Recovering Exchanged Data

Gösta Grahne
Concordia University
Montreal, Canada, H3G 1M8
grahne@cs.concordia.ca

Ali Moallemi
Concordia University
Montreal, Canada, H3G 1M8
moa_ali@encs.concordia.ca

Adrian Onet [*]
Concordia University
Montreal, Canada, H3G 1M8
adrian_onet@yahoo.com

ABSTRACT

The inversion of data exchange mappings is one of the thorniest issues in data exchange. In this paper we study inverse data exchange from a novel perspective. Previous work has dealt with the static problem of finding a target-to-source mapping that captures the "inverse" of a source-to-target data exchange mapping. As we will show this approach has some drawbacks when it comes to actually applying the inverse mapping in order to recover a source instance from a materialized target instance. More specifically *(1):* As is well known, the inverse mappings have to be expressed in a much more powerful language than the mappings they invert. *(2):* There are simple cases where a source instance computed by the inverse mapping misses sound information that one may easily obtain when the particular target instance is available. *(3):* In some cases the inverse mapping can introduce unsound information in the recovered source instance.

To overcome these drawbacks we focus on the dynamic problem of recovering the source instance using the source-to-target mapping as well as a given target instance. Similarly to the problem of finding "good" target instances in forward data exchange, we look for "good" source instances to restore, i.e. to materialize. For this we introduce a new semantics to capture instance based recovery. We then show that given a target instance and a source-to-target mapping expressed as set of tuple generating dependencies, there are chase-based algorithms to compute a representative finite set of source instances that can be used to get certain answers to any union of conjunctive source queries. We also show that the instance based source recovery problem unfortunately is coNP-complete. We therefore present a polynomial time algorithm that computes a "small" set of source instances that can be used to get sound certain answers to any union of conjunctive source queries. This algorithm is then extended to extract more sound information for the case when only conjunctive source queries are allowed.

[*]Contact author.

Categories and Subject Descriptors

H.2.5 [**Heterogeneous Databases**]: Data translation

General Terms

Algorithms; Theory

Keywords

Chase; Date Exchange; Data Repair; Incomplete databases; Complexity

1. INTRODUCTION

Data exchange mappings are now widely used in translating data from one database (the source database) into data in another database (the target database) that has a schema distinct from the source database. A data exchange mapping $\mathcal{M}$ is specified by a triple $(\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S}$ and $\mathbf{T}$ are relational schemas such that $\mathbf{S} \cap \mathbf{T} = \emptyset$, and Σ is a set of constraints (typically, formulas in some logic) expressing the relationship between $\mathbf{S}$ (the source schema) and $\mathbf{T}$ (the target schema). A data exchange mapping $\mathcal{M}$ is identified with the set of pairs (I, J), where I is an instance over $\mathbf{S}$, J is an instance over $\mathbf{T}$, and $I \cup J \vDash \Sigma$. We will henceforth express this by writing $(I, J) \in \mathcal{M}$.

In their seminal paper [13], Fagin et al. proposed the language of tuple generating dependencies (tgds) to represent such schema mappings. The language of tgds was shown to be rich enough to capture most mappings occurring in practice. We recall that a tgd is a first order formula of the form $\forall \bar{x} \forall \bar{y}\ (\alpha(\bar{x}, \bar{y}) \to \exists \bar{z}\ \beta(\bar{x}, \bar{z}))$, where α and β are conjunctions of relational atoms and $\bar{x}$, $\bar{y}$ and $\bar{z}$ are sequences of variables. For simplicity, when representing tgds we will omit the universal quantifiers. *Source-to-target* tgds (s-t tgds) are those tgds where α contains only relation names from $\mathbf{S}$ and β contains only relation names from $\mathbf{T}$.[1]

In the framework for managing data exchange mappings, Bernstein [10] proposed a set of operators for the mappings. The operators included composition, merging, matching and

[1]Additionally, one may also specify *target constraints.* In this paper we however only consider constraints Σ, where Σ is a (finite) set of s-t tgds.

inverse. Two of these operators has gained a lot of attention in the database community, namely the composition [14, 22, 5, 3, 17, 4, 18] and the inversion of schema mappings [12, 15, 8, 5, 6, 17, 16, 7]. In the case of composition, the semantics of [14] has been widely adopted as standard. As mappings actually are binary relations, the standard composition of $\mathcal{M}$ and $\mathcal{M}'$ is simply $\mathcal{M} \circ \mathcal{M}'$, where $\circ$ is relational composition. In contrast, due to the fact that a straightforward application of the standard algebraic concept of mapping inversion renders most schema mappings non-invertible, several alternative semantics have been proposed. In [12], Fagin et al. defined $\mathcal{M}'$ to be an inverse of a mapping $\mathcal{M}$, if $\mathcal{M} \circ \mathcal{M}' = \{(I, I') : I \subseteq I'\}$, where I and I' are instances over $\mathbf{S}$. This semantics, later named *Fagin-inverse*, turned out to be too restrictive, as even simple source-to-target tgds do not have Fagin-inverses. In a subsequent work Fagin et al. [15] introduced the notion of *quasi-inverse* that does not differentiate between source instances that are equivalent for data exchange. Loosely speaking, $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$, if $\mathcal{M} \circ \mathcal{M}' \circ \mathcal{M} = \mathcal{M}$. Even with this relaxation the quasi-inverse remains rather restrictive. To overcome the restrictions, Arenas et al. [8] introduced the notion of a *recovery.* A mapping $\mathcal{M}'$ is a recovery of a mapping $\mathcal{M}$ if $(I, I) \in \mathcal{M} \circ \mathcal{M}'$ for any instance I over $\mathbf{S}$. As there can be several recoveries of a given mapping, Arenas et al. considered the notion of a *maximum recovery.* A mapping $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$ if $\mathcal{M}'$ is a recovery of $\mathcal{M}$, and for all recoveries $\mathcal{M}''$ of $\mathcal{M}$ it holds that $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$. This new operation of maximum recovery relaxed the previous inverse operations, and in [8] it was shown that any mapping specified by a set of source-to-target tgds has a maximum recovery. In the same paper it was also shown that large classes of mappings, including ones containing target dependencies, have maximum recoveries. Unfortunately, the maximum recovery mappings of mappings specified by source-to-target tgds cannot be specified for target-to-source tgds. In [6] Arenas et al. showed that in case one is interested in sound (in the recovery semantics) certain answers for conjunctive queries, the maximum recovery mapping of a mapping expressed by a set of source-to-target tgds can be expressed by a set of target-to-source tgds generalized so that the body of these tgds allows the presence of the inequality predicate ($\neq$) and a special predicate that identifies constants. More recently Fagin et al. [16] introduced the notion of *extended-recovery* mappings that also deals with source instances that contain nulls. In this case it is shown that all mappings specified by source-to-target tgds have an extended-recovery mapping, and that the recovery mapping can be represented by a set of target-to-source tgds with inequalities and the constant predicate allowed to occur in the body, and with disjunctions and Skolem functions in the head (actually second order disjunctive tgds).

Although all these inverses play an important role in the development of the model management framework of Bernstein [10], they are less likely to be used in practice due their complicated structure involving disjunctions, and due to the fact that when actually restoring a source database from a target database, the inverse mapping does not necessarily recover the source completely. For example, let the source-to-target mapping be specified by dependency set $\Sigma = \{R(x, y) \rightarrow S(x), P(y)\}$. In this case the maximum recovery mapping will consist of the following set of target-to-source tgds $\Sigma' = \{\xi'_1, \xi'_2\}$, where:

$$\begin{aligned} \xi'_1 &= S(x) \rightarrow \exists y\, R(x, y); \\ \xi'_2 &= P(y) \rightarrow \exists x\, R(x, y). \end{aligned} \tag{1}$$

Consider target instance $J = \{S(a), P(b_1), \dots, P(b_n)\}$. In this case chasing J with Σ' will yield source instance

$$I = \{R(a, Y), R(X_1, b_1), \dots, R(X_n, b_n)\}, \tag{2}$$

where y and the x_i's are nulls. It is however easy to see that the information contained in Σ and J allows us to deduce that the source instance actually must be

$$I' = \{R(a, b_1), R(a, b_2), \dots, R(a, b_n)\}. \tag{3}$$

We would therefore expect the conjunctive source query $Q(x) = \{(x) : R(x, b_2)\}$ to return tuple (a), which is not the case if the recovered source is I in (2). Similar anomalies plague the other semantics as well. This means that these semantics are not complete even when considering only conjunctive queries.

Another practical issue with the proposed inverse semantics is that these semantics do not seem to be "data-exchange sound." As an example consider the following set of full s-t tgds $\Sigma = \{\xi_1, \xi_2, \xi_3\}$, where:

$$\begin{aligned} \xi_1 &= R(x) \rightarrow T(x); \\ \xi_2 &= R(x) \rightarrow S(x); \\ \xi_3 &= M(x) \rightarrow S(x). \end{aligned} \tag{4}$$

For this example both the maximum recovery [8] and the extended-recovery mapping [16] are logically equivalent with $\Sigma' = \{\xi'_1, \xi'_2\}$, where:

$$\begin{aligned} \xi'_1 &= T(x) \rightarrow R(x); \\ \xi'_2 &= S(x) \rightarrow R(x) \vee M(x). \end{aligned} \tag{5}$$

If we now consider target instance $J = \{S(a)\}$ and use Σ' to restore the source database, we will obtain the following three possible source instances: $I_1 = \{R(a)\}$, $I_2 = \{M(a)\}$ and $I_3 = \{R(a), M(a)\}$. It is easy to observe that neither I_1 nor I_3 can be source instances that would yield target instance J in a data exchange process using Σ and using the extended-chase procedure from [11].

