha_i(X_i)$ *that satisfies conditions 1–3 of Definition 6.27. Then for all* $u \in \mathrm{dom}(T_i)$ *and* $u' \in \mathrm{dom}(T_j)$,

$$r_i(u) = r_j(u') \implies u = u' \text{ or: } u \in \alpha_i(X_i), \ u' \in \alpha_j(X_j) \text{ and } u \sim u'.$$

Furthermore, if $\sim$ *is the smallest such equivalence relation, then*

$$r_i(u) = r_j(u') \iff u = u' \text{ or: } u \in \alpha_i(X_i), \ u' \in \alpha_j(X_j) \text{ and } u \sim u'.$$

(3) r_i *is injective.*
(4) $\tilde{\alpha}(\bar{x}_i) \in R_i^{\tilde{T}}$.

Proof. Let us first see that $(\tilde{T}, \tilde{\alpha})$ can be computed in time $O(n)$, where n is the size of T, given $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ as input. By Proposition 6.28, the relation $\sim$ can be computed in time $O(n)$. Since k and $X_1, \ldots, X_k$ are fixed, the linear order $\preceq$ can be computed in constant time. Furthermore, since k is constant and all T_i have size at most n, the mappings r_i and the join $(\tilde{T}, \tilde{\alpha})$ can be computed in time $O(n)$. We next prove 1–4.

Ad 1: Let $c \in \mathrm{const}(T_i)$. If $c \notin \alpha_i(X_i)$, then by the construction of r_i we have $r_i(c) = c$. Otherwise, if $c \in \alpha_i(X_i)$, there is some $x \in X_i$ with $\alpha_i(x) = c$, so that by the construction of r_i we have $c = \alpha_i(x) \sim r_i(\alpha_i(x)) = r_i(c)$. Condition 2 of Definition 6.27 then yields

$r_i(c) = c$. Next let $\perp \in \text{nulls}(T_i)$. As above, if $\perp \notin \alpha_i(X_i)$, then $r_i(\perp) = \perp \in Null$. Otherwise, $\perp \sim r_i(\perp)$, so that by condition 2 of Definition 6.27, $r_i(\perp) \in Null$.

Ad 2: Let $\sim$ be an equivalence relation on D that satisfies conditions 1–3 of Definition 6.27. We first prove the statement for the case that $\sim$ is the smallest such relation. That is, we have to show

$$r_i(u) = r_j(u') \iff u = u' \text{ or: } u \in \alpha_i(X_i), u' \in \alpha_j(X_j) \text{ and } u \sim u'. \qquad (6.21)$$

We first prove the direction from right to left. Suppose that $u = u'$. If $i = j$, we have $r_i(u) = r_j(u')$. If $i \neq j$, then $\text{nulls}(T_i) \cap \text{nulls}(T_j) = \emptyset$ implies that u and u' are constants, and therefore $r_i(u) = u = u' = r_j(u')$ by 1. Suppose next that $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim u'$. Pick $x \in X_i$ and $x' \in X_j$ such that $\alpha_i(x) = u$ and $\alpha_j(x') = u'$. Then, $\alpha_i(x) \sim \alpha_j(x')$, and the construction of r_i and r_j immediately implies $r_i(u) = r_j(u')$.

We next prove the direction from left to right. Let $r_i(u) = r_j(u')$. We distinguish the following cases:

(a) $u \notin \alpha_i(X_i) \cap Null$ and $u' \notin \alpha_j(X_j) \cap Null$.
(b) $u \in \alpha_i(X_i) \cap Null$ or $u' \in \alpha_j(X_j) \cap Null$.

In case (a), by the construction of r_i, r_j and by 1, we have $r_i(u) = u$ and $r_j(u') = u'$. Since $r_i(u) = r_j(u')$, this implies $u = u'$.

So assume case (b). By symmetry it suffices to deal with the case that $u \in \alpha_i(X_i) \cap Null$. By the construction of r_i, we then have

$$u \sim r_i(u) = r_j(u').$$

We claim that $u' \in \alpha_j(X_j)$. Suppose, to the contrary, that $u' \notin \alpha_j(X_j)$. By the construction of r_j, we have

$$u' = r_j(u') = r_i(u) \sim u.$$

Note that $u \in \alpha_i(X_i)$, $r_i(u) = u'$ and the construction of r_i imply that $u' \in D$. Pick $p \in \{1, \ldots, k\}$ and $x \in X_p$ with $\alpha_p(x) = u'$. By $u \in Null$, $r_i(u) = u'$ and 1, we have $u' \in Null$. Moreover, since $u' \in \text{nulls}(T_j)$, $u' = \alpha_p(X_p) \in \text{nulls}(T_p)$, and $\text{nulls}(T_j) \cap \text{nulls}(T_p) = \emptyset$ for $j \neq p$, we have $p = j$. This, however, implies that $u' \in \alpha_j(X_j)$, which is a contradiction to our assumption that $u' \notin \alpha_j(X_j)$. Hence, $u' \in \alpha_j(X_j)$. By the construction of r_j, we have $u' \sim r_j(u') \sim u$. In particular, $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim u'$, as desired.

Finally, let $\sim^*$ be another equivalence relation on D that satisfies conditions 1–3 of Definition 6.27. We show that

$$r_i(u) = r_j(u') \implies u = u' \text{ or: } u \in \alpha_i(X_i), u' \in \alpha_j(X_j) \text{ and } u \sim^* u'.$$

Let $r_i(u) = r_j(u')$. By (6.21), we have $u = u'$, or: $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim u'$. By minimality of $\sim$, $u \sim u'$ implies $u \sim^* u'$, so that $u = u'$, or: $u \in \alpha_i(X_i)$, $u' \in \alpha_j(X_j)$ and $u \sim^* u'$, as desired.

Ad 3: Let $u, u' \in \text{dom}(T_i)$ be such that $r_i(u) = r_i(u')$. We have to show that $u = u'$. By 2, we have $u = u'$, or: $u, u' \in \alpha_i(X_i)$ and $u \sim u'$. If $u = u'$, we are done. So assume that $u, u' \in \alpha_i(X_i)$ and $u \sim u'$. Let $x, x' \in X_i$ be such that $\alpha_i(x) = u$ and $\alpha_i(x') = u'$. Then $\alpha_i(x) \sim \alpha_i(x')$, and by condition 3 of Definition 6.27, we have $u = \alpha_i(x) = \alpha_i(x') = u'$, as desired.

Ad 4: This follows immediately from the construction of $\tilde{T}$, $\tilde{\alpha}$, and 1. $\qquad \square$

We can now give the algorithm for $\text{CoreEval}_{\tau,bs}$:

Algorithm 6.31 (Main algorithm)**.**
Input: an instance T over τ that is a core and each atom block of T is packed and contains
                at most bs nulls; a tuple $\bar{t} \in Const^{|\bar{x}|}$
Output: "yes" if there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ such that
                $\bigcup \mathcal{T} \models q(\bar{t})$; otherwise "no"
(1) Compute $\tilde{q} = \exists \bar{y}\, \psi(\bar{y})$ and choose $\rho_1, \ldots, \rho_s$. (Recall that each ρ_i is an injective mapping
     from $\text{dom}(T) \cup C$ to Dom that is the identity on constants, maps nulls to nulls, and that
     $\text{nulls}(\rho_i(T)) \cap \text{nulls}(\rho_j(T)) = \emptyset$ for distinct i, j.)
(2) Compute the sets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ according to (6.20).
(3) For all $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$:
     (a) Check whether $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible;
          if not, continue with next $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$.
     (b) Let $(\tilde{T}, \tilde{\alpha})$ be the join of $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$.
     (c) If $\tilde{T} \cup \bigcup_{i=k+1}^{s} \rho_i(T)$ satisfies $\tilde{q}$, output "yes".
(4) Output "no".

Let us now show that the algorithm decides $\text{CoreEval}_{\tau,bs}$ in polynomial time. For a more
precise upper bound on the algorithm's running time, see [21, Lemma 5.40].

Lemma 6.32. *Algorithm 6.31 runs in time polynomial in the size of T. Furthermore, the
following two statements are equivalent:*
(1) *There is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models \tilde{q}$.*
(2) *Algorithm 6.31 outputs "yes" on input T and $\bar{t}$.*

Proof. It is not hard to see that the algorithm runs in time polynomial in the size of T.
Indeed, the transformation from q to $\tilde{q}$ can be accomplished in constant time (since q is
fixed), and the mappings $\rho_1, \ldots, \rho_s$ can be generated in polynomial time. It is also not hard
to compute the sets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ in polynomial time: All we need to do in order to compute
$\mathcal{X}_i$ for $i \in \{1, \ldots, k\}$ is to iterate through all $T'_0 \in min_C(T, B)$, where B is an atom block
of T, and all assignments $\alpha \colon X_i \to \text{dom}(\rho(T'_0))$, and to check whether $R_i(\alpha(\bar{x}_i)) \in \rho(T'_0)$.
By Proposition 6.18, and since X_i is fixed, this can be done in polynomial time. Since k is
constant, all the sets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ can thus be computed in polynomial time. In particular,
since each of these sets has polynomial size, there are at most a polynomial number of
iterations of the algorithm's main loop. Propositions 6.28 and 6.30 imply that steps 3(a)
and 3(b) of the main loop run in polynomial time. Finally, step 3(c) clearly takes only a
polynomial number of steps. Altogether, the algorithm runs in polynomial time.
     It remains to show that the two statements 1 and 2 are equivalent.

$2 \Longrightarrow 1$: Assume that Algorithm 6.31 outputs "yes" on input T and $\bar{t}$. Then there are
compatible $(T_1, \alpha_1) \in \mathcal{X}_1, \ldots, (T_k, \alpha_k) \in \mathcal{X}_k$ such that the join $(\tilde{T}, \tilde{\alpha})$ of $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$
has the following property: the instance $T^* := \tilde{T} \cup \bigcup_{i=k+1}^{s} \rho_i(T)$ satisfies $\tilde{q}$. We construct a
nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ with $\bigcup \mathcal{T} \models \tilde{q}$.

     By Proposition 6.30, we have $\tilde{T} = \bigcup_{i=1}^{k} r_i(T_i)$, where each r_i is an injective mapping
from $\text{dom}(T_i)$ to Dom with $r_i(c) = c$ for each $c \in \text{const}(T_i)$, and $r_i(\bot) \in Null$ for each
$\bot \in \text{nulls}(T_i)$. For each $i \in \{k+1, \ldots, s\}$, let $T_i := \rho_i(T)$, and let r_i be the identity

mapping on $\mathrm{dom}(T_i)$. Then,

$$T^* = \bigcup_{i=1}^{s} r_i(T_i). \tag{6.22}$$

Note that each T_i is isomorphic to an instance $\hat{T}_i \in min_C(T)$. For $i \in \{1, \ldots, k\}$, this follows from Lemma 6.19 and the fact that T_i is isomorphic to an instance in $min_C(T, B)$ for some atom block B of T. For $i \in \{k+1, \ldots, s\}$, this follows from the fact that $T_i = \rho_i(T)$, that T is a core, and Proposition 6.11(2). For each $i \in \{1, \ldots, s\}$, let f_i be an isomorphism from $\hat{T}_i$ to T_i.