In this paper we will consider the problem of recovering the source instance when the source-to-target mapping and the target instance are given. In the semantics considered a source instance I is a recovery of a target instance J under a set of s-t tgds Σ if the following holds:

1. $I \cup J \vDash \Sigma$, and
2. all tuples from J are justified through Σ by the existence of some tuples in I.

The second requirement guarantees that all the tuples in J were obtained from instance I. Thus the empty instance will not be considered as recovery even though we have that $\emptyset \cup J \vDash \Sigma$ for any target instance J and any set of s-t tgds Σ. We note that not all target instances are recoverable given a set of s-t tgds. Consider for example the dependencies in equation (4) and target instance $J = \{T(a)\}$.

It is easy to see that in the general case there might be an infinite number of recovered source instances. We show that there is a chase-based method to construct a finite set of such recoveries, and that this set may be used to get the certain answer for any source UCQ. This chase will be applied with the set of target-to-source tgds obtained from the initial source-to-target tgds by inverting the arrows in the tgds. As we will see our chase is not the standard chase [9, 13]; as it is a more involved computation. There are three main cases that need to be covered by our chase process in order to ensure the soundness (wrt the semantics) and completeness (wrt UCQ certain answers) of the recovered set of instances.

As a first case consider the following source-to-target tgds

$$\Sigma = \{R(x) \to S(x); \\ M(y) \to S(y)\}$$

and target instance $J = \{S(a)\}$. By reversing the arrows we get

$$\Sigma^{-1} = \{S(x) \to R(x); \\ S(y) \to M(y)\}$$

Using the standard chase on J with Σ^{-1} will return instance $I = \{R(a), M(a)\}$. Even though I is a recovery it is not the only one, as $I_1 = \{R(a)\}$ and $I_2 = \{M(a)\}$ also are recoveries of J under Σ. In order to achieve this our chase method will not apply all the existing triggers (homomorphisms from the body of the tgds to the instance) but only enough to be sure that the target instance is covered. The set of triggers chosen will determine the recovery. We note that the target instance and the recovery do not need to be a model of Σ^{-1}. In our example neither I_1 nor I_2 is a model of Σ^{-1} and J.

For the second case consider target instance $J = \{S(a)\}$ and Σ from equation (4). It is easy to see that when chasing instance J with Σ^{-1} we should not trigger the tuple generating dependency $S(x) \to R(x)$. This is because a tuple $R(a)$ in the source instance requires a tuple $T(a)$ in target instance J. Since $T(a) \notin J$ the soundness of the recovery is violated. This led us to the notion of subsumption constraints that we obtain from the set of s-t tgds Σ. The subsumption constraints will ensure that such unsound recoveries do not occur.

Finally, for the third case consider the set of source-to-target tgds $\Sigma = \{\xi_1, \xi_2\}$, where:

$$\begin{aligned} \xi_1 &= R(x, x, y) \to T(x); \\ \xi_2 &= R(v, w, z) \to S(z). \end{aligned} \tag{6}$$

Let $J = \{T(a), S(b)\}$ be the target instance. By simply chasing instance J with Σ^{-1} we will obtain source instance $I = \{R(a, a, X), R(Y, Z, b)\}$, where X, Y and Z are nulls. It is easy to see that I is not a recovery of J under Σ because the existence of tuple $R(a, a, X)$ in the source instance requires the existence of tuple $S(X)$ in the target instance. In this case the chase process needs to be "smart enough" to "see" that the recoveries are instances of the form $I_1 = \{R(a, a, b)\}$, $I_2 = I_1 \cup \{R(Y_2, Z_2, b)\}$, $I_3 = I_2 \cup \{R(Y_3, Z_3, b)\}$, $\ldots$, $I_n = I_{n-1} \cup \{R(Y_n, Z_n, b)\}$, $\ldots$, where the Y_i's and Z_i's are distinct nulls.

The results. First we precisely define the instance recovery semantics, that is, what makes a source instance a valid recovery given a set of source-to-target tgds and a target instance. This semantics contains both the universal solutions [13] and canonical solutions [20]. Next we show that the problem of testing if a target instance J is recoverable under a set of source-to-target tgds Σ is an NP-complete problem (the same complexity also holds when testing if J is a universal or canonical solution for some source instance I and Σ). We then show that in order to obtain complete certain answers for any UCQ query over the source schema it is sufficient to compute a finite number of recoveries. The downside is that the problem of testing if a tuple t is in the certain answer of such a UCQ is coNP-complete, and that the hardness result holds even when the query considered is a conjunctive one. On the other hand, we show that there is a simple tractable algorithm that computes a source instance that is part of *any* recovery. Such a source instance can be used to compute *sound* certain answers to any UCQ. Next the method is extended, retaining tractability, to obtain a source instance that has a homomorphism into every recovery. This source instance can be used to compute sound certain answers to any CQ. We also compare our recoveries with the instances obtained by chasing the target with the inverse/recovery mappings considered in the literature.

Interestingly enough, query answering over the recovered instances is a generalization of query answering over materialized views. Our current work extends the previous work [1] by considering s-t tgds instead of the full GAV dependencies used for views. Also we introduce tractable cases for both the UCQ and CQ query classes.

2. PRELIMINARIES

In this section we introduce the basic technical preliminaries and definitions. More information on relational database theory can be obtained from e.g. [2]. We will consider the complexity classes P, NP and coNP. For the definition of these classes we refer to [23].

A finite mapping f, where $f(a_i) = b_i$, for $i = 1, \ldots, n$, will be represented as $\{a_1/b_1, a_2/b_2, \ldots, a_n/b_n\}$. The composition of mappings f and g is $f \circ g$, defined as $(f \circ g)(x) = f(g(x))$. If f is a mapping with domain A, and $S \subseteq A$, the restriction of f to S is denoted $f|_S$.

A *schema* $\mathbf{R}$ is a finite set $\{R_1, \ldots, R_n\}$ of relational symbols, each R_i having a fixed arity k_i. Let $Cons$ be a countably infinite set of constants, and $Nulls$ a countably infinite set of nulls, such that $Cons \cap Nulls = \emptyset$. An *instance* I of $\mathbf{R}$ is an interpretation that assigns to each relational symbol R_i a finite k_i-ary relation $R_i^I \subset (Cons \cup Nulls)^{k_i}$. An instance I over $\mathbf{R}$ is usually identified with the set of tuples $\{R_i(\bar{a}) : \bar{a} \in R_i^I, R_i \in \mathbf{R}\}$. We denote with $|I|$ the size of I, i.e. the number of tuples in I, and with $dom(I)$ we denote the set of all constants and nulls that occur in I. An instance I, such that $dom(I) \subseteq Cons$ is called a *ground instance*. If I and J are two instances over the same schema $\mathbf{R}$, we denote with $I \subseteq J$ the fact that $R_i^I \subseteq R_i^J$, for all $i \in \{1, \ldots, n\}$.

Let I and J be instances over a schema $\mathbf{R}$. A *homomorphism* from I to J is a mapping h on $Cons \cup Nulls$, identity on $Cons$, and extended to tuples and relations in the natural way, such that $h(R_i^I) \subseteq R_i^J$, for all $i \in \{1, \ldots, n\}$. The existence of a homomorphism between instance I and instance J is denoted $I \to J$.

In data-exchange systems the mapping between the source and the target schema is usually expressed as a set of *source-to-target tuple generating dependencies (s-t tgds)*. An s-t tgd

ξ is a first order formula of the form

$$\forall \bar{x} \forall \bar{y}\; \alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\; \beta(\bar{x}, \bar{z}), \tag{7}$$

where $\alpha(\bar{x}, \bar{y})$ is a conjunction of atoms over the source schema, $\beta(\bar{x}, \bar{z})$ is a conjunction of atoms over the target schema, and $\bar{x}$, $\bar{y}$ and $\bar{z}$ are sequences of variables. In case $\bar{z} = \epsilon$, the tgd is called a *full* tgd. Similarly, in case $\bar{y} = \epsilon$ the tgd is called a *quasi-guarded* tgd. Moreover, by $vars(\xi)$ we mean the set of all variables in $\bar{x}$, $\bar{y}$, and $\bar{z}$. When there is no danger of confusion we will also regard a sequence $\bar{x}$ as a set, and write $x_2 \in \bar{x}$, when for example $\bar{x} = (x_1, x_2, x_3)$. We will also often view a conjunction of atoms as a set of atoms, i.e. as an instance where each variable corresponds to a null value. A source instance I and target instance J is said to *satisfy* dependency ξ, denoted $(I, J) \models \xi$, if $I \cup J$ is a model of ξ in the model-theoretic sense. This is extended to sets of s-t tgds Σ, by stipulating that $(I, J) \models \Sigma$ if $(I, J) \models \xi$ for all $\xi \in \Sigma$. For an s-t tgd ξ of the form (7), by ξ^{-1} we denote the following first order formula, called the *reverse* of ξ:

$$\forall \bar{x} \forall \bar{z}\; \beta(\bar{x}, \bar{z}) \rightarrow \exists \bar{y}\; \alpha(\bar{x}, \bar{y}). \tag{8}$$

Note that if ξ is quasi-guarded, then ξ^{-1} is a full tgd. For a set Σ of s-t tgds we define $\Sigma^{-1} = \{\xi^{-1} : \xi \in \Sigma\}$. In the rest of the paper, we assume without loss of generality, that for each set of s-t tgds every two tgds from that set do not share any variables. For simplicity, we will often omit the universal quantifiers when representing tgds.

Given a source instance I and an s-t tgd ξ of the form $\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\; \beta(\bar{x}, \bar{z})$, with $Chase(\xi, I)$ we denote the target instance constructed as follows: Start with $Chase(\xi, I) = I$. For each homomorphism h such that $h(\alpha(\bar{x}, \bar{y})) \subseteq I$, extend h to h' such that $h'(z)$ is a new null that was not used before, for each $z \in \bar{z}$. Then add tuples $h'(\beta(\bar{x}, \bar{z}))$ to $Chase(\xi, I)$. The chase process is extended to a set Σ of s-t tgds by $Chase(\Sigma, I) = \bigcup_{\xi \in \Sigma} Chase(\xi, I)$. It was shown in [13] that $Chase(\Sigma, I) \rightarrow J$ for any target instance J such that $(I, J) \models \Sigma$. If H is a set of homomorphisms from the bodies of tgds in Σ to I, we denote by $Chase_H(\Sigma, I)$ the subset of $Chase(\Sigma, I)$ obtained by using only homomorphisms from H.

A conjunctive query Q over schema $\mathbf{R}$ is an expression of the form $\{(\bar{x}) : \exists \bar{y}\; \alpha(\bar{x}, \bar{y})\}$, where $\alpha(\bar{x}, \bar{y})$ is a conjunction of atoms over $\mathbf{R}$ and $\bar{x}$, $\bar{y}$ are sequences of variables. Given an instance I over schema $\mathbf{R}$, the result of the conjunctive query Q on I is:

$$Q(I) = \{(h(\bar{x})) \; : \; h(\alpha(\bar{x}, \bar{y})) \in I \text{ for some homomorphism } h\}. \tag{9}$$

With $Q(I){\downarrow}$ we denote the set of those tuples from $Q(I)$ that do not contain any values from $Nulls$. A query with no free variables is called a *Boolean query*. The class of all conjunctive queries is denoted CQ. A *union Q of conjunctive queries* over schema $\mathbf{R}$ is an expression of the form

$$\{(\bar{x}) \; : \; \exists \bar{y}\; (\alpha_1(\bar{x}, \bar{y}_1) \vee \alpha_2(\bar{x}, \bar{y}_2) \vee \ldots \vee \alpha_n(\bar{x}, \bar{y}_n)), \tag{10}$$

where $\bar{y} = \bar{y}_1 \cup \bar{y}_2 \cup \ldots \cup \bar{y}_n$ and each $\alpha_i(\bar{x}, \bar{y}_i)$ is a conjunction of atoms over $\mathbf{R}$. The result of applying Q on an instance I is defined as

$$Q(I) = \{(h(\bar{x})) \; : \; h(\alpha(\bar{x}, \bar{y}_i)) \in I \text{ for some } 1 \leq i \leq n \text{ and homomorphism } h\}. \tag{11}$$

With UCQ is denoted the class of all union of conjunctive queries.

3. INSTANCE BASED RECOVERY

In this section we introduce the notion of *instance based recovery*, not be confused with the notion of a *recovery mapping* introduced by Arenas et al. in [8]. We focus on recovering a source instance I, given a target instance J and an s-t mapping $\mathcal{M}$. This is in contrast with [12, 15, 8, 5, 6, 17, 16, 7] that consider the problem of computing a general target-to-source mapping $\mathcal{M}'$, given a source-to-target mapping $\mathcal{M}$. While such a mapping $\mathcal{M}'$ can be used to compute a source instance from a given target instance, the above cited papers mostly focus on the role of $\mathcal{M}'$ in model management. It turns out that our semantics differs from the semantics previously considered. We argue that our semantics is the natural one for practical data recovery, i.e. for restoring a source instance. In this paper we will focus only on mappings specified by a set of s-t tgds.

Definition 1. **(Minimal solution)** *Given a set of s-t tgds Σ, a source instance I, and a target instance J, we say that J is a minimal solution with respect to Σ and I, if $(I, J) \models \Sigma$ and for any $J' \subset J$, it is the case that $(I, J') \not\models \Sigma$.*

Example 1. *Consider $\Sigma = \{S(x) \rightarrow \exists y\; T(x, y)\}$, and target instance $J_1 = \{T(a, b), T(b, c)\}$. In this case J_1 is a minimal solution wrt Σ and $I_1 = \{S(a), S(b)\}$, but J_1 is not a minimal solution wrt Σ and $I_2 = \{S(a)\}$, even though $(I_2, J_1) \models \Sigma$. Note that there are target instances that are not a minimal solution wrt Σ and any source instance I. For example, $J_2 = \{T(a, b), T(a, c)\}$ is not a minimal solution wrt Σ for any source instance I.*

Definition 2. **(Justified solution)** *Given a set of s-t tgds Σ, a target instance J is said to be justified by source instance I under Σ, if*

1. *$(I, J) \models \Sigma$, and*
2. *$J \rightarrow J'$, for some minimal solution J' wrt Σ and I.*

It is easy to see that the *universal solutions* [13] and the *canonical solutions* [20] are justified solutions. On the other hand, there exists justified solutions that are neither universal nor canonical solutions. E.g. instance J_1 in Example 1 is a justified solution by source instance I_1 under Σ, but is not a universal or canonical solution for I_1 and Σ. Note that all the results to be presented hold even if one considers only universal solution (or canonical solution) based semantics, i.e. only universal (canonical) solutions are considered as recoverable. Using the notion of justified solutions, it is clear that it is not possible to find recoveries for all target instances. The following definition describes the target instances for which one may compute recoveries.

Definition 3. **(Valid for recovery)** *Given a set of s-t tgds Σ, a target instance J is said to be valid for recovery under Σ, if there exists a source instance I such that J is justified by I under Σ. Such a source instance I is said to be a recovery for J under Σ. With $\mathsf{REC}(\Sigma, J)$ we denote the set of all recoveries for J under Σ. We thus have*

$$\mathsf{REC}(\Sigma, J) = \{I : J \textit{ is justified for } I \textit{ under } \Sigma\}.$$

Note that the above semantics is more restrictive than the corresponding data-exchange semantics in [13]. Thus, we will not allow mappings that contain empty source instances and non-empty target instances. This restriction is natural as it considers only pairs of instances (I, J) such that each tuple in J is "justified" by the presence of some tuples in I. With this we have the certain answer for Σ and target instance J defined as

$$\mathsf{CERT}(Q, \Sigma, J) = \bigcap_{I \in \mathsf{REC}(\Sigma, J)} Q(I).$$

Definition 4. **($\mathcal{C}$-universal recovery)** *Let $\mathcal{C}$ be a class of queries. A set $\mathcal{I}$ of source instances is said to be a $\mathcal{C}$-*universal recovery*, if*

$$\mathsf{CERT}(Q, \Sigma, J) = \bigcap_{I \in \mathcal{I}} Q(I),$$

for every source query $Q \in \mathcal{C}$,

Because of their importance and frequency of use, the classes of CQ- and UCQ-universal recoveries have been widely studied in most of papers recently published on schema mapping and inversion. Accordingly, in this paper we focus only on the class of CQ- and UCQ-universal recoveries.

4. HOMOMORPHISMS

Let ξ be an s-t tgd of the form $\alpha(\bar{x}, \bar{y}) \to \exists \bar{z}\ \beta(\bar{x}, \bar{z})$, where α and β are conjunctions of predicates over source and target instance, respectively. By $head(\xi)$ we mean the set of atoms in $\beta(\bar{x}, \bar{z})$ and by $body(\xi)$ we mean the set of atoms in $\alpha(\bar{x}, \bar{y})$. Given a target instance J, with $\mathsf{HOM}(\xi, J)$ we denote the set of all homomorphisms $h : dom(head(\xi)) \to dom(J)$, with $h(\beta(\bar{x}, \bar{z})) \subseteq J$. That is,

$$\mathsf{HOM}(\xi, J) = \{h :\ h(\beta(\bar{x}, \bar{z})) \subseteq J\}.$$

We extend this definition to a set Σ of source-to-target tgds, by $\mathsf{HOM}(\Sigma, J) = \bigcup_{\xi \in \Sigma} \mathsf{HOM}(\xi, J)$. Note that because each tgd in Σ contains distinct variables, each homomorphism $h \in \mathsf{HOM}(\Sigma, J)$ uniquely identifies a source-to-target tgd in Σ, which we denoted by ξ_h.

To better visualize the notions introduced in this section we illustrate all the definitions through the following running example:

Example 2. *Let $\Sigma = \{\xi, \rho, \sigma\}$, where:*

$$\xi = R(x, x, y) \to \exists z\ S(x, z);$$
$$\rho = R(u, v, w) \to T(w);$$
$$\sigma = D(k, p) \to T(p).$$

For target instance $J = \{S(a, b), T(c), T(d)\}$, the set of all homomorphisms is:

$$\mathsf{HOM}(\Sigma, J) = \{h_1 = \{x/a, z/b\}, h_2 = \{w/c\}, h_3 = \{w/d\}, h_4 = \{p/c\}, h_5 = \{p/d\}\}.$$

Note in the previous example that both homomorphisms h_2 and h_4 cover the same tuple $T(c)$ in J. This brings us to the following definition.

Definition 5. **(Covering of a target instance.)** *Let Σ be a set of s-t tgds and J a target instance. Then*

$$\mathsf{COV}(\Sigma, J) = \{H \subseteq \mathsf{HOM}(\Sigma, J)\ :\ \bigcup_{h \in H} h(head(\xi_h)) = J\}.$$

Example 3. *Returning to the set of s-t tgds Σ, target instance J, and set of homomorphisms $\mathsf{HOM}(\Sigma, J)$ from Example 2 we have*

$$\mathsf{COV}(\Sigma, J) = \big\{\{h_1, h_2, h_3\}, \{h_1, h_2, h_3, h_4\}, \{h_1, h_2, h_5\}, \{h_1, h_4, h_5\}, \{h_1, h_2, h_3, h_4, h_5\}, \{h_1, h_2, h_3, h_5\}, \{h_1, h_3, h_4\}, \{h_1, h_2, h_4, h_5\}, \{h_1, h_3, h_4, h_5\}\big\}.$$

The intuition behind the covering notion is that the tuples in the target instance can be generated, in a data-exchange process, only using variable mappings from the homomorphisms in one of these coverings. For example, by chasing source instance $I = \{R(a, a, c), D(a, d)\}$ it will generate target instance J by using the covering $\{h_1, h_2, h_5\}$.