Let $v\colon \mathrm{dom}(T^*) \to \mathrm{const}(T^*) \cup (Const \setminus C)$ be an injective valuation of T^*, and for every $i \in \{1, \ldots, s\}$ let

$$v_i := v \circ r_i \circ f_i.$$

Note that v_i is an injective valuation of $\hat{T}_i$. To see this, note that f_i is an injective mapping from $\mathrm{dom}(\hat{T}_i)$ to $\mathrm{dom}(T_i)$ that is legal for $\hat{T}_i$, that r_i is an injective mapping from $\mathrm{dom}(T_i)$ to $\mathrm{dom}(T^*)$ that is legal for T_i, and that v is an injective valuation of T^*. Furthermore, for each $\bot \in \mathrm{nulls}(\hat{T}_i)$ we have $v_i(\bot) \notin C$, since both f_i and r_i map nulls to nulls, and v maps nulls to constants in $Const \setminus C$. In summary, $\hat{T}_i \in min_C(T)$, v_i is an injective valuation of $\hat{T}_i$, and $v_i^{-1}(c) = c$ for all $c \in \mathrm{dom}(v_i(\hat{T}_i)) \cap C$. Together with Proposition 6.11(1), this implies that $v_i(\hat{T}_i)$ is a minimal instance in $poss(T)$.

So,

$$\mathcal{T} := \{v_i(\hat{T}_i) \mid 1 \le i \le s\}$$

is a finite nonempty set of minimal instances in $poss(T)$, and

$$\bigcup \mathcal{T} = \bigcup_{i=1}^{s} v_i(\hat{T}_i) = \bigcup_{i=1}^{s} v(r_i(T_i)) = v\left(\bigcup_{i=1}^{s} r_i(T_i)\right) \overset{(6.22)}{=} v(T^*).$$

Since $T^* \models \tilde{q}$, v is injective, and v maps nulls in T^* to constants that do not occur in $\tilde{q}$, we conclude that $\bigcup \mathcal{T} \models \tilde{q}$.

$1 \Longrightarrow 2$: Assume that there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(T)$ such that $\bigcup \mathcal{T} \models \tilde{q}$. We show that Algorithm 6.31 outputs "yes" on input T and $\bar{t}$.

Since $\bigcup \mathcal{T} \models \tilde{q}$, there is an assignment $\beta\colon \bar{y} \to \mathrm{dom}(\bigcup \mathcal{T}) \cup C$ with

$$\bigcup \mathcal{T} \models \psi(\beta).$$

In particular, we can pick for each $i \in \{1, \ldots, k\}$ an instance $\hat{T}_i \in \mathcal{T}$ such that

$$\beta(\bar{x}_i) \in R_i^{\hat{T}_i}.$$

Note that there are at most $s - k$ values in $\beta(\bar{y}) \setminus C$ that do not occur in $\beta(\bar{x}_i)$ for some $i \in \{1, \ldots, k\}$. Thus, we can fix instances $\hat{T}_{k+1}, \ldots, \hat{T}_s \in \mathcal{T}$ such that each of the values in $\beta(\bar{y}) \setminus C$ that does not occur in $\beta(\bar{x}_i)$ for some $i \in \{1, \ldots, k\}$ belongs to $\mathrm{dom}(\hat{T}_j)$ for some $j \in \{k+1, \ldots, s\}$. Now β is an assignment for ψ with range in $\mathrm{dom}(\hat{T}_1 \cup \cdots \cup \hat{T}_s) \cup C$, and we have:

$$\bigcup_{i=1}^{s} \hat{T}_i \models \psi(\beta). \tag{6.23}$$

Let $i \in \{1, \ldots, k\}$. By Proposition 6.11(1), there is an instance $\tilde{T}_i \in min_C(T)$ and an injective valuation v_i of $\tilde{T}_i$ such that $v_i(\tilde{T}_i) = \hat{T}_i$, and $v_i^{-1}(c) = c$ for all $c \in \text{dom}(\hat{T}_i) \cap C$. In particular,

$$A_i := R_i(v_i^{-1}(\beta(\bar{x}_i))) \in \tilde{T}_i.$$

By Lemma 6.23, there is an atom block B_i of T, an instance $T_i' \in min_C(T, B_i)$, an atom $A_i'' \in T_i'$ with $A_i'' \cong A_i$, and a homomorphism h_i' from T_i' to $\tilde{T}_i$ such that $h_i'(T_i') = \tilde{T}_i$ and $h_i'(A_i'') = A_i$. In particular, $h_i := h_i' \circ \rho_i^{-1}$ is a homomorphism from $T_i := \rho_i(T_i')$ to $\tilde{T}_i$ with

$$h_i(T_i) = \tilde{T}_i \qquad \text{and} \qquad h_i(A_i') = A_i,$$

where $A_i' := \rho_i(A_i'') \cong A_i$. Let α_i be an assignment for X_i such that

$$A_i' = R_i(\alpha_i(\bar{x}_i)).$$

Note that $(T_i, \alpha_i) \in \mathcal{X}_i$.

In the following, we show that $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible, and if $(\tilde{T}, \tilde{\alpha})$ is the join of these pairs, then $\tilde{T} \cup \bigcup_{i=k+1}^{s} \rho_i(T)$ satisfies $\tilde{q}$. In particular, Algorithm 6.31 outputs "yes" on input T and $\bar{t}$.

The following properties of the assignments α_i are crucial for showing this:

Claim 1. Let $i, j \in \{1, \ldots, k\}$, $x, x' \in X_i$ and $x'' \in X_j$. Then,
(1) $v_i(h_i(\alpha_i(\bar{x}_i))) = \beta(\bar{x}_i)$. In particular, $v_i(h_i(\alpha_i(x))) = \beta(x)$.
(2) If $\beta(x) \in \text{const}(T) \cup C$, then $\alpha_i(x) = \beta(x)$.
(3) $\alpha_i(x) = \alpha_i(x')$ if and only if $\beta(x) = \beta(x')$.
(4) $\alpha_i(x) = \alpha_j(x'')$ implies $\beta(x) = \beta(x'')$.

Proof. Ad 1: Recall that $h_i(A_i') = A_i \in \tilde{T}_i$, and that v_i is injective on $\text{dom}(\tilde{T}_i)$. In particular, we have $h_i(\alpha_i(\bar{x}_i)) = v_i^{-1}(\beta(\bar{x}_i))$. Applying v_i to both sides yields $v_i(h_i(\alpha_i(\bar{x}_i))) = \beta(\bar{x}_i)$.

Ad 2: Let $\beta(x) \in \text{const}(T) \cup C$. By 1, we have

$$v_i(h_i(\alpha_i(x))) = \beta(x), \tag{6.24}$$

which implies

$$h_i(\alpha_i(x)) = \beta(x). \tag{6.25}$$

Indeed, if $\beta(x) \in \text{const}(T)$, (6.25) follows immediately from (6.24), $\text{const}(T) \subseteq \text{const}(\tilde{T}_i)$, and the fact that v_i is an injective mapping from $\text{dom}(\tilde{T}_i)$ that is the identity on constants. On the other hand, if $\beta(x) \in C$, then (6.25) follows immediately from (6.24), $\beta(x) \in \text{dom}(\hat{T}_i)$, and the fact that $v_i^{-1}(c) = c$ for all $c \in \text{dom}(\hat{T}_i) \cap C$.

Now (6.25) and $h_i(A_i') \cong A_i'$ imply that $\alpha_i(x)$ is a constant, and since h_i is the identity on constants, we have $\alpha_i(x) = \beta(x)$.

Ad 3: By 1, we have $v_i(h_i(\alpha_i(\bar{x}_i))) = \beta(\bar{x}_i)$. Recall also that v_i is injective, and that $h_i(A_i') = A_i \cong A_i'$, which implies that h_i is injective on $\alpha_i(X_i)$. Altogether, $f_i := v_i \circ h_i$ is a bijection from $\alpha_i(X_i)$ to $\beta(X_i)$. This implies that $\alpha_i(x) = \alpha_i(x')$ if and only if $\beta(x) = \beta(x')$.

Ad 4: Let $\alpha_i(x) = \alpha_j(x'')$. If $i = j$, then $\beta(x) = \beta(x'')$ follows immediately from 3. So assume that $i \neq j$. Since $\alpha_i(x) \in \text{dom}(T_i)$, $\alpha_j(x'') \in \text{dom}(T_j)$ and $\text{nulls}(T_i) \cap \text{nulls}(T_j) = \emptyset$, $\alpha_i(x)$ and $\alpha_j(x'')$ must be constants. By 1 and the fact that the homomorphisms h_i, h_j as well as the valuations v_i, v_j are the identity on constants, we conclude that $\beta(x) = \alpha_i(x) = \alpha_j(x'') = \beta(x'')$. $\lrcorner$

We now show that $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible. To this end, we consider the relation

$$\sim := \big\{ (\alpha_i(x), \alpha_j(x')) \mid i, j \in \{1, \ldots, k\},\ x \in X_i,\ x' \in X_j,\ \beta(x) = \beta(x') \big\}$$

on $D := \bigcup_{i=1}^k \alpha_i(X_i)$.

Claim 2.

(1) For all $i, j \in \{1, \ldots, k\}$, $x \in X_i$ and $x' \in X_j$, we have

$$\alpha_i(x) \sim \alpha_j(x') \iff \beta(x) = \beta(x').$$

(2) The relation $\sim$ is an equivalence relation on D that satisfies conditions 1–3 of Definition 6.27.

Proof. Ad 1: Let $i, j \in \{1, \ldots, k\}$, $x \in X_i$ and $x' \in X_j$. If $\beta(x) = \beta(x')$, then the definition of $\sim$ immediately yields $\alpha_i(x) \sim \alpha_j(x')$.

On the other hand, let $\alpha_i(x) \sim \alpha_j(x')$. Then there are $i', j' \in \{1, \ldots, k\}$, $y \in X_{i'}$ and $y' \in X_{j'}$ such that

$$\alpha_{i'}(y) = \alpha_i(x) \quad \text{and} \quad \alpha_{j'}(y') = \alpha_j(x'), \tag{6.26}$$

and

$$\beta(y) = \beta(y'). \tag{6.27}$$

By (6.26) and Claim 1(4), we have $\beta(y) = \beta(x)$ and $\beta(y') = \beta(x')$, which by (6.27) yields $\beta(x) = \beta(x')$, as desired.