In this example we also observe that the source instance may not contain the tuple $R(a, a, a)$, even if homomorphism h_1 will cover tuple $S(a, b)$. This is because, based on the second dependency, the target instance would need to contain tuple $T(a)$ as well. This gives the intuition for the notion of subsumption constraint introduced by the next definitions.

Definition 6. **(Minimal subsummit of a tgd)** *Let Σ be a set of s-t tgds, $\{\xi_1, \ldots, \xi_n\} \subseteq \Sigma$ and ξ_0 a s-t tgd in $\Sigma \setminus \{\xi_1, \ldots, \xi_n\}$. We say that $\{\xi_1, \ldots, \xi_n\}$ is a subsummit for s-t tgd ξ_0, if there are mappings $\theta_i : vars(\xi_i) \to V$, where $V = \mathsf{Vars} \setminus \bigcup_{i=0}^{n} vars(\xi_i)$, such that when each θ_i is extended to be identity on constants and homomorphically to sets of atoms, it holds that*

$$\theta_0(body(\xi_0)) \subseteq \theta_1(body(\xi_1)) \cup \ldots \cup \theta_n(body(\xi_n))$$

We also require that each θ_i, for $i = 1, \ldots, n$, maps each variable y that occurs in $body(\xi_i)$ but not in $head(\xi_i)$ to a unique variable in V. If in addition the inclusion does not hold for any proper subset of $\{\xi_1, \ldots, \xi_n\}$, we say that $\{\xi_1, \ldots, \xi_n\}$ is a minimal subsummit of ξ_0 with $\{\theta_0, \theta_1, \ldots, \theta_n\}$.

Intuitively, the previous definition states that if a source instance I, recovered from a target instance J, triggers the s-t tgds $\{\xi_1, \ldots, \xi_n\}$ in a chase process, then instance I will also trigger dependency ξ_0.

Example 4. *Consider the set of source-to-target tgds Σ and instance J from Example 2. It easy to see that for homomorphisms $\theta_0 = \{u/r_1, v/r_1, w/r_2\}$ and $\theta_1 = \{x/r_1, y/r_2\}$ we have $\theta_0(body(\rho)) \subseteq \theta_1(body(\xi))$, meaning that $\{\xi\}$ is a minimal subsummit for ρ with $\{\theta_0, \theta_1\}$. Intuitively the subsummit is stating that the existence of a tuple of the form $R(r_1, r_1, r_2)$ in a source instance recovered by the existence of a tuple in the target instance under relation S and tgd ξ, will also trigger tgd ρ and thus a corresponding tuple $T(r_1)$ will need to exists in the target instance. Note that $\{\rho\}$ can't be a subsummit for ξ for any set of homomorphisms because variables u and v needs to be mapped to distinct values.*

We next define the relationship between subsummits and homomorphisms.

Definition 7. **(Subsumption constraints.)** *Let Σ be a set of s-t tgds. Then*

$$\mathsf{SUB}(\Sigma) = \big\{\theta_1, \theta_2, \ldots, \theta_n \to \theta_0 :\ \{\xi_1, \ldots, \xi_n\} \subseteq \Sigma \text{ is a minimal subsummit of } \xi_0 \in \Sigma \text{ with } \{\theta_0, \theta_1, \theta_2, \ldots, \theta_n\}\big\}.$$

From Example 4 we have constraint $\{\theta_1 \to \theta_0\} \subseteq \mathsf{SUB}(\Sigma)$. On the other hand, from the same example we also have $\{\theta_0 \to \theta_1\} \notin \mathsf{SUB}(\Sigma)$. We next define what it means for a set of homomorphisms to satisfy a subsumption constraint.

Definition 8. **(Model of SUB)** *Let H be a set of homomorphisms and ζ a subsumption constraint of the form $\theta_1, \theta_2, \ldots, \theta_n \to \theta_0$. We say that H is a model of ζ, denoted $H \models \zeta$, if for all $i \in \{1, \ldots, n\}$ and mappings m, such that $(m \circ \theta_i)|_{vars(head(\xi_{h_i}))} = h_i \in H$ there exists an extension m' of m such that $(m' \circ \theta_0)|_{vars(head(\xi_{h_0}))} = h_0 \in H$.*

A subsumption constraint ζ is said to be *tautological* if for any set H of homomorphisms we have $H \models \zeta$. From now on we will consider the set $\mathsf{SUB}(\Sigma)$ containing only non-tautological subsumption constraints. The previous definition is extended to sets Υ of subsumption constraints, by $H \models \Upsilon$ iff $H \models \zeta$, for all $\zeta \in \Upsilon$.

Example 5. *Continuing our Example 2 we have, by removing all tautological subsumption constraints, that the set $\mathsf{SUB}(\Sigma)$ equals $\{\theta_1 \to \theta_0\}$. For the set of homomorphisms $H = \{h_1, h_4, h_5\} \in \mathsf{COV}(\Sigma, J)$ we have that $H \not\models \mathsf{SUB}(\Sigma)$. This is because the existence of $h_1 = \{x/a, z/b\}$ in H, based on subsumption constraint $\{\theta_1 \to \theta_0\}$, requires the existence of a homomorphism over variables in ρ in H. Intuitively this restriction states that by using only the homomorphisms in H we will not be able to construct a source instance that will be a recovery for J even if H is a cover for target instance J. On the other hand, the covering $\{h_1, h_2, h_3\}$ is a model of $\mathsf{SUB}(\Sigma)$.*

Let H be a set of homomorphisms, Σ a set of s-t tgds, and I an instance. Recall that $Chase_H(\Sigma, I)$ denotes the result of applying the standard chase with Σ on I, with the restriction that only homomorphisms from H are used to trigger dependencies. For example consider instance $I = \{R(a), R(b)\}$, and the set of source-to-target dependencies

$$\Sigma = \{R(x) \to \exists y\, T(x, y);\\ R(z) \to \exists v\, V(z, v)\}.$$

Let the set of homomorphisms $H = \{\{x/a\}, \{x/b\}\}$. Then we have $Chase_H(\Sigma, I) = \{T(a, X_1), T(b, X_2)\}$, where X_1 and X_2 are new null values. If we consider $H' = \{\{x/a\}, \{z/b\}\}$, we have $Chase_{H'}(\Sigma, I) = \{T(a, X_3),\ V(b, X_4)\}$, where X_3 and X_4 are new nulls.

The idea is to compute a recovery $Chase_H(\Sigma^{-1}, J)$, for each $H \in \mathsf{COV}(\Sigma, J)$, such that $H \models \mathsf{SUB}(\Sigma)$. However, as seen in the following example, this does not yet guarantee that the result is a recovery of J wrt Σ.

Example 6. *Let Σ and J be the set of s-t tgds and target instance from Example 2, and let $H_1 = \{h_1, h_2, h_3\}$. Then $H_1 \in \mathsf{COV}(\Sigma, J)$ and $H_1 \models \mathsf{SUB}(\Sigma)$. Nevertheless, $Chase_{H_1}(\Sigma^{-1}, J) = I'$, where $I' = \{R(a, a, X_1), R(X_2, X_3, c), R(X_4, X_5, d)\}$, is not a recovery of J wrt Σ because we have that $(I', J) \not\models \Sigma$.*

We note that $Chase(\Sigma, I') = J'$, where target instance $J' = \{S(a, a, Z_1),\ T(X_1), T(c), T(d)\}$. Also there are two homomorphisms g_1 and g_2 from J' to J, where homomorphisms $g_1 = \{Z_1/b, X_1/c\}$ and $g_2 = \{Z_1/b, X_1/d\}$. We can note that $g_1(I') = \{R(a, a, c), R(X_2, X_3, c), R(X_4, X_5, d)\}$ and $g_2(I') = \{R(a, a, d), R(X_2, X_3, c), R(X_4, X_5, d)\}$ are both recoveries of J.

This leads to the following algorithm.

Definition 9. *Let Σ be a set of source-to-target tgds and J a target instance valid for recovery under Σ. Suppose $\mathsf{COV}(\Sigma, J) = \{H_1, \ldots, H_n\}$. Let $Chase_{H_i}(\Sigma^{-1}, J) = I_i$, for each $H_i \models \mathsf{SUB}(\Sigma)$, and let $Chase(\Sigma, I_i) = J_i$. Furthermore, let $\{g_{i1}, \ldots, g_{im_i}\}$ be all homomorphisms from J_i to J that are the identity on $dom(J)$. Then define the set of instances*

$$Chase^{-1}(\Sigma, J) = \bigcup_{1 \leq i \leq n} \bigcup_{1 \leq j \leq m_i} \{g_{ij}(I_i)\}.$$

With this we are now ready to state the main theorem of this section.

Theorem 1. *Let Σ be a set of s-t tgds and J a target instance valid for recovery under Σ. Then*

$$Chase^{-1}(\Sigma, J) \subseteq \mathsf{REC}(\Sigma, J).$$

Proof. (sketch) It is easy to note that for any set of homomorphisms $H \in \mathsf{COV}(\Sigma, J)$ such that $H \models \mathsf{SUB}(\Sigma, J)$ we have that

$$Chase(\Sigma, Chase_H(\Sigma^{-1}, J)) \to J$$

with homomorphism h identity on the set $dom(J)$. On the other hand, from the construction of h and $Chase_H(\Sigma^{-1}, J)$, it is easy to see that $(h(Chase_H(\Sigma^{-1}, J)), J) \models \Sigma$. Finally, because $H \in \mathsf{COV}(\Sigma, J)$ it follows that each tuple from J is justified by some tuples in $h(Chase_H(\Sigma^{-1}, J))$. Note that h is non-identity only on new nulls created during the creation of $Chase_H(\Sigma^{-1}, J)$. $\square$

Intuitively, Theorem 1 shows that using the special chase $Chase^{-1}$ we get recoveries for Σ and J.