Ad 2: It is easy to verify that $\sim$ is an equivalence relation on D. Reflexivity and symmetry are clear, and transitivity is easy to show using 1.

It follows easily from 1 that $\sim$ satisfies condition 1 of Definition 6.27: Let $i, j \in \{1, \ldots, k\}$ and $x \in X_i \cap X_j$. Since $\beta(x) = \beta(x)$, 1 yields $\alpha_i(x) \sim \alpha_j(x)$.

For proving that $\sim$ satisfies condition 2 of Definition 6.27, let $u, u' \in D$ be such that $u \sim u'$ and $u \in Const$. Since $u \sim u'$, there are $i, j \in \{1, \ldots, k\}$, $x \in X_i$ and $x' \in X_j$ such that $\alpha_i(x) = u$, $\alpha_j(x') = u'$, and

$$\beta(x) = \beta(x'). \tag{6.28}$$

By Claim 1(1), we have $v_i(h_i(\alpha_i(x))) = \beta(x)$. Since $\alpha_i(x)$ is a constant and h_i, v_i are the identity on constants, this implies that $\alpha_i(x) = \beta(x)$. In particular,

$$\beta(x') \overset{(6.28)}{=} \beta(x) = \alpha_i(x) \in \mathrm{const}(T_i) \subseteq \mathrm{const}(T) \cup C. \tag{6.29}$$

By Claim 1(2), this yields $\beta(x') = \alpha_j(x')$, and therefore,

$$u = \alpha_i(x) \overset{(6.29)}{=} \beta(x') = \alpha_j(x') = u',$$

as desired.

Finally, for proving that $\sim$ satisfies condition 3 of Definition 6.27, let $i \in \{1, \ldots, k\}$ and $x, x' \in X_i$. Then,

$$\alpha_i(x) = \alpha_i(x') \overset{\text{Claim 1(3)}}{\iff} \beta(x) = \beta(x') \overset{\text{Claim 2(1)}}{\iff} \alpha_i(x) \sim \alpha_i(x'),$$

as desired.   ⌐

By Claim 2, $(T_1, \alpha_1), \ldots, (T_k, \alpha_k)$ are compatible. Let (T_0, α_0) be their join. We show that

$$T^* := T_0 \cup \bigcup_{i=k+1}^{s} T_i$$

satisfies $\tilde{q}$, where $T_i := \rho_i(T)$ for each $i \in \{k+1, \ldots, s\}$. To this end, we construct an assignment α for ψ such that $T^* \models \psi(\alpha)$.

Claim 3. There is a homomorphism h_0 from T_0 to $\hat{T}_0 := \bigcup_{i=1}^{k} \hat{T}_i$ with $h_0(T_0) = \hat{T}_0$, and $h_0(\alpha_0(\bar{x}_i)) = \beta(\bar{x}_i)$ for each $i \in \{1, \ldots, k\}$.

Proof. Let $r_1, \ldots, r_k$ be the mappings used to construct the join $(\tilde{T}, \tilde{\alpha})$. Then,

$$T_0 = \bigcup_{i=1}^{k} r_i(T_i), \tag{6.30}$$

and for all $i \in \{1, \ldots, k\}$ and $x \in X_i$,

$$\alpha_0(x) = r_i(\alpha_i(x)). \tag{6.31}$$

By Proposition 6.30, each r_i is injective; furthermore, for all $i, j \in \{1, \ldots, k\}$, $u \in \operatorname{dom}(T_i)$ and $u' \in \operatorname{dom}(T_j)$,

$$r_i(u) = r_j(u') \implies u = u', \text{ or: } u \in \alpha_i(X_i), \ u' \in \alpha_j(X_j) \text{ and } u \sim u'. \tag{6.32}$$

Define $h_0 \colon \operatorname{dom}(T_0) \to \operatorname{dom}(\hat{T}_0)$ such that for all $i \in \{1, \ldots, k\}$ and $u \in \operatorname{dom}(r_i(T_i))$,

$$h_0(u) = v_i(h_i(r_i^{-1}(u))). \tag{6.33}$$

We claim that h_0 is a homomorphism from T_0 to $\hat{T}_0$ with $h_0(T_0) = \hat{T}_0$, and that for each $i \in \{1, \ldots, k\}$ we have $h_0(\alpha_0(\bar{x}_i)) = \beta(\bar{x}_i)$.

Step 1: h_0 is well-defined.
Let $u \in \operatorname{dom}(r_i(T_i)) \cap \operatorname{dom}(r_j(T_j))$, where $i, j \in \{1, \ldots, k\}$ are distinct. Let $u_i := r_i^{-1}(u) \in \operatorname{dom}(T_i)$ and $u_j := r_j^{-1}(u) \in \operatorname{dom}(T_j)$. We must show that

$$v_i(h_i(u_i)) = v_j(h_j(u_j)).$$

Since $r_i(u_i) = u = r_j(u_j)$, (6.32) implies that $u_i = u_j$, or: $u_i \in \alpha_i(X_i)$, $u_j \in \alpha_j(X_j)$ and $u_i \sim u_j$. If $u_i = u_j$, then both u_i and u_j are constants, since $\operatorname{nulls}(T_i) \cap \operatorname{nulls}(T_j) = \emptyset$ for $i \neq j$; therefore,

$$v_i(h_i(u_i)) = u_i = u_j = v_j(h_j(u_j)),$$

as desired. On the other hand, let $x_i \in X_i$ and $x_j \in X_j$ such that $u_i = \alpha_i(x_i)$, $u_j = \alpha_j(x_j)$ and $\alpha_i(x_i) \sim \alpha_j(x_j)$. Then Claim 2(1) implies $\beta(x_i) = \beta(x_j)$. By Claim 1(1),

$$v_i(h_i(u_i)) = v_i(h_i(\alpha_i(x_i))) = \beta(x_i) = \beta(x_j) = v_j(h_j(\alpha_j(x_j))) = v_j(h_j(u_j)),$$

as desired. Altogether, this shows that h_0 is well-defined.