Example 7. *Let Σ and J be the set of s-t tgds and instance from Examples 2 and 3. For simplicity we will consider only the set of minimal covers in $\mathsf{COV}(\Sigma, J)$ which are: $H_1 = \{h_1, h_2, h_3\}$, $H_2 = \{h_1, h_2, h_5\}$, $H_3 = \{h_1, h_3, h_4\}$ and $H_4 = \{h_1, h_4, h_5\}$. As mentioned we have $H_4 \not\models \mathsf{SUB}(\Sigma)$, and for any $i \in \{1, 2, 3\}$, we have $H_i \models \mathsf{SUB}(\Sigma)$. Using the notation from Definition 9, $Chase^{-1}$ consists of the following instances:*

$$Chase_{H_1}(\Sigma^{-1}, J) = I_1 = \{R(a, a, X_1), R(X_2, X_3, c),\\ R(X_4, X_5, d)\};$$

$$Chase_{H_2}(\Sigma^{-1}, J) = I_2 = \{R(a, a, Y_1), R(Y_2, Y_3, c), D(Y_4, d)\};$$

$$Chase_{H_3}(\Sigma^{-1}, J) = I_3 = \{R(a, a, Z_1), R(Z_2, Z_3, d), D(Z_4, c)\}.$$

For the target instance $J_1 = Chase(\Sigma, I_1)$, where

$$J_1 = \{S(a, V_1), T(X_1), T(c), T(d)\},$$

we have $J_1 \to J$ with homomorphism $g_{11} = \{V_1/b, X_1/c\}$ and homomorphism $g_{12} = \{V_1/b, X_1/d\}$. Similarly, from source instance I_2 we get homomorphisms $g_{21} = \{V_2/b, Y_1/c\}$ and $g_{22} = \{V_2/b, Y_1/d\}$ and finally from I_3 we get homomorphisms $g_{31} = \{V_3/b, Z_1/c\}$ and $g_{32} = \{V_3/b, Z_1/d\}$.

Based on Definition 9 from these homomorphisms the set $Chase^{-1}(\Sigma, J)$ will contain the following recoveries:

$$\begin{aligned}
g_{11}(I_1) &= \{R(a,a,c), R(X_2,X_3,c), R(X_4,X_5,d)\},\\
g_{12}(I_1) &= \{R(a,a,d), R(X_2,X_3,c), R(X_4,X_5,d)\},\\
g_{21}(I_2) &= \{R(a,a,c), R(X_2,X_3,c), D(X_4,d)\},\\
g_{22}(I_2) &= \{R(a,a,d), R(X_2,X_3,c), D(X_4,d)\},\\
g_{31}(I_3) &= \{R(a,a,c), R(X_2,X_3,d), D(X_4,c)\},\\
g_{32}(I_3) &= \{R(a,a,d), R(X_2,X_3,d), D(X_4,c)\}
\end{aligned}$$

Note that the theorem 1 does not guarantee that all possible recoveries are computed. Following Example 2 it is easy to see $I = \{R(a,a,c), R(a,a,d), D(e,c)\} \in \mathsf{REC}(\Sigma, J)$ but obviously $I \notin Chase^{-1}(\Sigma, J)$. Nevertheless, the next theorem shows that the set of instances $Chase^{-1}(\Sigma, J)$ are sufficient for computing the certain answer for any UCQ query.

Theorem 2. *Let Σ be a set of s-t tgds and target instance J valid for recovery under Σ, then $Chase^{-1}(\Sigma, J)$ is a UCQ-universal recovery of J under Σ.*

Proof. (sketch) Given two sets of instances $\mathcal{L}$ and $\mathcal{K}$, we write $\mathcal{K} \to \mathcal{L}$ iff $(\forall J \in \mathcal{L} \;\; \exists I \in \mathcal{K} \; : \; I \to J)$. In case $\mathcal{K} \to \mathcal{L}$ and $\mathcal{L} \to \mathcal{K}$, we say that $\mathcal{K}$ and L are homomorphically equivalent and denoted it by $\mathcal{K} \leftrightarrow \mathcal{L}$. In [11] it was shown that $\mathsf{CERT}(Q, \mathcal{K}) = \mathsf{CERT}(Q, \mathcal{L})$ for any UCQ query Q iff $\mathcal{K} \leftrightarrow \mathcal{L}$. It follows that we only need to show that $\mathsf{REC}(\Sigma, J) \leftrightarrow Chase^{-1}(\Sigma, J)$. From Theorem 1 it follows that $\mathsf{REC}(\Sigma, J) \to Chase^{-1}(\Sigma, J)$. Now let $I \in \mathsf{REC}(\Sigma, J)$. Because $Chase^{-1}$ uses sets of homomorphisms that cover the target instance and also from the way $Chase^{-1}$ is constructed, it follows that there exists an instance $I_i \in Chase^{-1}(\Sigma, J)$ such that $I_i \to I$. $\square$

5. COMPLEXITY

We first show that testing if a target instance J is valid for recovery under Σ is NP-complete in the number of tuples in J. We have

Problem:	J-validity.
Parameter:	Set Σ of s-t tgds.
Input:	Target instance J
Question:	Is J valid for recovery under Σ?

Theorem 3. *The J-validity problem in NP-complete.*

Proof. (sketch) Upper bound: It is easy to verify that an instance J is valid for recovery wrt Σ iff there exists a $H \in \mathsf{COV}(\Sigma, J)$, such that $H \vDash \mathsf{SUB}(\Sigma, J)$. Thus one may simply guess a subset H of $\mathsf{HOM}(\Sigma, J)$. The size of H is at most $m \cdot n^k$, where n is the number of tuples in J, m is the number of tgds in Σ, and k is the maximum number of variables in the head of any tgd in Σ. It remains to test if $H \in \mathsf{COV}(\Sigma, J)$ and if $H \vDash \mathsf{SUB}(\Sigma, J)$. It is easy to see that both tests can be done in time polynomial in n.

The lower bound can be inferred directly from the "view consistency" problem that is shown in [1] to be NP-hard even when the view consists of a single GAV dependency. $\square$

The reduction above can also be used to prove the following proposition.

Proposition 1. *Let Σ be a fixed set of s-t tgds and J a target instance. Testing whether J is a universal solution for some source instance I under Σ is NP-complete in the number of tuples in J.*

We saw in the previous section that given a UCQ Q, we can find the certain answer $\mathsf{CERT}(Q, \Sigma, J)$ by computing the set $Chase^{-1}(\Sigma, J)$, and then evaluate Q on this set. It seems that there can be an exponential blow-up when going from J to $Chase^{-1}(\Sigma, J)$. This raises the question whether the blow-up is avoidable. In this section we show that the answer is "no" (assuming P≠NP), by considering the following decision problem.

Problem:	Q-certainty.
Parameters:	Set Σ of s-t tgds, query Q.
Input:	Tuple t, target instance J valid for recovery under Σ.
Question:	Is $t \in \mathsf{CERT}(Q, \Sigma, J)$?

We shall see that the problem is coNP-complete when Q is a CQ or a UCQ.

Theorem 4. *Let Q be a CQ. Then the Q-certainty problem is coNP-complete.*

Proof. (sketch) For the upper bound we note that the answer is "no" if and only if there exists an instance I in $Chase^{-1}(\Sigma, J)$, such that $t \notin Q(I)$. The instance I can be guessed as follows.

First we generate $\mathsf{SUB}(\Sigma)$. This can be done in polynomial time since the size of Σ is a constant. Then we compute $\mathsf{HOM}(\Sigma, J)$ in time $\mathcal{O}(m \cdot n^k)$, where n is the number of tuples in J, m is the number of tgds in Σ, and k is the maximum number of variables in the head of any tgd in Σ. We also compute, in polynomial time, the instance $Chase_{\mathsf{HOM}(\Sigma,J)}(\Sigma^{-1}, J)$. Let

$$N = Nulls(Chase_{\mathsf{HOM}(\Sigma,J)}(\Sigma^{-1}, J)) \smallsetminus Nulls(J)$$

be the set of "new" nulls. Now the size of N is bounded by $j \cdot m \cdot n^k$, where j is the maximum number of variables in the body of a dependency in Σ. We then guess a mapping h from the new nulls N to $dom(J)$, and guess a subset H of $\mathsf{HOM}(\Sigma, J)$. Before going to the verification phase, we compute $I = Chase_H(\Sigma^{-1}, J) \subseteq Chase_{\mathsf{HOM}(\Sigma,J)}(\Sigma^{-1}, J)$, as well as $Chase(\Sigma, I)$, both in time polynomial in n. It remains to verify that

1. $H \in \mathsf{COV}(\Sigma, J)$,
2. $H \vDash \mathsf{SUB}(\Sigma)$,
3. $\exists$ extension h' of h, identity on $dom(Chase(\Sigma, I)) \smallsetminus N$, such that $h'(Chase(\Sigma, I)) \subseteq J$, and
4. $t \notin h'(I)$.

It is straightforward to show that steps 1, 2 and 4 can be done in time polynomial in n. Step 3 can also be performed in polynomial in n time as Σ is a set of s-t tgds. If all verifications succeed, the answer to the Q-certainty problem in "no," which means that the problem of Q-certainty is in coNP.

The lower bound follows again from [1] where it was shown that already the special case of certain answers to CQs Q using materialized views under CWA when the view is defined as a GAV dependency, is coNP hard. $\square$

We note that the same lower bound and upper bound hold even if Q is a UCQ. We therefore have:

COROLLARY 1. *Let Q be a UCQ. Then the Q-certainty problem is coNP-complete.*

6. TRACTABLE CASES

As saw in the previous section, recovery and certain answer evaluation of CQ and UCQ queries is intractable in the general case. We will now look into some special cases that guarantee tractability. First we give criteria for Σ and J that are sufficient for $\mathsf{REC}(\Sigma, J)$ to contain a unique recovery computable in polynomial time. Such a unique recovery is naturally UCQ-complete. We also give a polynomial time algorithm that materializes a unique instance that gives sound answers to any CQ query.