Step 2: h_0 is a homomorphism from T_0 to $\hat{T}_0$ with $h_0(T_0) = \hat{T}_0$.
First note that for each $i \in \{1, \ldots, k\}$, we have

$$h_0(r_i(T_i)) \overset{(6.33)}{=} v_i(h_i(T_i)) = \hat{T}_i.$$

Hence,

$$h_0(T_0) \overset{(6.30)}{=} h_0\left(\bigcup_{i=1}^{k} r_i(T_i)\right) = \bigcup_{i=1}^{k} h_0(r_i(T_i)) = \bigcup_{i=1}^{k} \hat{T}_i = \hat{T}_0.$$

Step 3: For each $i \in \{1, \ldots, k\}$, we have $h_0(\alpha_0(\bar{x}_i)) = \beta(\bar{x}_i)$.
We have

$$h_0(\alpha_0(\bar{x}_i)) \overset{(6.31)}{=} h_0(r_i(\alpha_i(\bar{x}_i))) \overset{(6.33)}{=} v_i(h_i(\alpha_i(\bar{x}_i))) \overset{\text{Claim } 1(1)}{=} \beta(\bar{x}_i). \qquad \lrcorner$$

Let h_0 be a homomorphism as in Claim 3. It is easy to extend h_0 to a mapping h on $\mathrm{dom}(T^*) \cup C$ with the following properties:

(1) $h(T_0) = h_0(T_0) = \bigcup_{i=1}^{k} \hat{T}_i$,
(2) $h(T_i) = \hat{T}_i$ for each $i \in \{k+1, \ldots, s\}$, and
(3) $h(c) = c$ for each $c \in C$.

Note that the second condition can be satisfied, since for all distinct $i \in \{k+1, \ldots, s\}$ and $j \in \{1, \ldots, s\}$, we have $\mathrm{nulls}(T_i) \cap \mathrm{nulls}(T_j) = \emptyset$, $T_i \cong T$ and $\hat{T}_i \in poss(T)$. Note also that

$$h(T^*) = \bigcup_{i=1}^{s} \hat{T}_i. \tag{6.34}$$

Furthermore, extend α_0 to an assignment α for $\bar{y}$ such that

$$h(\alpha(y)) = \beta(y) \quad \text{for each } y \in \bar{y}. \tag{6.35}$$

Note that (6.35) holds for all variables y that occur in $\bar{x}_i$ for some $i \in \{1, \ldots, k\}$, because h is an extension of h_0, and α is an extension of α_0. For each variable $y \in \bar{y}$ that does not occur in $\bar{x}_i$ for some $i \in \{1, \ldots, k\}$, we pick an arbitrary value $u \in \mathrm{dom}(T^*) \cup C$ with $h(u) = \beta(y)$ and define $\alpha(y) := u$. Note that such a value u always exists. First recall that the range of β is in $\mathrm{dom}(\bigcup_{i=1}^{s} \hat{T}_i) \cup C$. If $\beta(y) \in \mathrm{dom}(\bigcup_{i=1}^{s} \hat{T}_i)$, then by (6.34) there is some $u \in \mathrm{dom}(T^*)$ with $h(u) = \beta(y)$. On the other hand, if $\beta(y) \in C$, then $h(\beta(y)) = \beta(y)$, because h is the identity on constants, so that we can choose $u = \beta(y)$.

We are finally ready to show that $T^* \models \psi(\alpha)$. First note that by Proposition 6.30(4), we have $\alpha_0(\bar{x}_i) \in R_i^{T_0}$ for each $i \in \{1, \ldots, k\}$; since $T_0 \subseteq T^*$ and α extends α_0, this implies

$$\alpha(\bar{x}_i) \in R_i^{T^*} \quad \text{for each } i \in \{1, \ldots, k\}. \tag{6.36}$$

Furthermore, we have

$$\alpha(\bar{w}_i) \notin Q_i^{T^*} \quad \text{for each } i \in \{1, \ldots, l\}. \tag{6.37}$$

Otherwise, if there is some $i \in \{1, \ldots, l\}$ with $\alpha(\bar{w}_i) \in Q_i^{T^*}$, then by (6.34) and (6.35), we have

$$\beta(\bar{w}_i) \in Q_i^{\bigcup_{i=1}^{s} \hat{T}_i},$$

which is impossible by (6.23). Finally, we have

$$\alpha(v_i) \neq \alpha(v_i') \quad \text{for each } i \in \{1, \ldots, m\}. \tag{6.38}$$

Indeed, let $i \in \{1, \ldots, m\}$. By (6.35), we have $h(\alpha(v_i)) = \beta(v_i)$ and $h(\alpha(v_i')) = \beta(v_i')$. On the other hand, (6.23) implies that $\beta(v_i) \neq \beta(v_i')$, so that $\alpha(v_i)$ and $\alpha(v_i')$ must be distinct.

Altogether, (6.36)–(6.38) imply that $T^* \models \psi(\alpha)$. In particular, Algorithm 6.31 outputs "yes" on input T and $\bar{t}$. $\qquad \square$

6.3.4. *Proof of Proposition 6.4.* We conclude this section by proving Proposition 6.4. Let $M = (\sigma, \tau, \Sigma)$ be a schema mapping, where Σ consists of st-tgds, and let q be a universal query over τ. As in Section 6.3.3, we can assume that $\neg q$ is logically equivalent to a query $\bar{q}$ of the form

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

where each q_i is an existential query of the form

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

and each $\varphi_{i,j}$ is an atomic FO formula or the negation of an atomic FO formula.

Let S be a source instance for M, and let $\bar{t} \in Const^{|\bar{x}|}$. As shown in Section 6.3.1, we have $\bar{t} \notin cert_{\mathrm{GCWA}^*}(q, M, S)$ if and only if there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(\mathrm{Core}(M,S))$ with $\bigcup \mathcal{T} \models \neg q(\bar{t})$. Hence, on input S and $\bar{t}$, a nondeterministic Turing machine can decide whether $\bar{t} \notin cert_{\mathrm{GCWA}^*}(q, M, S)$ by computing $\mathrm{Core}(M,S)$, and by deciding for each $i \in \{1, \ldots, m\}$ whether there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(\mathrm{Core}(M,S))$ with $\bigcup \mathcal{T} \models q_i(\bar{t})$. If so, it accepts the input, and otherwise, it rejects it.

By Theorem 2.1, $\mathrm{Core}(M,S)$ can be computed in time polynomial in the size of S (for fixed M).

In order to check whether there is a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(\mathrm{Core}(M,S))$ with $\bigcup \mathcal{T} \models q_i(\bar{t})$, it suffices to "guess" a set $\mathcal{T}$ of at most

$$s := n_i \cdot \max \{\mathrm{ar}(R) \mid R \in \tau\}$$

instances in $poss(\mathrm{Core}(M,S))$, and to check whether $\bigcup \mathcal{T} \models q_i(\bar{t})$. Indeed, let $\mathcal{T}$ be a set of minimal instances in $poss(\mathrm{Core}(M,S))$ with $\bigcup \mathcal{T} \models q_i(\bar{t})$. Then there is an assignment α for the variables in $\bar{x}$ and $\bar{y}_i$ such that $\alpha(\bar{x}) = \bar{t}$ and $\bigcup \mathcal{T} \models \varphi_{i,j}(\alpha)$ for each $j \in \{1, \ldots, n_i\}$. Without loss of generality, assume that $\varphi_{i,1}, \ldots, \varphi_{i,k}$ (for $0 \leq k \leq n_i$) are all the relational atomic FO formulas in q_i. For each $j \in \{1, \ldots, k\}$, there is an instance $T_j \in \mathcal{T}$ with $T_j \models \varphi_{i,j}(\alpha)$. Let $\mathcal{T}_0' := \{T_1, \ldots, T_k\} \subseteq \mathcal{T}$. Then $\bigcup \mathcal{T}_0' \models \varphi_{i,j}(\alpha)$ for each $j \in \{1, \ldots, k\}$. To obtain a set $\mathcal{T}_0 \subseteq \mathcal{T}$ that satisfies $\bigcup \mathcal{T}_0 \models q_i(\bar{t})$, we extend $\mathcal{T}_0'$ as follows. Let $j \in \{k+1, \ldots, n_i\}$. Then there are at most $\max \{\mathrm{ar}(R) \mid R \in \tau\}$ values that occur in $\varphi_{i,j}(\alpha)$. In particular, we can pick $\max \{\mathrm{ar}(R) \mid R \in \tau\}$ instances from $\mathcal{T}$ that contain all these values. Add those instances to $\mathcal{T}_0'$. The resulting set $\mathcal{T}_0$ is a subset of $\mathcal{T}$, and satisfies $\bigcup \mathcal{T}_0 \models q_i(\bar{t})$, since $\mathcal{T}_0 \models \varphi_{i,j}(\alpha)$ for each $j \in \{1, \ldots, k\}$. Furthermore, $\mathcal{T}_0$ contains at most s instances.

Note also that to find a nonempty finite set $\mathcal{T}$ of minimal instances in $poss(\mathrm{Core}(M,S))$ with $|\mathcal{T}| \leq s$ and $\bigcup \mathcal{T} \models q_i(\bar{t})$, it suffices to consider valuations v of $\mathrm{Core}(M,S)$ with range in C, where C contains all constants in $\mathrm{Core}(M,S)$, all constants in q_i, all constants in $\bar{t}$, and all constants in $\{c_1, \ldots, c_{s \cdot k}\}$, where k is the number of nulls in $\mathrm{Core}(M,S)$, and $c_1, \ldots, c_{s \cdot k}$ is a sequence of pairwise distinct constants that do not occur in $\mathrm{Core}(M,S)$, q_i and $\bar{t}$.

Finally, it is easy for a Turing machine to check whether a given $T \in poss(\mathrm{Core}(M,S))$ is minimal. For each atom $A \in T$, it just has to check that the instance $T \setminus \{A\}$ is not a solution for S under M.