6.1 Unique recoveries

We shall first look at an example that illustrates the intuition behind unique recoveries.

EXAMPLE 8. *Consider the following dependency that describes a schema evolution in a company.*

Emp(Name, Dept), Bnf(Dept, Benefit) →
EmpDept(Name, Dept),EmpBnf(Emp, Benefit).

The Emp relation records the department where each employee works, and the Bnf relation lists the benefits (medical insurance, pension contributions, profit sharing, etc.) for all the employees of the given department.

After the data exchange for the new schema is carried out, the company changes its policy to allow an employee to work for more than one department. It therefore decides to restore the database according to the old schema.

It is easy to see that the set $\mathsf{SUB}(\Sigma)$ contains only one constraint, stating that a department gives the same set of benefits to each of its employees.

$$\{Name/v_1, Dept/w, Benefit/u_1\},$$
$$\{Name/v_2, Dept/w, Benefit/u_2\} \rightarrow$$
$$\{Name/v_1, Dept/w, Benefit/u_2\}.$$

Suppose the target instance J is as follows:

J
EmpDept(Joe, HR)
EmpDept(Bill, Sales)
EmpDept(Sue, HR)
EmpBnf(Joe, medical)
EmpBnf(Joe, pension)
EmpBnf(Bill, medical)
EmpBnf(Bill, profit)
EmpBnf(Sue, medical)
EmpBnf(Sue, pension)

It is easy to see that J is valid for recovery under the given dependency and also that it has a unique recovery I:

I
EmpDept(Joe, HR)
EmpDept(Bill, Sales)
EmpDept(Sue, HR)
Bnf(HR, medical)
Bnf(HR, pension)
Bnf(Sales, medical)
Bnf(Sales, profit)

By querying this source instance one may obtain both sound and complete answers. Note that in this example the maximal recovery mapping Σ' [8] for Σ is the same as the maximum CQ recovery mapping [6] for Σ, that is:

EmpDept(Name, Dept) →
∃x Emp(Name, Dept), Bnf(Dept, x);
EmpBnf(Name, Benefit) →
∃y Emp(Name, y), Bnf(y, Benefit).

If one now is interested in the benefits of the HR department, that is the conjunctive query Q = Bnf(HR,x), evaluating Q on $Chase(\Sigma', J)$ yields an empty certain answer, whereas using the instance based recovery I the certain answer is {medical, pension}.

We can now introduce the notion of a complete UCQ recovery.

DEFINITION 10. *Let Σ be a set of s-t tgds and J an instance valid for recovery under Σ. A source instance I is said to be a complete UCQ recovery for Σ and J, if $Q(I)\downarrow = \mathsf{CERT}(Q, \Sigma, J)$, for all UCQ queries Q.*

Before presenting a sufficient condition that guarantees the existence of a complete UCQ recovery we need to introduce the following lemma:

LEMMA 1. *Let Σ be a set of s-t tgds such that all the constraints in $\mathsf{SUB}(\Sigma)$ were constructed using only quasi-guarded tgds from Σ. Then*

$$|Chase^{-1}(\Sigma, J)| \leq |\mathsf{COV}(\Sigma, J)|,$$

for any target instance J valid for recovery under Σ.

PROOF. (sketch) The lemma follows from the observation that in case $\mathsf{SUB}(\Sigma)$ is constructed using only quasi-guarded tgds, then $Chase(\Sigma, Chase_H(\Sigma^{-1}, J))$, for $H \in \mathsf{COV}(\Sigma, J)$ and $H \models \mathsf{SUB}(\Sigma)$, will not contain any new nulls generated by the $Chase_H(\Sigma^{-1}, J)$ process. □

Intuitively, the previous lemma states that each covering from $\mathsf{COV}(\Sigma, J)$ that is a model for $\mathsf{SUB}(\Sigma)$ will only generate one recovery using the inverse chase process. Recall that in general, based on Definition 9, for each covering H there maybe an exponential number of recoveries. Consider for example s-t tgds $\Sigma = \{R(x, y) \rightarrow S(x);\ R(u, v) \rightarrow T(v)\}$ and target instance $J = \{S(a), S(b), T(c), T(d)\}$. It is easy to see that with this configuration we have $|\mathsf{COV}(\Sigma, J)| = 1$ and $|Chase^{-1}(\Sigma, J)| = 7$. We say that a set of s-t tgds Σ with the properties from Lemma 1 is *quasi-guarded safe.*

Based on Lemma 1, it seems that a sufficient condition that guarantees the existence of a complete UCQ recovery for a given set of s-t tgd Σ and target instance J, is that there exists only one covering for J under Σ, and that Σ is quasi-guarded safe. The following theorem confirms this intuition.

THEOREM 5. *Let Σ be a set of s-t tgds and J a target instance valid for recovery under Σ. Then there exists an instance I that is a complete UCQ recovery for Σ and J if*

1. $|\mathsf{COV}(\Sigma, J)| = 1$, *and*
2. Σ *is quasi-guarded safe.*

Moreover, the complete UCQ instance I can be computed in time polynomial in the number of tuples in J.

PROOF. (Sketch) Based on Lemma 1 it can be easily noted that the computation of $Chase^{-1}(\Sigma, J)$ becomes deterministic and that it computes a complete UCQ recovery. $\square$

In Example 8 the set of homomorphism constraints $\mathsf{SUB}(\Sigma)$ was constructed from the only tgd in Σ, which was both a full and quasi-guarded. It is easy to see that in case $\mathsf{REC}(\Sigma, J) = \{I\}$, one may use the recovered instance I to obtain both sound and complete answers to any query. Note that the existence of a complete UCQ recovery for Σ and J does not guarantee that $|\mathsf{REC}(\Sigma, J)| = 1$. For this, let $\Sigma = \{R(x, y) \to S(x)\}$ and $J = \{S(a), S(b), S(c)\}$. In this case there are an infinite number of recoveries but there exists a complete UCQ recovery $I = \{R(a, X_1), R(b, X_2), R(c, X_3)\}$.

The second condition of Theorem 5 is clearly easy to check. The following theorem gives us a necessary and sufficient condition for the first condition to hold.

THEOREM 6. *Let Σ be a set of s-t tgd and J a target instance valid for recovery under Σ. Then $|\mathsf{COV}(\Sigma, J)| = 1$ iff for all $h \in \mathsf{HOM}(\Sigma, J)$ there exists a tuple $t \in J$ such that $t \in h(head(\xi_h))$, and for any homomorphism $h' \in \mathsf{HOM}(\Sigma, J)$, where $h' \neq h$, we have $t \notin h'(head(\xi_{h'}))$.*

PROOF. (sketch) For the if direction, it is easy to see that a covering needs to contain all homomorphisms from $\mathsf{HOM}(\Sigma, J)$, in order to contain all the tuples from J. For the only if direction, if $|\mathsf{COV}(\Sigma, J)| = 1$ the covering contains all the homomorphisms from $\mathsf{HOM}(\Sigma, J)$. This means that by removing any of the homomorphisms, J will not be fully covered. Thus, there exists a tuple covered only by that homomorphism. $\square$

Based on this theorem it is easy to see that testing if $|\mathsf{COV}(\Sigma, J)| = 1$ can be done in quadratic time. There are two important observations to be made:

First, in order to cover more cases, instead of a unique recovery that is UCQ-complete, one can compute a set of $k > 1$ recoveries that is complete for UCQ queries, for a fixed constant k. In Example 8, suppose the target instance has only one employee working for two departments, and each of these two departments offers exactly one benefit (and all the other employees work for exactly one department). In this case there exists a set of two source recoveries $\{I_1, I_2\}$ such that the certain answer $Q(I_1) \cap Q(I_2) = \mathsf{CERT}(Q, \Sigma, J)$, for all UCQ queries Q. By changing the first condition of Theorem 5 to $|\mathsf{COV}(\Sigma, J)| \leq k$, for a fixed k, we can keep the tractability result for a larger class of pairs Σ and J.

Second, even if the first condition Theorem 5 is not satisfied, one may find, in polynomial time, a unique maximal subset J' of J with this property. From J' one can compute a source instance that can be used to get sound answers to any UCQ query. Formally, we have the following theorem.

THEOREM 7. *Let Σ be a set of s-t tgds, and J a target instance valid for recovery under Σ. There is a quadratic (in the number of tuples in J) algorithm that computes a maximal subset J' of J, such that $|\mathsf{COV}(\Sigma, J')| = 1$. Based on J' one may compute in polynomial time a source instance I, such that $Q(I)\!\downarrow \subseteq \mathsf{CERT}(Q, \Sigma, J)$, for all UCQ queries Q.*

PROOF. (sketch) For the first part we compute a set K that contains all tuples from J that are covered by only one homomorphism in $\mathsf{HOM}(\Sigma, J)$. After this, instance J' is constructed as $J' = \cup_{t \in K} h_t(head(\xi_{h_t}))$, where h_t is the unique homomorphisms that covers tuple t. For the second part, it can be verified that there exists $I \in \mathsf{REC}(\Sigma, J)$ such that $I' \subseteq I$, where $I' = Chase^{-1}(\Sigma, Chase_H(\Sigma^{-1}, J'))$, and that for all $I'' \in \mathsf{REC}(\Sigma, J)$ we have $I' \to I''$. $\square$

EXAMPLE 9. *Let $\Sigma = \{\xi_1, \xi_2\}$, where*

$$\xi_1 = R(x, y) \to S(x), S(y);$$
$$\xi_2 = D(z) \to T(z).$$

Consider target instance $J = \{S(a), S(b), T(c), T(d)\}$. In this case $J' = \{T(c), T(d)\}$ is the maximal subset of J, such that $|\mathsf{COV}(\Sigma, J')| = 1$. Note that $\mathsf{SUB}(\Sigma) = \varnothing$. The source instance $I = \{D(c), D(d)\}$, that is not a recovery, can be used to get sound answers to any UCQ query. For example for conjunctive query $Q(x) = D(x)$, the result will be $\{c, d\}$.