Altogether, given a source instance S for M, and a tuple $\bar{t} \in Const^{|\bar{x}|}$, a nondeterministic Turing machine can check whether $\bar{t} \notin cert_{\mathrm{GCWA}^*}(q, M, S)$. This proves $\mathrm{EVAL}_{\mathrm{GCWA}^*}(M, q) \in$ co-NP, and in particular, Proposition 6.4.

7. Conclusion

A new semantics, called *GCWA*-semantics*, for answering non-monotonic queries in relational data exchange has been proposed. The GCWA*-semantics is inspired by non-monotonic query answering semantics from the area of deductive databases, where the problem of answering non-monotonic queries has been studied extensively since the late seventies. In contrast to non-monotonic query answering semantics proposed earlier in the data exchange literature, the GCWA*-semantics can be applied to a broader class of schema mappings (not just schema mappings defined by tgds and egds), and possesses the following natural properties: (1) it is invariant under logically equivalent schema mappings, and (2) it interprets existential quantifiers "inclusively" as explained in Section 3. Furthermore, under schema mappings defined by st-tgds and egds (and even more general schema mappings like schema mappings defined by right-monotonic $L_{\infty\omega}$-st-tgds), the answers to a query under the GCWA*-semantics can be defined as the certain answers to the query with respect to all ground solutions that are unions of minimal solutions.

However, the GCWA*-semantics is not meant to be a replacement for earlier semantics proposed in the data exchange literature. Each of the earlier semantics is interesting in its own right. In fact, I think that there is no ultimate semantics for answering non-monotonic queries in relational data exchange. Depending on the concrete application, and the user's expectations, one or the other of the proposed semantics may be appropriate. Nevertheless, query answers under the GCWA*-semantics seem to be very natural – especially due to the two properties mentioned above.

We have shown that the problem of answering non-monotonic queries under the GCWA*-semantics can be hard, or even undecidable, in considerably simple settings. Unfortunately, this is true not only for the GCWA*-semantics, but also for earlier semantics. This seems to be the price that one has to pay for automatically inferring "negative data". Nevertheless, we were able to show (Theorem 6.6) that for schema mappings M defined by packed st-tgds, and for universal queries q, there is a polynomial time algorithm that, given the core solution for some source instance S for M as input, outputs the set of answers to q with respect to M and S under the GCWA*-semantics.

Quite a number of interesting research problems remain open. First, I believe that the techniques used for proving Theorem 6.6 can be extended to prove the analogous result for the more general case of schema mappings defined by st-tgds. In fact, it seems that all that has to be done is to provide a proof of Lemma 6.23 for the case that the blocks of T are not packed. Second, a lot of more work has to be done for understanding the complexity of answering non-monotonic queries not only under the GCWA*-semantics, but also under the semantics proposed earlier. The fact that for some schema mappings M defined by st-tgds, and for some existential queries q the data complexity of computing the GCWA*-answers to q under M is hard does not imply that it could not be in polynomial time for other schema mappings defined by st-tgds and other existential queries. Third, we only considered the data complexity of evaluating queries – we did not consider the combined complexity, where the schema mapping and the query to be answered belong to the input. Finally, instead of answering queries under a non-monotonic semantics, it could be an interesting task to study the problem of answering queries using the OWA-semantics, but allow more expressive constraints to explicitly exclude "unwanted" tuples from solutions (rather than implicitly by a variant of the CWA). For instance, instead of using the st-tgd θ in Example 1.1, we could have used $\forall x \forall y \left(R(x,y) \leftrightarrow R'(x,y) \right)$. Then, under the OWA-semantics, the answer to a

query would be as desired. However, this approach requires schema mappings to be fully specified.

Acknowledgment

I am grateful to Nicole Schweikardt for many helpful discussions on the subject and comments on the proceedings version of this paper. Also, I thank the referees of this paper and the referees of the conference version for their comments and suggestions regarding the presentation of this paper's results.

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, June 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, June 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. In *Proceedings of the 12th International Conference on Database Theory (ICDT)*, pages 73–83, Mar. 2009.

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

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

[8] 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, June 2008.

[9] 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.

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

[11] 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, June 2008.

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

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

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

[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, Mar. 2010.

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

[17] G. Gottlob, R. Pichler, and V. Savenkov. Normalization and optimization of schema mappings. *PVLDB*, 2(1):1102–1113, 2009.

[18] L. M. Haas, M. A. Hernández, C. T. H. Ho, L. Popa, and M. Roth. Clio grows up: From research prototype to industrial tool. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 805–810, June 2005.

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

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

[21] 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.

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

[26] 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, June 2005.

[27] P. G. Kolaitis, J. Panttaja, and W. C. Tan. The complexity of data exchange. In *Proceedings of the 25th ACM Symposium on Principles of Database Systems (PODS)*, pages 30–39, June 2006.

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

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

[30] L. Libkin and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*, pages 139–148, June 2008.

[31] W. Lipski, Jr. On semantic issues connected with incomplete information in databases. *ACM Transactions on Database Systems*, 4(3):262–296, Sept. 1979.

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

[33] 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, June 2009.

[34] G. Mecca, P. Papotti, S. Raunich, and M. Buoncristiano. Concise and expressive mappings with +Spicy. In *PVLDB*, volume 2, pages 1582–1585, 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, June 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] N. C. Shu, B. C. Housel, R. W. Taylor, S. P. Ghosh, and V. Y. Lum. EXPRESS: A data EXtraction, Processing, and REStructuring system. *ACM Transactions on Database Systems*, 2(2):134–174, June 1977.

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