6.2 Sound CQ answers

In this section we will show that even without any restrictions on the mapping or the target instance one may obtain in polynomial time sound certain answers for any source CQ query. For this we will present a tractable algorithm that computes a "sub-universal" source instance that can be used to obtain the sound answers. Let us first introduce some notation.

Let I_1 and I_2 be instances. A *homomorphic greatest lower bound* of I_1 and I_2, denoted $glb\{I_1, I_2\}$, is an instance K, such that $K \to I_1$ and $K \to I_2$, and for all instances L, if $L \to I_1$ and $L \to I_2$, then $L \to K$. This lower bound can be computed as follows [19, 21]: Let ι be an injective mapping from $Nulls \cup Cons$ to $Nulls \cup Cons$, such that

- $\iota(x, x) = x$ for any $x \in Nulls \cup Cons$, and
- $\iota(x, y) = z$ for any $x, y \in Nulls \cup Cons$, with $x \neq y$ and z a new null from $Nulls$.

Initialize $glb\{I_1, I_2\}$ to be the empty instance. For all pairs of tuples

- $(R(x_1, x_2, \ldots, x_k), R(y_1, y_2, \ldots, y_k)) \in I_1 \times I_2$,

the tuple

- $R(\iota(x_1, y_1), \iota(x_2, y_2), \ldots, \iota(x_k, y_k))$

is added to instance $glb\{I_1, I_2\}$. It is easily shown that in case I_1 and I_2 are two ground instances, then we have $Q(glb\{I_1, I_2\})\!\downarrow = Q(I_1) \cap Q(I_2)$, for all CQ queries Q. The lower bound is extended to larger sets recursively by the equation $glb\{I_1, I_2, \ldots, I_n\} = glb\{glb\{I_1, I_2, \ldots, I_{n-1}\}, I_n\}$.

Let Σ be a set of s-t tgds, J a target instance valid for recovery under Σ, and $h \in \mathsf{HOM}(\Sigma, J)$. The set of tuples in instance $h(head(\xi_h))$ is denoted J_h. Note that $J_h \subseteq J$. With I_h we then denote the source instance $Chase_{\{h\}}(\Sigma^{-1}, J)$. This notation is extended to sets of homomorphisms $H \subseteq \mathsf{HOM}(\Sigma, J)$, by $J_H = \bigcup_{h \in H} J_h$ and $I_H = \bigcup_{h \in H} I_h$.

Definition 11. *Let Σ be a set of s-t tgds, J a target instance valid for recovery under Σ, and h a homomorphism from $\mathsf{HOM}(\Sigma, J)$. A set of homomorphisms $H \subseteq \mathsf{HOM}(\Sigma, J)$ is said to be a minimal covering for h under Σ and J if*

1. *$J_h \subseteq J_H$, and*
2. *there is no $H' \subset H$ such that $J_h \subseteq J_{H'}$.*

We denote the set of all the minimal coverings for h under Σ and J with $\mathsf{COV}_h(\Sigma, J)$.

Note that based on the previous definition we have that $\{h\} \in \mathsf{COV}_h(\Sigma, J)$, for any $h \in \mathsf{HOM}(\Sigma, J)$. Intuitively any minimal covering $H \in \mathsf{COV}_h(\Sigma, J)$ for a homomorphism h represents an alternative way of obtaining tuples J_h from a source instance in a chase process.

Example 10. *Consider the following set of source-to-target tgds $\Sigma = \{\xi_1, \xi_2\}$, where:*

$$\xi_1 = R(x, y) \to S(x);$$
$$\xi_2 = R(z, v) \to S(z), T(v).$$

For target instance $J = \{S(a), T(b_1), \ldots, T(b_n)\}$ we have the set $\mathsf{HOM}(\Sigma, J) =$

$$\{h = \{x/a\},\ h_1 = \{z/a, v/b_1\},\ \ldots,\ h_n = \{z/a, v/b_n\}\}.$$

Based on the previous definition,

$$\mathsf{COV}_h(\Sigma, J) = \{\{h\}, \{h_1\}, \ldots, \{h_n\}\}$$

and $\mathsf{COV}_{h_i}(\Sigma, J) = \{\{h_i\}\}$ for all $i \in \{1, \ldots, n\}$.

From the above example it can be observed that the size of the set $\mathsf{COV}_h(\Sigma, J)$ may be polynomial in the size of target instance J. As we will see next, in order to materialize a CQ-universal source instance we need to compute the *glb* of the source instances generated by homomorphisms from $\mathsf{HOM}(\Sigma, J)$. On the other hand, in [21] it is shown that the size of the *glb* can be exponential in the number of instances. Consequently, in order to keep the tractability of this method, we need to reduce the size of the set for the considered instances.

As an intuition on how the polynomial number of instances can be reduced, note in the previous example that only variable z plays a role in covering homomorphism h, and instead of using source instances $\{R(a, b_1)\}$, $\{R(a, b_2)\}$ to $\{R(a, b_n)\}$ in the *glb* computation, we may use the more generic instance $\{R(a, X)\}$ that was obtain by replacing with new nulls all variables that do not contribute to the covering. In the next paragraphs we introduce some tools needed in this approach.

For a homomorphism h and sequence $\bar{x}$ of variables, with $h|_{\bar{x}}$ we denote the homomorphism h restricted to the variables in $\bar{x}$. Let Σ be a set of s-t tgd's, J a target instance, and h a homomorphism in $\mathsf{HOM}(\Sigma, J)$. Let H and G be sets of homomorphisms in $\mathsf{COV}_h(\Sigma, J)$. The H is said to be *equivalent with G wrt h and* Σ, denoted with $H \equiv_{(h,\Sigma)} G$, if the following conditions hold:

1. $|H| = |G|$, and
2. there exists orderings $h_1, \ldots, h_k$ and $g_1, \ldots, g_k$ of the mappings in H and G, respectively, and k sequences of variables $\bar{x}_1, \ldots, \bar{x}_k$, such that
   - $J_h = \bigcup_{i \in \{1,\ldots,k\}} J_{h_{i|\bar{x}_i}}$, and
   - $J_{h_{i|\bar{x}_i}} = J_{g_{i|\bar{x}_i}}$,

   for all $i \in \{1, \ldots, k\}$.

It is easy to see that $\equiv_{(h,\Sigma)}$ is an equivalence relation. For a set $H \in \mathsf{COV}_h(\Sigma, J)$, with $H_{(h,\Sigma)}$ we denote the set of representatives of the equivalence classes generated by $\equiv_{(h,\Sigma)}$. For a homomorphism $h_i \in H_{(h,\Sigma)}$, with $J_{h_i(h,\Sigma)}$ we denote the instance $f_i(head(\xi_{h_i}))$, where f_i is an extension of $h_{i|\bar{x}_i}$ that assigns a new null value to each variable from $head(\xi_{h_i})$ distinct from the variables in $\bar{x}_i$. We then define the source instance $I_{h_i(h,\Sigma)} = Chase_f(\Sigma^{-1}, J)$ and finally $I_{H(h,\Sigma)} = \bigcup_{h_i \in H_{(h,\Sigma)}} I_{h_i(h,\Sigma)}$.

Example 11. *Given the configuration in Example 10 we have $\{h_i\} \equiv_{(h,\Sigma)} \{h_j\}$, for all $i, j \in \{1, \ldots, n\}$. Thus we have $J_{h_i(h,\Sigma)} = \{S(a), T(X_i)\}$, where X_i is a new fresh null. Finally, $I_{\{h_i\}(h,\Sigma)} = \{R(a, X_i)\}$.*

Definition 12. *Let Σ be a set of s-t tgds and target instance J valid for recovery under Σ. Then*

$$I_{\Sigma,J} = \bigcup_{h \in \mathsf{HOM}(\Sigma,J)} glb\{I_{H(h,\Sigma)} : H \in \mathsf{COV}_h(\Sigma, J)\}.$$

The source instance $I_{\Sigma,J}$ can be computed in polynomial time in the size of J as stated below.

Theorem 8. *Let Σ be a set of s-t tgds and J a target instance valid for recovery under Σ. Then the instance $I_{\Sigma,J}$ can be computed in time $\mathcal{O}(n^m j^{k^2 \ell})$, where n is the size of the domain of J, m is the maximum number of variables occurring in the head of a tgd from Σ, j the maximum number of atoms in the body of a tgd in Σ, k is the largest number of atoms that occur in the head of a tgd in Σ, and ℓ is the total number of tgds in Σ.*

Proof. (sketch) It is easy to see that the size of the set $\mathsf{HOM}(\Sigma, J)$ is bounded by n^m. Also for each homomorphism $h \in \mathsf{HOM}(\Sigma, J)$ we have $|J_h| < k$ and thus it follows that $|\{I_{H(h,\Sigma)} : H \in \mathsf{COV}_h(\Sigma, J)\}| \leq k^2 \ell$. With this we have $|glb(\{I_{H(h,\Sigma)} : H \in \mathsf{COV}_h(\Sigma, J)\})| \leq j^{k^2 \ell}$ and since the set $\mathsf{HOM}(\Sigma, J)$ is bounded by n^m, the claim of the theorem follows. □

It can be easily observed that in general $I_{\Sigma,J}$ is not a recovery for J under Σ, see Example 12 below. The following theorem shows what makes the source instance $I_{\Sigma,J}$ "CQ sub-universal."

Theorem 9. *Let Σ be a set of s-t tgds and J a target instance valid for recovery under Σ. Then for all recovered instances $I \in \mathsf{REC}(\Sigma, J)$ we have that $I_{\Sigma,J} \to I$. We also have $Q(I_{\Sigma,J})\downarrow \subseteq \mathsf{CERT}(Q, \Sigma, J)$, for all CQ queries Q.*

Proof. (sketch) Let $I \in \mathsf{REC}(\Sigma, J)$ and $h \in \mathsf{HOM}(\Sigma, J)$. From the definition of $\mathsf{REC}(\Sigma, J)$ it follows that there exists an instance $I' \subseteq I$ such that $Chase(\xi_h, I') \to J_h$. For the source instance $I'_h = glb(\{I_{H(h,\Sigma)} : H \in \mathsf{COV}_h(\Sigma, J)\})$, we have $I'_h \to I'$. As h was arbitrarily chosen it follows that $I_{\Sigma,J} = \cup_{h \in \mathsf{HOM}(\Sigma,J)} I'_h \to I$. $\Box$

The example below illustrates the construction.

Example 12. *Let $\Sigma = \{\xi_1, \xi_2, \xi_3\}$ be the set of s-t tgds where:*

$$\xi_1 = R(x, y) \to T(x);$$
$$\xi_2 = U(z) \to S(z);$$
$$\xi_3 = R(v, v) \to T(v), S(v).$$

Consider target instance $J = \{T(a), S(a), S(b)\}$. Clearly J is valid for recovery under Σ. The set of homomorphisms $\mathsf{HOM}(\Sigma, J)$ is $\{h_1 : \{x/a\}, h_2 : \{z/a\}, h_3 : \{z/b\}, h_4 : \{v/a\}\}$. We then have

$$\begin{aligned}
&\mathsf{COV}_{h_1}(\Sigma, J) = \{\{h_1\}, \{h_4\}\},\\
&\mathsf{COV}_{h_2}(\Sigma, J) = \{\{h_2\}, \{h_4\}\},\\
&\mathsf{COV}_{h_3}(\Sigma, J) = \{\{h_3\}\}, \text{ and}\\
&\mathsf{COV}_{h_4}(\Sigma, J) = \{\{h_4\}, \{h_1, h_2\}\}.
\end{aligned}$$

Clearly these are not equivalent homomorphisms sets. For simplicity, we will use the set itself to represent its equivalence class. Corresponding to these homomorphisms we have: $I_{\{h_1\}} = \{R(a, X_1)\}$, $I_{\{h_2\}} = \{U(a)\}$, $I_{\{h_3\}} = \{U(b)\}$, $I_{\{h_4\}} = \{R(a, a)\}$ and finally $I_{\{h_1,h_2\}} = \{R(a, X_2), U(a)\}$, where X_1 and X_2 are distinct null values. Thus, the "CQ sub-universal" instance will be:

$$\begin{aligned}
I_{\Sigma,J} =\,& glb\{I_{\{h_1\}}, I_{\{h_4\}}\} \cup glb\{I_{\{h_2\}}, I_{\{h_4\}}\} \cup\\
& glb\{I_{\{h_3\}}\} \cup glb\{I_{\{h_4\}}, I_{\{h_1,h_2\}}\}\\
=\,& \{R(a, Y_1), U(b), R(a, Y_2)\}
\end{aligned}$$

where Y_1 and Y_2 are distinct null values. Note that even though $(I_{\Sigma,J}, J) \vDash \Sigma$, the source instance $I_{\Sigma,J}$ is not a recovery for J under Σ, as tuple $S(a)$ from J is not justified by any tuples from $I_{\Sigma,J}$. For the query $Q_1(x) = \{(x) : U(x)\}$, we have $Q_1(I_{\Sigma,J})\downarrow = \{(b)\} \subseteq \mathsf{CERT}(Q_1, \Sigma, J)$. To show that the method is not complete, consider query $Q_2(x) = \{(x) : R(x, x)\}$. For this we have $\mathsf{CERT}(Q_2, \Sigma, J) = \{(a)\}$ and $Q_2(I_{\Sigma,J})\downarrow = \varnothing$.

One may note that for the construction of instance $I_{\Sigma,J}$ the subsumption constraints are not considered. It is still an open problem if one may filter, in polynomial time, the coverings used in Definition 12 based on the subsumption constraint in order to get more sound answers to CQ queries.

As shown in [6], the CQ-maximum recovery mapping can be represented as a set of target-to-source tgds. The following theorem shows that for any set of s-t tgds Σ, target instance J and CQ query Q, the "sub-universal" source instance will return at least the same sound CQ answers over the recoveries as one would obtain by chasing J with the CQ-maximum recovery mapping Σ' obtained from Σ.

Theorem 10. *Let Σ be a set of s-t tgds, J a target instance valid for recovery under Σ, and let Σ' be the CQ-maximum recovery mapping [6] for Σ. Then it holds that $Q(Chase(\Sigma', J))\downarrow \subseteq Q(I_{\Sigma,J})\downarrow$, for all $Q \in CQ$.*

Proof. (sketch) Let J be a target instance and Σ' be the CQ-maximum recovery mapping for Σ. It can be easily noted, from the way Σ' is constructed [6], that for any homomorphism f from the body of a tgd $\xi' \in \Sigma'$ into J, one can find homomorphism $h \in \mathsf{HOM}(\Sigma, J)$ such that we have $f(body(\xi')) \subseteq h(head(\xi_h))$, and for any set $H \in \mathsf{COV}_h(\Sigma, J)$, $Chase_f(\xi', J) \to I_{H(h,\Sigma)}$. This means that there is a homomorphism $Chase_{\{f\}}(\xi', J) \to glb\{I_{H(h,\Sigma)} : H \in \mathsf{COV}_h(\Sigma, J)\}$. Extending this to the entire target instance and all dependencies in Σ', it follows that $Chase(\Sigma', J) \to I_{\Sigma,J}$. From this the statement from the theorem follows directly. $\Box$

Next example shows that there exists configurations of Σ, J, and Q such that using instance $I_{\Sigma,J}$ we may get strictly more sound information than obtained by chasing J with the CQ-maximum recovery mapping Σ'.

Example 13. *Consider the same setting as in Example 12. For Σ, the corresponding CQ-maximum recovery mapping is logically equivalent with $\Sigma' = \{T(x) \to \exists z\, R(x, z)\}$. For target instance J, the source instance obtained by chasing J with Σ' is $I = \{R(a, Z_1)\}$, where Z_1 is a null value. Considering the conjunctive query $Q_3(x) = \{(x) : U(x)\}$, we have $\varnothing = Q_3(I)\downarrow \subset Q_3(I_{\Sigma,J})\downarrow = \{(b)\}$.*

7. CONCLUSIONS

In this paper we proposed a new semantics for the inversion problem in data-exchange. We argue that our instance based recovery is more useful than the previously proposed inversion mappings, as one may recover not only more source data but also only sound data. We introduced a new chase algorithm to compute these recoveries when the initial mapping is given by a set of s-t tgds. The new chase algorithm permits the computation of a finite set of recovered source instances that may be used to get sound and complete answers to any UCQ query over the source schema. On the negative side we showed that getting these answers is coNP-complete even when considering only source CQ queries. Our approach does not have the restriction of only allowing ground instances. We think that our semantics opens the door to new interesting problems, such as finding recoveries after the target instance already has been altered by some operations (in the current work we only consider target instances that were valid for recovery). Another interesting problem is to find syntactical characterizations for the initial source-to-target mapping that will allow tractable computation for CQ (or UCQ) queries over the materialized set of source recoveries.

Acknowledgements

We are grateful to the anonymous referees for detailed and constructive comments.

8. REFERENCES

[1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–263, 1998.

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

[3] M. Arenas, R. Fagin, and A. Nash. Composition with target constraints. In *ICDT*, pages 129–142, 2010.

[4] M. Arenas, R. Fagin, and A. Nash. Composition with target constraints. *Logical Methods in Computer Science*, 7(3), 2011.

[5] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Composition and inversion of schema mappings. *SIGMOD Record*, 38(3):17–28, 2009.

[6] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Inverting schema mappings: Bridging the gap between theory and practice. *PVLDB*, 2(1):1018–1029, 2009.

[7] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Query language-based inverses of schema mappings: semantics, computation, and closure properties. *VLDB J.*, 21(6):823–842, 2012.

[8] M. Arenas, J. Pérez, and C. Riveros. The recovery of a schema mapping: Bringing exchanged data back. *ACM Trans. Database Syst.*, 34(4), 2009.

[9] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4):718–741, 1984.

[10] P. A. Bernstein. Applying model management to classical meta data problems. In *CIDR*, 2003.

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

[12] R. Fagin. Inverting schema mappings. *ACM Trans. Database Syst.*, 32(4), 2007.

[13] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

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

[15] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Quasi-inverses of schema mappings. *ACM Trans. Database Syst.*, 33(2), 2008.

[16] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Reverse data exchange: Coping with nulls. *ACM Trans. Database Syst.*, 36(2):11, 2011.

[17] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Schema mapping evolution through composition and inversion. In Z. Bellahsene, A. Bonifati, and E. Rahm, editors, *Schema Matching and Mapping*, Data-Centric Systems and Applications, pages 191–222. Springer, 2011.

[18] G. Grahne and A. Onet. Representation systems for data exchange. In *ICDT*, pages 208–221, 2012.

[19] P. Hell and J. Nesetril. *Graphs And Homomorphisms.* Oxford University Press, 2004.

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

[21] L. Libkin. Incomplete information and certain answers in general data models. In *PODS*, pages 59–70, 2011.

[22] A. Nash, P. A. Bernstein, and S. Melnik. Composition of mappings given by embedded dependencies. *ACM Trans. Database Syst.*, 32(1):4, 2007.

[23] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.